Control Engineering and Its Applications

Karyn Lavender

First Edition, 2012

ISBN 978-81-323-3036-3

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Introduction

Chapter 1 - Control System

Chapter 2 - Applications of Multiple Coordinate Systems

Chapter 3 - Model-Based Design

Chapter 4 - Furuta Pendulum

Chapter 5 - Aircraft Flight Control System

Chapter 6 - Distributed Control System

Chapter 7 - High Redundancy Actuation

Chapter 8 - Fault-Tolerant System

Chapter 9 - Control Theory

Chapter 10 - Embedded System

Chapter 11 - Engine Control Unit

Chapter 12 - Digifant Engine Management System

Chapter 13 - Orifice Plate

Introduction

Control systems play a critical role in space flight

Control engineering or **Control systems engineering** is the engineering discipline that applies control theory to design systems with predictable behaviors. The practice uses sensors to measure the output performance of the device being controlled (often a

vehicle) and those measurements can be used to give feedback to the input actuators that can make corrections toward desired performance. When a device is designed to perform without the need of human inputs for correction it is called automatic control (such as cruise control for regulating a car's speed). Multi-disciplinary in nature, control systems engineering activities focus on implementation of control systems mainly derived by mathematical modeling of systems of a diverse range.

Overview

Modern day control engineering (also called control systems engineering) is a relatively new field of study that gained a significant attention during 20th century with the advancement in technology. It can be broadly defined as practical application of control theory. Control engineering has an essential role in a wide range of control systems, from simple household washing machines to high-performance F-16 fighter aircraft. It seeks to understand physical systems, using mathematical modeling, in terms of inputs, outputs and various components with different behaviors; use control systems design tools to develop controllers for those systems; and implement controllers in physical systems employing available technology. A system can be mechanical, electrical, fluid, chemical, financial and even biological, and the mathematical modeling, analysis and controller design uses control theory in one or many of the time, frequency and complex-s domains, depending on the nature of the design problem.

History

Automatic control Systems were first developed over two thousand years ago. The first feedback control device on record is thought to be the ancient water clock of Ktesibios in Alexandria Egypt around the third century B.C. It kept time by regulating the water level in a vessel and, therefore, the water flow from that vessel. This certainly was a successful device as water clocks of similar design were still being made in Baghdad when the Mongols captured the city in 1258 A.D. A variety of automatic devices have been used over the centuries to accomplish useful tasks or simply to just entertain. The latter includes the automata, popular in Europe in the 17th and 18th centuries, featuring dancing figures that would repeat the same task over and over again; these automata are examples of open-loop control. Milestones among feedback, or "closed-loop" automatic control devices, include the temperature regulator of a furnace attributed to Drebbel, circa 1620, and the centrifugal flyball governor used for regulating the speed of steam engines by James Watt in 1788.

In his 1868 paper "On Governors", J. C. Maxwell (who discovered the Maxwell electromagnetic field equations) was able to explain instabilities exhibited by the flyball governor using differential equations to describe the control system. This demonstrated the importance and usefulness of mathematical models and methods in understanding complex phenomena, and signaled the beginning of mathematical control and systems theory. Elements of control theory had appeared earlier but not as dramatically and convincingly as in Maxwell's analysis.

Control theory made significant strides in the next 100 years. New mathematical techniques made it possible to control, more accurately, significantly more complex dynamical systems than the original flyball governor. These techniques include developments in optimal control in the 1950s and 1960s, followed by progress in stochastic, robust, adaptive and optimal control methods in the 1970s and 1980s. Applications of control methodology have helped make possible space travel and communication satellites, safer and more efficient aircraft, cleaner auto engines, cleaner and more efficient chemical processes, to mention but a few.

Before it emerged as a unique discipline, control engineering was practiced as a part of mechanical engineering and control theory was studied as a part of electrical engineering, since electrical circuits can often be easily described using control theory techniques. In the very first control relationships, a current output was represented with a voltage control input. However, not having proper technology to implement electrical control systems, designers left with the option of less efficient and slow responding mechanical systems. A very effective mechanical controller that is still widely used in some hydro plants is the governor. Later on, previous to modern power electronics, process control systems for industrial applications were devised by mechanical engineers using pneumatic and hydraulic control devices, many of which are still in use today.

Control theory

There are two major divisions in control theory, namely, classical and modern, which have direct implications over the control engineering applications. The scope of classical control theory is limited to single-input and single-output (SISO) system design. The system analysis is carried out in time domain using differential equations, in complex-s domain with Laplace transform or in frequency domain by transforming from the complex-s domain. All systems are assumed to be second order and single variable, and higher-order system responses and multivariable effects are ignored. A controller designed using classical theory usually requires on-site tuning due to design approximations. Yet, due to easier physical implementation of classical controller designs as compared to systems designed using modern control theory, these controllers are preferred in most industrial applications. The most common controllers designed using classical control theory are PID controllers.

In contrast, modern control theory is carried out strictly in the complex-s or the frequency domain, and can deal with multi-input and multi-output (MIMO) systems. This overcomes the limitations of classical control theory in more sophisticated design problems, such as fighter aircraft control. In modern design, a system is represented as a set of first order differential equations defined using state variables. Nonlinear, multivariable, adaptive and robust control theories come under this division. Being fairly new, modern control theory has many areas yet to be explored. Scholars like Rudolf E. Kalman and Aleksandr Lyapunov are well-known among the people who have shaped modern control theory.

Control systems

Control engineering is the engineering discipline that focuses on the modeling of a diverse range of dynamic systems (e.g. mechanical systems) and the design of controllers that will cause these systems to behave in the desired manner. Although such controllers need not be electrical many are and hence control engineering is often viewed as a subfield of electrical engineering. However, the falling price of microprocessors is making the actual implementation of a control system essentially trivial. As a result, focus is shifting back to the mechanical engineering discipline, as intimate knowledge of the physical system being controlled is often desired.

Electrical circuits, digital signal processors and microcontrollers can all be used to implement Control systems. Control engineering has a wide range of applications from the flight and propulsion systems of commercial airliners to the cruise control present in many modern automobiles.

In most of the cases, control engineers utilize feedback when designing control systems. This is often accomplished using a PID controller system. For example, in an automobile with cruise control the vehicle's speed is continuously monitored and fed back to the system which adjusts the motor's torque accordingly. Where there is regular feedback, control theory can be used to determine how the system responds to such feedback. In practically all such systems stability is important and control theory can help ensure stability is achieved.

Although feedback is an important aspect of control engineering, control engineers may also work on the control of systems without feedback. This is known as open loop control. A classic example of open loop control is a washing machine that runs through a pre-determined cycle without the use of sensors.

Control engineering education

At many universities, control engineering courses are taught in Electrical and Electronic Engineering, Mechatronics Engineering, Mechanical engineering, and Aerospace engineering; in others it is connected to computer science, as most control techniques today are implemented through computers, often as Embedded systems (as in the automotive field). The field of control within chemical engineering is often known as process control. It deals primarily with the control of variables in a chemical process in a plant. It is taught as part of the undergraduate curriculum of any chemical engineering program, and employs many of the same principles in control engineering. Other engineering disciplines also overlap with control engineering, as it can be applied to any system for which a suitable model can be derived.

Control engineering has diversified applications that include science, finance management, and even human behavior. Students of control engineering may start with a linear control system course dealing with the time and complex-s domain, which requires a thorough background in elementary mathematics and Laplace transform (called

classical control theory). In linear control, the student does frequency and time domain analysis. Digital control and nonlinear control courses require z transformation and algebra respectively, and could be said to complete a basic control education. From here onwards there are several sub branches.

Recent advancement

Originally, control engineering was all about continuous systems. Development of computer control tools posed a requirement of discrete control system engineering because the communications between the computer-based digital controller and the physical system are governed by a computer clock. The equivalent to Laplace transform in the discrete domain is the z-transform. Today many of the control systems are computer controlled and they consist of both digital and analog components.

Therefore, at the design stage either digital components are mapped into the continuous domain and the design is carried out in the continuous domain, or analog components are mapped in to discrete domain and design is carried out there. The first of these two methods is more commonly encountered in practice because many industrial systems have many continuous systems components, including mechanical, fluid, biological and analog electrical components, with a few digital controllers.

Similarly, the design technique has progressed from paper-and-ruler based manual design to computer-aided design, and now to computer-automated design (CAutoD), which has been made possible by evolutionary computation. CAutoD can be applied not just to tuning a predefined control scheme, but also to controller structure optimisation, system identification and invention of novel control systems, based purely upon a performance requirement, independent of any specific control scheme ,.

Chapter 1

Control System

A **control system** is a device or set of devices to manage, command, direct or regulate the behavior of other devices or systems.

There are two common classes of control systems, with many variations and combinations: logic or sequential controls, and feedback or linear controls. There is also fuzzy logic, which attempts to combine some of the design simplicity of logic with the utility of linear control. Some devices or systems are inherently not controllable.

Overview

The term "control system" may be applied to the essentially manual controls that allow an operator, for example, to close and open a hydraulic press, perhaps including logic so that it cannot be moved unless safety guards are in place.

An automatic sequential control system may trigger a series of mechanical actuators in the correct sequence to perform a task. For example various electric and pneumatic transducers may fold and glue a cardboard box, fill it with product and then seal it in an automatic packaging machine.

In the case of linear feedback systems, a **control loop**, including sensors, control algorithms and actuators, is arranged in such a fashion as to try to regulate a variable at a setpoint or reference value. An example of this may increase the fuel supply to a furnace when a measured temperature drops. PID controllers are common and effective in cases such as this. Control systems that include some sensing of the results they are trying to achieve are making use of feedback and so can, to some extent, adapt to varying circumstances. Open-loop control systems do not make use of feedback, and run only in pre-arranged ways.

Logic control

Logic control systems for industrial and commercial machinery were historically implemented at mains voltage using interconnected relays, designed using ladder logic. Today, most such systems are constructed with programmable logic controllers (PLCs) or microcontrollers. The notation of ladder logic is still in use as a programming idiom for PLCs.

Logic controllers may respond to switches, light sensors, pressure switches, etc., and can cause the machinery to start and stop various operations. Logic systems are used to sequence mechanical operations in many applications. Examples include elevators, washing machines and other systems with interrelated stop-go operations.

Logic systems are quite easy to design, and can handle very complex operations. Some aspects of logic system design make use of Boolean logic.

On–off control

For example, a thermostat is a simple negative-feedback control: when the temperature (the "process variable" or PV) goes below a set point (SP), the heater is switched on. Another example could be a pressure switch on an air compressor: when the pressure (PV) drops below the threshold (SP), the pump is powered. Refrigerators and vacuum pumps contain similar mechanisms operating in reverse, but still providing negative feedback to correct errors.

Simple **on–off** feedback control systems like these are cheap and effective. In some cases, like the simple compressor example, they may represent a good design choice.

In most applications of on–off feedback control, some consideration needs to be given to other costs, such as wear and tear of control valves and maybe other start-up costs when power is reapplied each time the PV drops. Therefore, practical on–off control systems are designed to include hysteresis, usually in the form of a deadband, a region around the setpoint value in which no control action occurs. The width of deadband may be adjustable or programmable.

Linear control

Linear control systems use linear negative feedback to produce a control signal mathematically based on other variables, with a view to maintaining the controlled process within an acceptable operating range.

The output from a linear control system into the controlled process may be in the form of a directly variable signal, such as a valve that may be 0 or 100% open or anywhere in between. Sometimes this is not feasible and so, after calculating the current required corrective signal, a linear control system may repeatedly switch an actuator, such as a

pump, motor or heater, fully on and then fully off again, regulating the duty cycle using pulse-width modulation.

Proportional control

When controlling the temperature of an industrial furnace, it is usually better to control the opening of the fuel valve **in proportion to** the current needs of the furnace. This helps avoid thermal shocks and applies heat more effectively.

Proportional negative-feedback systems are based on the difference between the required set point (SP) and process value (PV). This difference is called the error. Power is applied in direct proportion to the current measured error, in the correct sense so as to tend to reduce the error (and so avoid positive feedback). The amount of corrective action that is applied for a given error is set by the gain or sensitivity of the control system.

At low gains, only a small corrective action is applied when errors are detected: the system may be safe and stable, but may be sluggish in response to changing conditions; errors will remain uncorrected for relatively long periods of time: it is over-damped. If the proportional gain is increased, such systems become more responsive and errors are dealt with more quickly. There is an optimal value for the gain setting when the overall system is said to be critically damped. Increases in loop gain beyond this point will lead to oscillations in the PV; such a system is under-damped.

Under-damped furnace example

In the furnace example, suppose the temperature is increasing towards a set point at which, say, 50% of the available power will be required for steady-state. At low temperatures, 100% of available power is applied. When the PV is within, say 10° of the SP the heat input begins to be reduced by the proportional controller. (Note that this implies a 20° "proportional band" (PB) from full to no power input, evenly spread around the setpoint value). At the setpoint the controller will be applying 50% power as required, but stray stored heat within the heater sub-system and in the walls of the furnace will keep the measured temperature rising beyond what is required. At 10° above SP, we reach the top of the proportional band (PB) and no power is applied, but the temperature may continue to rise even further before beginning to fall back. Eventually as the PV falls back into the PB, heat is applied again, but now the heater and the furnace walls are too cool and the temperature falls too low before its fall is arrested, so that the oscillations continue.

Over-damped furnace example

The temperature oscillations that an under-damped furnace control system produces are unacceptable for many reasons, including the waste of fuel and time (each oscillation cycle may take many minutes), as well as the likelihood of seriously overheating both the furnace and its contents.

Suppose that the gain of the control system is reduced drastically and it is restarted. As the temperature approaches, say 30° below SP (60° proportional band or PB now), the heat input begins to be reduced, the rate of heating of the furnace has time to slow and, as the heat is still further reduced, it eventually is brought up to set point, just as 50% power input is reached and the furnace is operating as required. There was some wasted time while the furnace crept to its final temperature using only 52% then 51% of available power, but at least no harm was done. By carefully increasing the gain (i.e. reducing the width of the PB) this over-damped and sluggish behavior can be improved until the system is critically damped for this SP temperature. Doing this is known as 'tuning' the control system. A well-tuned proportional furnace temperature control system will usually be more effective than on-off control, but will still respond slower than the furnace could under skillful manual control.

PID control

Apart from sluggish performance to avoid oscillations, another problem with proportional-only control is that power application is always in direct proportion to the error. In the example above we assumed that the set temperature could be maintained with 50% power. What happens if the furnace is required in a different application where a higher set temperature will require 80% power to maintain it? If the gain was finally set to a 50° PB, then 80% power will not be applied unless the furnace is 15° below setpoint, so for this other application the operators will have to remember always to set the setpoint temperature 15° higher than actually needed. This 15° figure is not completely constant either: it will depend on the surrounding ambient temperature, as well as other factors that affect heat loss from or absorption within the furnace.

To resolve these two problems, many feedback control schemes include mathematical extensions to improve performance. The most common extensions lead to proportional-integral-derivative control, or PID control (pronounced pee-eye-dee).

Derivative action

The derivative part is concerned with the rate-of-change of the error with time: If the measured variable approaches the setpoint rapidly, then the actuator is backed off early to allow it to coast to the required level; conversely if the measured value begins to move rapidly away from the setpoint, extra effort is applied—in proportion to that rapidity—to try to maintain it.

Derivative action makes a control system behave much more intelligently. On systems like the temperature of a furnace, or perhaps the motion-control of a heavy item like a gun or camera on a moving vehicle, the derivative action of a well-tuned PID controller can allow it to reach and maintain a setpoint better than most skilled human operators could.

If derivative action is over-applied, it can lead to oscillations too. An example would be a PV that increased rapidly towards SP, then halted early and seemed to "shy away" from the setpoint before rising towards it again.

Integral action

The integral term magnifies the effect of long-term steady-state errors, applying ever-increasing effort until they reduce to zero. In the example of the furnace above working at various temperatures, if the heat being applied does not bring the furnace up to setpoint, for whatever reason, integral action increasingly *moves* the proportional band relative to the setpoint until the PV error is reduced to zero and the setpoint is achieved.

Other techniques

Another common technique is to filter the PV or error signal. Such a filter can reduce the response of the system to undesirable frequencies, to help eliminate instability or oscillations. Some feedback systems will oscillate at just one frequency. By filtering out that frequency, one can use very "stiff" feedback and the system can be very responsive without shaking itself apart.

The most complex linear control systems developed to date are in oil refineries (model predictive control). The chemical reaction paths and control systems are normally designed together using specialized computer-aided-design software.

Feedback systems can be combined in many ways. One example is **cascade control** in which one control loop applies control algorithms to a measured variable against a setpoint, but then actually outputs a setpoint to another controller, rather than affecting power input directly.

Usually if a system has several measurements to be controlled, feedback systems will be present for each of them.

Fuzzy logic

Fuzzy logic is an attempt to get the easy design of logic controllers and yet control continuously-varying systems. Basically, a measurement in a fuzzy logic system can be partly true, that is if yes is 1 and no is 0, a fuzzy measurement can be between 0 and 1.

The rules of the system are written in natural language and translated into fuzzy logic. For example, the design for a furnace would start with: "If the temperature is too high, reduce the fuel to the furnace. If the temperature is too low, increase the fuel to the furnace."

Measurements from the real world (such as the temperature of a furnace) are converted to values between 0 and 1 by seeing where they fall on a triangle. Usually the tip of the triangle is the maximum possible value which translates to "1."

Fuzzy logic, then, modifies Boolean logic to be arithmetical. Usually the "not" operation is "output = 1 - input," the "and" operation is "output = input.1 multiplied by input.2," and "or" is "output = 1 - ((1 - input.1) multiplied by (1 - input.2))". This reduces to Boolean arithmetic if values are restricted to 0 and 1, instead of allowed to range in the unit interval [0,1].

The last step is to "defuzzify" an output. Basically, the fuzzy calculations make a value between zero and one. That number is used to select a value on a line whose slope and height converts the fuzzy value to a real-world output number. The number then controls real machinery.

If the triangles are defined correctly and rules are right the result can be a good control system.

When a robust fuzzy design is reduced into a single, quick calculation, it begins to resemble a conventional feedback loop solution and it might appear that the fuzzy design was unnecessary. However, the fuzzy logic paradigm may provide scalability for large control systems where conventional methods become unwieldy or costly to derive.

Fuzzy electronics is an electronic technology that uses fuzzy logic instead of the two-value logic more commonly used in digital electronics.

Physical implementations

Since modern small microprocessors are so cheap (often less than $1 US), it's very common to implement control systems, including feedback loops, with computers, often in an embedded system. The feedback controls are simulated by having the computer make periodic measurements and then calculating from this stream of measurements.

Computers emulate logic devices by making measurements of switch inputs, calculating a logic function from these measurements and then sending the results out to electronically-controlled switches.

Logic systems and feedback controllers are usually implemented with programmable logic controllers which are devices available from electrical supply houses. They include a little computer and a simplified system for programming. Most often they are programmed with personal computers.

Logic controllers have also been constructed from relays, hydraulic and pneumatic devices, and electronics using both transistors and vacuum tubes (feedback controllers can also be constructed in this manner).

Chapter 2

Applications of Multiple Coordinate Systems

The **application of multiple coordinate systems** is an effective tool in control systems. The usage of multiple coordinate systems can improve the efficiency of calculations as well as enhance clarity in operations.

Mathematically, coordinate systems can be viewed as ordered bases. Consider the scenario where V is a vector space with ordered bases A and B. Let A and B consist of vectors a_1 through a_n and b_1 through b_n, respectively. Then any vector v contained in V can be expressed as a unique linear combination of vectors in A. In equation form,

$$v = \sum_{i=1}^{n} x_i a_i$$

where x is a coefficient. This linear combination uses a unique ordered set of coefficients corresponding to vectors in A. Therefore, for a fixed v and A, there is one and only one possible set of x_i's for $v = \sum_{i=1}^{n} x_i a_i$. Similarly, v can be expressed as a unique linear combination of vectors in B as well. Thus, there is another unique ordered set of coefficients corresponding to vectors in B to generate v.

If the relationship between the elements in the two ordered bases A and B is known, then the relationship between the two unique ordered set of coefficients to generate v can be found. The equation form of this concept is $[u]_A = P[u]_B$. In this equation, $[u]_A$ represents a column vector composed of the unique ordered set of coefficients to generate v from the elements in A. In other words, $[u]_A$ is composed of the x_i's mentioned above. Similarly, $[u]_B$ represents a column vector composed of the unique ordered set of coefficients to generate v from the vectors in B.

The matrix P is found from the relationship between the elements in the two ordered bases. Each element of the ordered bases, a_1 through a_n and b_1 through b_n, is an element

of the vector space. Therefore, it is guaranteed that any element in B can be expressed as a unique linear combination of elements in A and the following sets of equations apply:

$$b_1 = x_{11}a_1 + x_{12}a_2 + \cdots + x_{1n}a_n$$
$$b_2 = x_{21}a_1 + x_{22}a_2 + \cdots + x_{2n}a_n$$
$$\cdots$$
$$b_n = x_{n1}a_1 + x_{n2}a_2 + \cdots + x_{nn}a_n$$

Note that the x_{ij} terms are coefficients. The process to find P starts with this set of n equations. In matrix form, the set of equations can been seen as $b = Xa$. The i^{th} row, j^{th} column term of the matrix X is equal to x_{ij}. P is simply the transpose of the matrix X. In equation form, $p_{ij} = x_{ji}$. Since X is an n-by-n matrix, then P is also an n-by-n matrix.

P is also invertible. An informal proof will be provided here. First, the elements a_1 through a_n are linearly independent since A is a basis. This implies that the rows of matrix X are unique. Second, the elements b_1 through b_n are linearly independent since B is a basis. Thus, the rows of matrix X are linearly independent. Third, the rows of matrix X are the columns of matrix P, so the columns of matrix P are linearly independent. Fourth, a matrix is invertible if the columns are linearly independent, hence P is invertible.

Since P is invertible, then $[u]_B = P^{-1}[u]_A$. There are multiple ways to invert matrix P, but the following method is easy to memorize and works for all n-by-n matrices. To begin, create an n-by-2n matrix with the form $(P|I)$, where I is the n-by-n identity matrix. A series of row reductions should be used to manipulate the first n columns, P, to I. These row reductions will also affect the n+1 to 2n columns. The end result will be $(I|P^{-1})$.

Lastly, there is a quick way to verify the accuracy of P. If v is chosen to be b_1, then the representation of v in B is simply $1 \times b_1$. According to the linear equations given above, this particular v is represented in A as $b_1 = x_{11}a_1 + x_{12}a_2 + \cdots + x_{1n}a_n$. Therefore, $[u]_B$ is a column vector starting with one and zero for the remainder of the vector and $[u]_A$ is a column vector where $[u]_A^T = (x_{11}\ x_{12}\ \ldots\ x_{1n})$. The matrix P should confirm the relationship between these two sets of coefficients.

The relationship of multiple coordinate systems is an important part of remote control systems. In a remote control system, there is an operator, an object being manipulated at a distance, and a target destination for the manipulated object. In a closed-loop control system, the operator sends commands to the object, the object executes the commands, and the object sends feedback to the operator. The feedback is used to determine the new location and orientation of the object. Closed-loop operations in remote control can be established with three fundamental frames: a frame of resolution (FOR), a command frame, and a display frame. The following sections describe these frames.

The purpose of the FOR is to reduce the concept of the object to an origin with three axes, i.e. a frame. The origin defines the location of the object and the axes define the orientation of the object. It should be clear that the FOR should translate and rotate with the object. In other words, the FOR should be fixed with respect to the manipulated object. This is also known as a body-fixed frame.

The command frame is used to interpret the user inputs. The axes of the command frame provide the direction of translation applied at the origin of the FOR. The origin of the command frame is used as the center for rotation. The axes of the command frame are also used to determine the line of rotation for roll, pitch, and yaw. Roll, pitch, and yaw are rotations about the x, y and z axes, respectively. The command frame can be fixed with respect to the object or fixed with respect to the vector space. These concepts are also known as body-fixed and space-fixed frames or internal and external frames.

The display frame is important for user feedback. The purpose of the display frame is to locate where the object is. More specifically, the translation and orientation from the display frame to the FOR is used to identify where the object is. The operator can use this information to determine if the operation is following the intended path and make corrections as needed. The display frame is a space-fixed frame and typically the location of the frame is chosen to be at the destination or a frame commonly used for the overall system.

A simple example using multiple coordinate systems is a remote controlled car in a 2D space. Consider a system with an operator, a car, and a target destination. The origin of the FOR is centered between the wheels of the car, the command frame will not be used, and the display frame corresponds to the target destination. The given information consists of the location and orientation of the operator and the display frame and the location of the car with respect to the operator. Let the display frame be located at the origin (0,0) with a x-y coordinate system defined by a_1=(1,0) and a_2=(0,1). Also, let the operator be located at (100, 50) with the same coordinate system. Thus, $b_1 = 1a_1 + 0a_2$ and $b_2 = 0a_1 + 1a_2$. Then $X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $P = X^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The purpose of P in this example is to state where the car is with respect to the operator's origin, but using the display frame's coordinate system axes (a_1 and a_2). In this case, P is trivial because the display frame's coordinate system and the operator's coordinate system are the same. The result is in the form $[u]_A = P[u]_B + [\delta]$, where $[u]_A$ is the location of the car with respect to the display frame, $[u]_B$ is the location of the car with respect to the operator, and $[\delta]$ accounts for the distance from the display frame to the operator with respect to the display frame. The final result for this example is $[u]_A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}[u]_B + \begin{bmatrix} 100 \\ 50 \end{bmatrix}$. Thus, if the car is at a location (4, 10) with respect to the operator, the location of the car with respect to the display frame is given by the following equations:

$$x_A = 1 \times x_B + 0 \times y_B + 100 \quad = 1 \times 4 + 100 \quad = 104$$
$$y_A = 0 \times x_B + 1 \times y_B + 50 \quad = 1 \times 10 + 50 \quad = 60$$

Thus, the location of the car with respect to the display frame is (104, 60).

The problem is more interesting if the operator's frame is not aligned with the display frame. Consider the same example above, except the operator has turned 45 degrees clockwise to get a better view of the car. The operator's x-y coordinate system was originally defined as b_1 = (1cos(0), 1sin(0)) = (1, 0) and b_2 = (1cos(90), 1sin(90)) = (0,1). The new coordinate system is defined as follows:

$$c_1 = (1cos(-45), 1sin(-45)) \quad = (1/\sqrt{2}, -1/\sqrt{2}) \quad = a_1/\sqrt{2} - a_2/\sqrt{2}$$
$$c_2 = (1cos(45), 1sin(45)) \quad = (1/\sqrt{2}, 1/\sqrt{2}) \quad = a_1/\sqrt{2} + a_2/\sqrt{2}$$

Then $X = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$ and $P = X^T = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$.

The final result for this example is $[u]_A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} [u]_C + \begin{bmatrix} 100 \\ 50 \end{bmatrix}$.

To partially verify the result, consider the case where the car is at a location c_1 away from the operator. For this case, $[u]_C = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $[u]_A$ should be equal to $c_1 + [\delta]$. When the value for $[u]_C$ is substituted into the equation,

$$[u]_A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 100 \\ 50 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} + 100 \\ -1/\sqrt{2} + 50 \end{bmatrix} = c_1 + [\delta].$$

Chapter 3

Model-Based Design

Model-Based Design (MBD) is a mathematical and visual method of addressing problems associated with designing complex control, signal processing and communication systems. It is used in many motion control, industrial equipment, aerospace, and automotive applications. Model-based design is a methodology applied in designing embedded software.

MBD provides an efficient approach for establishing a common framework for communication throughout the design process while supporting the development cycle ("V" diagram). In Model-based design of control systems, development is manifested in these four steps: 1) modeling a plant, 2) analyzing and synthesizing a controller for the plant, 3) simulating the plant and controller, and 4) integrating all these phases by deploying the controller. The model-based design paradigm is significantly different from traditional design methodology. Rather than using complex structures and extensive software code, designers can use MBD to define models with advanced functional characteristics using continuous-time and discrete-time building blocks. These built models used with simulation tools can lead to rapid prototyping, software testing, and verification. Not only is the testing and verification process enhanced, but also, in some cases, hardware-in-the-loop simulation can be used with the new design paradigm to perform testing of dynamic effects on the system more quickly and much more efficiently than with traditional design methodology.

The main steps in MBD approach are:

1. Plant modeling. Plant modeling can be data-driven or first principles based. Data-driven plant modeling uses techniques such as System identification. With system identification, the plant model is identified by acquiring and processing raw data from a real-world system and choosing a mathematical algorithm with which to identify a mathematical model. Various kinds of analysis and simulations can be performed using the identified model before it is used to design a model-based

controller. First principles based modeling is based on creating a block diagram model that implements known differential-algebraic equations governing plant dynamics. A type of first principles based modeling is physical modeling, where a model is created by connectings blocks that represent physical elements that the actual plant consists of.

2. Controller analysis and synthesis. The mathematical model conceived in step 1 is used to identify dynamic characteristics of the plant model. A controller can be then be synthesized based on these characteristics.
3. Offline simulation and real-time simulation. The time response of the dynamic system to complex, time-varying inputs is investigated. This is done by simulating a simple LTI or a non-linear model of the plant with the controller. Simulation allows specification, requirements, and modeling errors to be found immediately, rather than later in the design effort. Real-time simulation can be done by automatically generating code for the controller developed in step 3. This code can be deployed to a special real-time protoyping computer that can run the code and control the operation of the plant. If plant prototype is not available, or testing on the prototype is dangerous or expensive, code can be automatically generated from the plant model. This code can be deployed to the special real-time computer that can be connected to the target processor with running controller code. This way, controller can be tested in real-time against a real-time plant model.
4. Deployment. Ideally this is done via automatic code generation from the controller developed in step 3. It is unlikely that the controller will work on the actual system as well as it did in simulation, so an iterative debugging process is done by analyzing results on the actual target and updating the controller model. Model based design tools allow all these iterative steps to be performed in a unified visual environment.

Some of the notable advantages MBD offers in comparison to the traditional approach are:

- MBD provides a common design environment, which facilitates general communication, data analysis, and system verification between development groups.
- Engineers can locate and correct errors early in system design, when the time and financial impact of system modification are minimized.
- Design reuse, for upgrades and for derivative systems with expanded capabilities, is facilitated

History

The dawn of the electrical age brought many innovative and advanced control systems. As early as the 1920s two aspects of engineering, control theory and control systems, converged to make large-scale integrated systems possible. In those early days controls systems were commonly used in the industrial environment. Large process facilities started using process controllers for regulating continuous variables such as temperature,

pressure, and flow rate. Electrical relays built into ladder-like networks were one of the first discrete control devices to automate an entire manufacturing process.

Control systems gained momentum, primarily in the automotive and aerospace sectors. In the 1950s and 1960s the push to Space generated interest in embedded control systems. Engineers constructed control systems such as engine control units and flight simulators, that could be part of the end product. By the end of the twentieth century, embedded control systems were ubiquitous, as even White goods such as washing machines and air-conditions contained complex and advanced control algorithms, making them a much more "intelligent".

In the year 1969, the first computer-based controllers were introduced, These early programmable logic controllers (PLC), mimicked the operations of already available discrete control technologies that used the out-dated relay ladders. The advent of PC technology brought a drastic shift in the process and discrete control market. An off-the-shelf desktop loaded with adequate hardware and software can run an entire process unit, and execute complex and established PID algorithms or work as a Distributed Control System (DCS).

Challenges

Modeling and simulation tools have long been in use, but traditional text-based tools are inadequate for the complex nature of modern control systems. Because of the limitations of graphical tools, design engineers previously relied heavily on text-based programming and mathematical models. However, developing these models was difficult, time-consuming, and highly prone to error. In addition, debugging text-based programs was a tedious process, requiring much trial and error before a final fault-free model could be created, especially since mathematical models undergo unseen changes during the translation through the various design stages.

These challenges are overcome by the use of graphical modeling tools, used today in all aspects of design. These tools provide a very generic and unified graphical modeling environment, they reduce the complexity of model designs by breaking them into hierarchies of individual design blocks. Designers can thus achieve multiple levels of model fidelity by simply substituting one block element with another. Graphical models are also the best way to document engineers' ideas. It helps engineers to conceptualize the entire system and simplifies the process of transporting the model from one stage to another in the design process. Boeing's simulator EASY5 was among the first modeling tools to be provided with a graphical user interface. This was followed by many other tools.

When developing embedded control systems, designers are squeezed by two trends —- shrinking development cycles and growing design intricacy. The divide-and-conquer strategy for developing these complex systems means coordinating the resources of people with expertise in a wide range of disciplines. The traditional, text-based approach

of embedded system design is not efficient enough to handle such advanced, complex systems.

Chapter 4

Furuta Pendulum

Rotational Inverted Pendulum: Classic pedagogical example of application of control theory

The Furuta pendulum, or rotational inverted pendulum, consists of a driven arm which rotates in the horizontal plane and a pendulum attached to that arm which is free to rotate in the vertical plane (Fig 1).

Overview

The **Furuta pendulum** was first developed at the Tokyo Institute of Technology by Katsuhisa Furuta and his colleagues. The pendulum is underactutated and extremely non-linear due to the gravitational forces and the coupling arising from the Coriolis and centripetal forces. Since then, dozens, possibly hundreds of papers and theses have used the system to demonstrate linear and non-linear control laws . The system has also been the subject of two texts].

Equations of motion

Despite the great deal of attention the system has received, very few publications successfully derive (or use) the full dynamics. Many authors [1] have only considered the rotational inertia of the pendulum for a single principal axis (or neglected it altogether). In other words, the inertia tensor only has a single non-zero element (or none), and the remaining two diagonal terms are zero. It is possible to find a pendulum system where the moment of inertia in one the three principal axes is approximately zero, but not two.

The equations of motion presented here are an extract from a paper on the Furuta pendulum dynamics derived at the University of Adelaide.

Definitions

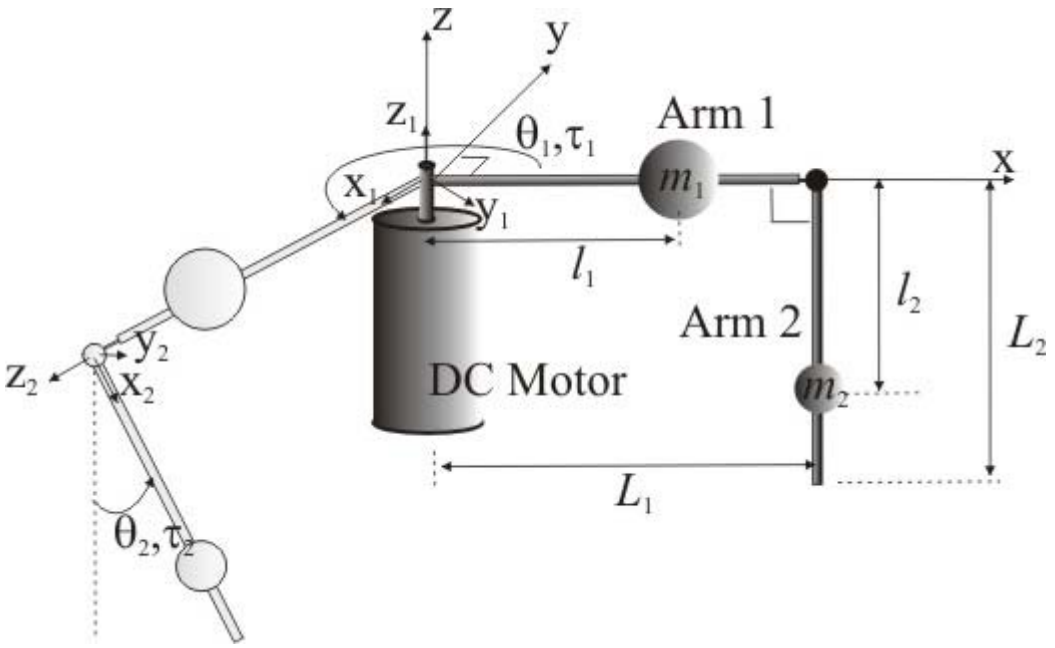


Fig. 1: Schematic of the single rotary inverted pendulum system.

Consider the rotational inverted pendulum mounted to a DC motor as shown in Fig. 1. The DC motor is used to apply a torque τ_1 to Arm 1. The link between Arm 1 and Arm 2 is not actuated but free to rotate. The two arms have lengths L_1 and L_2. The arms have masses m_1 and m_2 which are located at l_1 and l_2 respectively, which are the lengths from the point of rotation of the arm to its center of mass. The arms have inertia tensors $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ (about the centre of mass of the arm). Each rotational joint is viscously damped with damping coefficients b_1 and b_2, where b_1 is the damping provided by the motor bearings and b_2 is the damping arising from the pin coupling between Arm 1 and Arm 2.

A right hand coordinate system has been used to define the inputs, states and the Cartesian coordinate systems 1 and 2. The coordinate axes of Arm 1 and Arm 2 are the principal axes such that the inertia tensors are diagonal.

The angular rotation of Arm 1, θ_1, is measured in the horizontal plane where a counter-clockwise direction (when viewed from above) is positive. The angular rotation of Arm 2, θ_2 , is measured in the vertical plane where a counter-clockwise direction (when viewed from the front) is positive. When the Arm is hanging down in the stable equilibrium position $\theta_2 = 0$.

The torque the servo-motor applies to Arm 1, τ_1, is positive in a counter-clockwise direction (when viewed from above). A disturbance torque, τ_2, is experienced by Arm 2, where a counter-clockwise direction (when viewed from the front) is positive.

Assumptions

Before deriving the dynamics of the system a number of assumptions must be made. These are:

- The motor shaft and Arm 1 are assumed to be rigidly coupled and infinitely stiff.
- Arm 2 is assumed to be infinitely stiff.
- The coordinate axes of Arm1 and Arm 2 are the principal axes such that the inertia tensors are diagonal.
- The motor rotor inertia is assumed to be negligible. However, this term may be easily added to the moment of inertia of Arm 1.
- Only viscous damping is considered. All other forms of damping (such as Coulomb) have been neglected, however it is a simple exercise to add this to the final governing DE.

Non-linear Equations of Motion

The non-linear equations of motion are given by

$$\ddot{\theta}_1\left(J_{1zz}+m_1l_1^2+m_2L_1^2+\left(J_{2yy}+m_2l_2^2\right)\sin^2(\theta_2)+J_{2xx}\cos^2(\theta_2)\right)+\ddot{\theta}_2 m_2L_1l_2\cos(\theta_2)-m_2L_1l_2\sin(\theta_2)\dot{\theta}_2^2+\dot{\theta}_1\dot{\theta}_2\sin(2\theta_2)\left(m_2l_2^2+J_{2yy}-J_{2xx}\right)+b_1\dot{\theta}_1=\tau_1$$

and

$$\ddot{\theta}_1 m_2L_1l_2\cos(\theta_2)+\ddot{\theta}_2(m_2l_2^2+J_{2zz})+1/2\dot{\theta}_1^2\sin(2\theta_2)(-m_2l_2^2-J_{2yy}+J_{2xx})+b_2\dot{\theta}_2+gm_2l_2\sin(\theta_2)=\tau_2$$

Simplifications

Most Furuta pendulums tend to have long slender arms, such that the moment of inertia along the axis of the arms is negligible. In addition, most arms have rotational symmetry such that the moments of inertia in two of the principal axes are equal. Thus, the inertia tensors may be approximated as follows:

$$\boldsymbol{J}_1 = diag[J_{1xx}, J_{1yy}, J_{1zz}] = diag[0, J_1, J_1]$$

$$\boldsymbol{J}_2 = diag[J_{2xx}, J_{2yy}, J_{2zz}] = diag[0, J_2, J_2]$$

Further simplifications are obtained by making the following substitutions. The total moment of inertia of Arm 1 about the pivot point (using the parallel axis theorem) is $\hat{J}_1 = J_1 + m_1 l_1^2$. The total moment of inertia of Arm 2 about its pivot point is $\hat{J}_2 = J_2 + m_2 l_2^2$. Finally, define the total moment of inertia the motor rotor experiences when the pendulum (Arm 2) is in its equilibrium position (hanging vertically down), $\hat{J}_0 = \hat{J}_1 + m_2 L_1^2 = J_1 + m_1 l_1^2 + m_2 L_1^2$.

Substituting the previous definitions into the governing DEs gives the more compact form

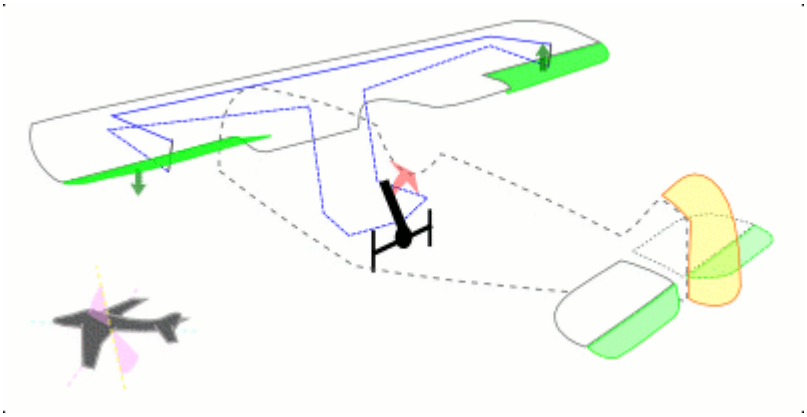
$$\ddot{\theta}_1 \left(\hat{J}_0 + \hat{J}_2 \sin^2(\theta_2) \right) + \ddot{\theta}_2 m_2 L_1 l_2 \cos(\theta_2) - m_2 L_1 l_2 \sin(\theta_2) \dot{\theta}_2^2 + \dot{\theta}_1 \dot{\theta}_2 \sin(2\theta_2) \hat{J}_2 + b_1 \dot{\theta}_1 = \tau_1$$

and

$$\ddot{\theta}_1 m_2 L_1 l_2 \cos(\theta_2) + \ddot{\theta}_2 \hat{J}_2 - 1/2 \dot{\theta}_1^2 \sin(2\theta_2) \hat{J}_2 + b_2 \dot{\theta}_2 + g m_2 l_2 \sin(\theta_2) = \tau_2$$

Chapter 5

Aircraft Flight Control System



A typical aircraft's primary flight controls in motion

A conventional fixed-wing **aircraft flight control system** consists of flight control surfaces, the respective cockpit controls, connecting linkages, and the necessary operating mechanisms to control an aircraft's direction in flight. Aircraft engine controls are also considered as flight controls as they change speed.

The fundamentals of aircraft controls are explained in flight dynamics.

Cockpit controls

Primary controls

Generally the primary cockpit controls are arranged as follows:

- A control column or a control yoke attached to a column—for roll and pitch, which moves the ailerons when turned or deflected left and right, and moves the elevators when moved backwards or forwards

- Rudder pedals, or the earlier, pre-1919 "rudder bar", to control yaw, which move the rudder; left foot forward will move the rudder left for instance.
- Throttle controls to control engine speed or thrust for powered aircraft.

Even when an aircraft uses different kinds of surfaces, such as a V-tail/ruddervator, flaperons, or elevons, to avoid pilot confusion the aircraft will still normally be designed so that the yoke or stick controls pitch and roll in the conventional way, as will the rudder pedals for yaw. The basic pattern for modern flight controls was pioneered by French aviation figure Robert Esnault-Pelterie, with fellow French aviator Louis Blériot popularizing Esnault-Pelterie's control format initially on Louis' Blériot VIII monoplane, and standardizing the format on the July 1909 Channel-crossing Blériot XI.

Secondary controls

In addition to the primary flight controls for roll, pitch, and yaw, there are often secondary controls available to give the pilot finer control over flight or to ease the workload. The most commonly-available control is a wheel or other device to control elevator trim, so that the pilot does not have to maintain constant backward or forward pressure to hold a specific pitch attitude (other types of trim, for rudder and ailerons, are common on larger aircraft but may also appear on smaller ones). Many aircraft have wing flaps, controlled by a switch or a mechanical lever or in some cases are fully automatic by computer control, which alter the shape of the wing for improved control at the slower speeds used for takeoff and landing. Other secondary flight control systems may be available, including slats, spoilers, air brakes and variable-sweep wings.

Flight control systems

Mechanical

de Havilland Tiger Moth elevator and rudder cables

Mechanical or manually-operated flight control systems are the most basic method of controlling an aircraft. They were used in early aircraft and are currently used in small aircraft where the aerodynamic forces are not excessive. Very early aircraft, such as the Wright Flyer I, Blériot XI and Fokker Eindecker used a system of wing warping where no conventionally hinged control surfaces were used on the wing, and sometimes not even for pitch control as on the Wright Flyer I and original versions of the 1909 Etrich Taube, which only had a hinged/pivoting rudder in addition to the warping-operated pitch and roll controls. A manual flight control system uses a collection of mechanical parts such as pushrods, tension cables, pulleys, counterweights, and sometimes chains to transmit the forces applied to the cockpit controls directly to the control surfaces. Turnbuckles are often used to adjust control cable tension. The Cessna Skyhawk is a typical example of an aircraft that uses this type of system. Gust locks are often used on parked aircraft with mechanical systems to protect the control surfaces and linkages from damage from wind. Some aircraft have gust locks fitted as part of the control system.

Increases in the control surface area required by large aircraft or higher loads caused by high airspeeds in small aircraft lead to a large increase in the forces needed to move

them, consequently complicated mechanical gearing arrangements were developed to extract maximum mechanical advantage in order to reduce the forces required from the pilots. This arrangement can be found on bigger or higher performance propeller aircraft such as the Fokker 50.

Some mechanical flight control systems use servo tabs that provide aerodynamic assistance. Servo tabs are small surfaces hinged to the control surfaces. The flight control mechanisms move these tabs, aerodynamic forces in turn move, or assist the movement of the control surfaces reducing the amount of mechanical forces needed. This arrangement was used in early piston-engined transport aircraft and in early jet transports. The Boeing 737 incorporates a system, whereby in the unlikely event of total hydraulic system failure, it automatically and seamlessly reverts to being controlled via servo-tab.

Hydro-mechanical

The complexity and weight of mechanical flight control systems increase considerably with the size and performance of the aircraft. Hydraulically powered control surfaces help to overcome these limitations. With hydraulic flight control systems, the aircraft's size and performance are limited by economics rather than a pilot's muscular strength. At first, only-partially boosted systems were used in which the pilot could still feel some of the aerodynamic loads on the control surfaces (feedback).

A hydro-mechanical flight control system has two parts:

- The *mechanical circuit*, which links the cockpit controls with the hydraulic circuits. Like the mechanical flight control system, it consists of rods, cables, pulleys, and sometimes chains.

- The *hydraulic circuit*, which has hydraulic pumps, reservoirs, filters, pipes, valves and actuators. The actuators are powered by the hydraulic pressure generated by the pumps in the hydraulic circuit. The actuators convert hydraulic pressure into control surface movements. The electro-hydraulic servo valves control the movement of the actuators.

The pilot's movement of a control causes the mechanical circuit to open the matching servo valve in the hydraulic circuit. The hydraulic circuit powers the actuators which then move the control surfaces. As the actuator moves, the servo valve is closed by a mechanical feedback linkage - one that stops movement of the control surface at the desired position.

This arrangement was found in the older-designed jet transports and in some high-performance aircraft. Examples include the Antonov An-225 and the Lockheed SR-71.

Artificial feel devices

With purely mechanical flight control systems, the aerodynamic forces on the control surfaces are transmitted through the mechanisms and are felt directly by the pilot. This gives tactile feedback of airspeed and aids flight safety.

With hydromechanical flight control systems however, the load on the surfaces cannot be felt and there is a risk of overstressing the aircraft through excessive control surface movement. To overcome this problem artificial feel systems are used. For example, for the controls of the RAF's Avro Vulcan jet bomber and the RCAF's Avro Canada CF-105 Arrow supersonic interceptor, both 1950's-era designs, the required force feedback was achieved by a spring device. The fulcrum of this device was moved in proportion to the square of the air speed (for the elevators) to give increased resistance at higher speeds. For the controls of the American Vought F-8 Crusader and the LTV A-7 Corsair II warplanes, a "bob-weight" was used in the pitch axis of the control stick, giving force feedback that was proportional to the airplane's normal acceleration.

Stick shaker

A stick shaker is a device (available in some hydraulic aircraft) which is fitted into the control column which shakes the control column when the aircraft is about to stall. Also in some aircraft like the McDonnell Douglas DC-10 there is/was a back-up electrical power supply which the pilot can turn on to re-activate the stick shaker in case the hydraulic connection to the stick shaker is lost.

Fly-by-wire control systems

A fly-by-wire (FBW) system replaces manual flight control of an aircraft with an electronic interface. The movements of flight controls are converted to electronic signals transmitted by wires (hence the fly-by-wire term), and flight control computers determine how to move the actuators at each control surface to provide the expected response. Commands from the computers are also input without the pilot's knowledge to stabilize the aircraft and perform other tasks.

Research

Several technology research and development efforts exist to integrate the functions of flight control systems such as ailerons, elevators, elevons and flaps, into wings to perform the aerodynamic purpose with the advantages of less: mass, cost, drag, inertia (for faster, stronger control response), complexity (mechanically simpler, fewer moving parts or surfaces, less maintenance), and radar cross section for stealth. These may be used in many unmanned aerial vehicles (UAVs) and 6th generation fighter aircraft. The two main approaches are flexible wings, and fluidics.

Flexible wings

In flexible wings, much or all of a wing surface can change shape in flight to deflect air flow. The X-53 Active Aeroelastic Wing is a NASA effort. The Adaptive Compliant Wing is a commercial effort.

Fluidics

In fluidics, forces in vehicles occur via circulation control, in which larger more complex mechanical parts are replaced by smaller simpler fluidic systems (slots which emit air flows) where larger forces in fluids are diverted by smaller jets or flows of fluid intermittently, to change the direction of vehicles.[1] In this use, fluidics promises lower mass, costs (up to 50% less), and very low inertia and response times, and simplicity. This was demonstrated in the Demon UAV which flew for the first time, in the UK, in September 2010.

Chapter 6

Distributed Control System

A **distributed control system** (DCS) refers to a control system usually of a manufacturing system, process or any kind of dynamic system, in which the controller elements are not central in location (like the brain) but are distributed throughout the system with each component sub-system controlled by one or more controllers. The entire system of controllers is connected by networks for communication and monitoring.

DCS is a very broad term used in a variety of industries, to monitor and control distributed equipment.

- Electrical power grids and electrical generation plants
- Environmental control systems
- Traffic signals
- radio signals
- Water management systems
- Oil refining plants
- Metallurgical Process Plants
- Chemical plants
- Pharmaceutical manufacturing
- Sensor networks
- Dry cargo and bulk oil carrier ships

Elements

A DCS typically uses custom designed processors as controllers and uses both proprietary interconnections and communications protocol for communication. Input and output modules form component parts of the DCS. The processor receives information from input modules and sends information to output modules. The input modules receive information from input instruments in the process (a.k.a. field) and transmit instructions to the output instruments in the field. Computer buses or electrical buses connect the

processor and modules through multiplexer or demultiplexers. Buses also connect the distributed controllers with the central controller and finally to the Human-Machine Interface (HMI) or control consoles.

Elements of a distributed control system may directly connect to physical equipment such as switches, pumps and valves or may work through an intermediate system such as a SCADA system.

Applications

Distributed Control Systems (DCSs) are dedicated systems used to control manufacturing processes that are continuous or batch-oriented, such as oil refining, petrochemicals, central station power generation, pharmaceuticals, food & beverage manufacturing, cement production, steelmaking, and papermaking. DCSs are connected to sensors and actuators and use setpoint control to control the flow of material through the plant. The most common example is a setpoint control loop consisting of a pressure sensor, controller, and control valve. Pressure or flow measurements are transmitted to the controller, usually through the aid of a signal conditioning Input/Output (I/O) device. When the measured variable reaches a certain point, the controller instructs a valve or actuation device to open or close until the fluidic flow process reaches the desired setpoint. Large oil refineries have many thousands of I/O points and employ very large DCSs. Processes are not limited to fluidic flow through pipes, however, and can also include things like paper machines and their associated quality controls, variable speed drives and motor control centers, cement kilns, mining operations, ore processing facilities, and many others.

A typical DCS consists of functionally and/or geographically distributed digital controllers capable of executing from 1 to 256 or more regulatory control loops in one control box. The input/output devices (I/O) can be integral with the controller or located remotely via a field network. Today's controllers have extensive computational capabilities and, in addition to proportional, integral, and derivative (PID) control, can generally perform logic and sequential control.

DCSs may employ one or several workstations and can be configured at the workstation or by an off-line personal computer. Local communication is handled by a control network with transmission over twisted pair, coaxial, or fiber optic cable. A server and/or applications processor may be included in the system for extra computational, data collection, and reporting capability.

History

Early minicomputers were used in the control of industrial processes since the beginning of the 1960s. The IBM 1800, for example, was an early computer that had input/output hardware to gather process signals in a plant for conversion from field contact levels (for digital points) and analog signals to the digital domain.

The first industrial control computer system was built 1959 at the Texaco Port Arthur, Texas, refinery with an RW-300 of the Ramo-Wooldridge Company.

The DCS was introduced in 1975. Both Honeywell and Japanese electrical engineering firm Yokogawa introduced their own independently produced DCSs at roughly the same time, with the TDC 2000 and CENTUM systems, respectively. US-based Bristol also introduced their UCS 3000 universal controller in 1975. In 1980, Bailey (now part of ABB) introduced the NETWORK 90 system. Also in 1980, Fischer & Porter Company (now also part of ABB) introduced DCI-4000 (DCI stands for Distributed Control Instrumentation).

The DCS largely came about due to the increased availability of microcomputers and the proliferation of microprocessors in the world of process control. Computers had already been applied to process automation for some time in the form of both Direct Digital Control (DDC) and Set Point Control. In the early 1970s Taylor Instrument Company, (now part of ABB) developed the 1010 system, Foxboro the FOX1 system and Bailey Controls the 1055 systems. All of these were DDC applications implemented within minicomputers (DEC PDP-11, Varian Data Machines, MODCOMP etc.) and connected to proprietary Input/Output hardware. Sophisticated (for the time) continuous as well as batch control was implemented in this way. A more conservative approach was Set Point Control, where process computers supervised clusters of analog process controllers. A CRT-based workstation provided visibility into the process using text and crude character graphics. Availability of a fully functional graphical user interface was a way away.

Central to the DCS model was the inclusion of control function blocks. Function blocks evolved from early, more primitive DDC concepts of "Table Driven" software. One of the first embodiments of object-oriented software, function blocks were self contained "blocks" of code that emulated analog hardware control components and performed tasks that were essential to process control, such as execution of PID algorithms. Function blocks continue to endure as the predominant method of control for DCS suppliers, and are supported by key technologies such as **Foundation Fieldbus** today.

Midac Systems of Sydney Australia developed an objected-oriented distributed direct digital control system in 1982. The central system ran 11 microprocessors sharing tasks and common memory and connected to a serial communication network of distributed controllers each running two Z80s. The system was installed at the University of Melbourne.

Digital communication between distributed controllers, workstations and other computing elements (peer to peer access) was one of the primary advantages of the DCS. Attention was duly focused on the networks, which provided the all-important lines of communication that, for process applications, had to incorporate specific functions such as determinism and redundancy. As a result, many suppliers embraced the IEEE 802.4 networking standard. This decision set the stage for the wave of migrations necessary when information technology moved into process automation and IEEE 802.3 rather than IEEE 802.4 prevailed as the control LAN.

The Network Centric Era of the 1980s

The DCS brought distributed intelligence to the plant and established the presence of computers and microprocessors in process control, but it still did not provide the reach and openness necessary to unify plant resource requirements. In many cases, the DCS was merely a digital replacement of the same functionality provided by analog controllers and a panelboard display. This was embodied in The Purdue Reference Model (PRM) that was developed to define Manufacturing Operations Management relationships. PRM later formed the basis for ISA95 standards activities today.

In the 1980s, users began to look at DCSs as more than just basic process control. A very early example of a Direct Digital Control DCS was completed by the Australian business Midac in 1981-1982 using R-Tec Australian designed hardware. The system installed at the University of Melbourne used a serial communications network, connecting campus buildings back to a control room "front end". Each remote unit ran 2 Z80 microprocessors whilst the front end ran 11 in a Parallel Processing configuration with paged common memory to share tasks and could run up to 20,000 concurrent controls objects.

It was believed that if openness could be achieved and greater amounts of data could be shared throughout the enterprise that even greater things could be achieved. The first attempts to increase the openness of DCSs resulted in the adoption of the predominant operating system of the day: *UNIX*. UNIX and its companion networking technology TCP-IP were developed by the Department of Defense for openness, which was precisely the issue the process industries were looking to resolve.

As a result suppliers also began to adopt Ethernet-based networks with their own proprietary protocol layers. The full TCP/IP standard was not implemented, but the use of Ethernet made it possible to implement the first instances of object management and global data access technology. The 1980s also witnessed the first PLCs integrated into the DCS infrastructure. Plant-wide historians also emerged to capitalize on the extended reach of automation systems. The first DCS supplier to adopt UNIX and Ethernet networking technologies was Foxboro, who introduced the I/A Series system in 1987.

The Application Centric Era of the 1990s

The drive toward openness in the 1980s gained momentum through the 1990s with the increased adoption of Commercial off-the-shelf (COTS) components and IT standards. Probably the biggest transition undertaken during this time was the move from the UNIX operating system to the Windows environment. While the realm of the real time operating system (RTOS) for control applications remains dominated by real time commercial variants of UNIX or proprietary operating systems, everything above real-time control has made the transition to Windows.

The introduction of Microsoft at the desktop and server layers resulted in the development of technologies such as OLE for Process Control (OPC), which is now a de

facto industry connectivity standard. Internet technology also began to make its mark in automation and the DCS world, with most DCS HMI supporting Internet connectivity. The '90s were also known for the "Fieldbus Wars", where rival organizations competed to define what would become the IEC fieldbus standard for digital communication with field instrumentation instead of 4-20 milliamp analog communications. The first fieldbus installations occurred in the 1990s. Towards the end of the decade, the technology began to develop significant momentum, with the market consolidated around Ethernet I/P, Foundation Fieldbus and Profibus PA for process automation applications. Some suppliers built new systems from the ground up to maximize functionality with fieldbus, such as Rockwell PAX System Honeywell with Experion & Plantscape SCADA systems, ABB with System 800xA, Emerson Process Management with the DeltaV control system, Siemens] with the Simatic PCS7 and **azbil** from Yamatake with the Harmonas-DEO system.

The impact of COTS, however, was most pronounced at the hardware layer. For years, the primary business of DCS suppliers had been the supply of large amounts of hardware, particularly I/O and controllers. The initial proliferation of DCSs required the installation of prodigious amounts of this hardware, most of it manufactured from the bottom up by DCS suppliers. Standard computer components from manufacturers such as Intel and Motorola, however, made it cost prohibitive for DCS suppliers to continue making their own components, workstations, and networking hardware.

As the suppliers made the transition to COTS components, they also discovered that the hardware market was shrinking fast. COTS not only resulted in lower manufacturing costs for the supplier, but also steadily decreasing prices for the end users, who were also becoming increasingly vocal over what they perceived to be unduly high hardware costs. Some suppliers that were previously stronger in the PLC business, such as Rockwell Automation and Siemens, were able to leverage their expertise in manufacturing control hardware to enter the DCS marketplace with cost effective offerings, while the stability/scalability/reliability and functionality of these emerging systems are still improving. The traditional DCS suppliers introduced new generation DCS System based on the latest Communication and IEC Standards, which resulting in a trend of combining the traditional concepts/functionalities for PLC and DCS into a one for all solution—named "Process Automation System". The gaps among the various systems remain at the areas such as: the database integrity, pre-engineering functionality, system maturity, communication transparency and reliability. While it is expected the cost ratio is relatively the same (the more powerful the systems are, the more expensive they will be), the reality of the automation business is often operating strategically case by case. The current next evolution step is called Collaborative Process Automation Systems.

To compound the issue, suppliers were also realizing that the hardware market was becoming saturated. The lifecycle of hardware components such as I/O and wiring is also typically in the range of 15 to over 20 years, making for a challenging replacement market. Many of the older systems that were installed in the 1970s and 1980s are still in use today, and there is a considerable installed base of systems in the market that are approaching the end of their useful life. Developed industrial economies in North

America, Europe, and Japan already had many thousands of DCSs installed, and with few if any new plants being built, the market for new hardware was shifting rapidly to smaller, albeit faster growing regions such as China, Latin America, and Eastern Europe.

Because of the shrinking hardware business, suppliers began to make the challenging transition from a hardware-based business model to one based on software and value-added services. It is a transition that is still being made today. The applications portfolio offered by suppliers expanded considerably in the '90s to include areas such as production management, model-based control, real-time optimization, Plant Asset Management (PAM), Real Time Performance Management (RPM) tools, alarm management, and many others. To obtain the true value from these applications, however, often requires a considerable service content, which the suppliers also provide.

Chapter 7

High Redundancy Actuation

High Redundancy Actuation (HRA) is a new approach to fault tolerant control in the area of mechanical actuation.

Overview

The basic idea is to use a lot of small actuation elements, so that a fault of one element has only a minor effect on the overall system. This way, a High Redundancy Actuator can remain functional even after several elements are at fault. This property is also called graceful degradation.

Fault-tolerant operation in the presence of actuator faults requires some form of redundancy. Actuators are essential, because they are used to keep the system stable and to bring it into the desired state. Both requires a certain amount of power or force to be applied to the system. No control approach can work unless the actuators produce this necessary force.

So the common solution is to err on the side of safety by over-actuation: much more control action than strictly necessary is built into the system. For critical systems, the normal approach involves straightforward replication of the actuators. Often three or four actuators are used in parallel for aircraft flight control systems, even if one would be sufficient from a control point of view. So if one actuator failts, the remaining actuator can always keep the system operation. While this approach certainly successful, it also makes the system expensive, heavy and ineffective.

Inspiration of High Redundancy Actuation

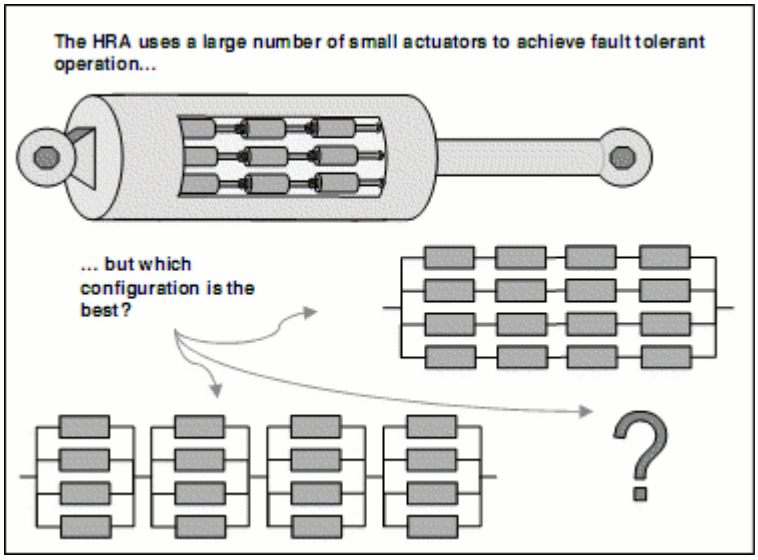
The idea of the High Redundancy Actuation (HRA) is inspired by the human musculature. A muscle is composed of many individual muscle cells, each of which provides only a minute contribution to the force and the travel of the muscle. These

properties allow the muscle as a whole to be highly resilient to damage of individual cells.

Technical Realisation

The aim of High Redundancy Actuation is not to produce man-made muscles, but to use the same principle of cooperation in technical actuator-s to provide intrinsic fault tolerance. To achieve this, a high number of small actuator elements are assembled in parallel and in series to form one actuator.

Faults within the actuator will affect the maximum capability, but through robust control, full performance can be maintained without either adaptation or reconfiguration. Some form of condition monitoring is necessary to provide warnings to the operator calling for maintenance. But this monitoring has no influence on the system itself, unlike in adaptive methods or control reconfiguration, which simplifies the design of the system significantly.



The HRA is an important new approach within the overall area of fault-tolerant control, using concepts of reliability engineering on a mechanical level. When applicable, it can provide actuators that have graceful degradation, and that continue to operate at close to nominal performance even in the presence of multiple faults in the actuator elements.

Using Actuation Elements in Series

An important feature of the High Redundancy Auctuation is that the actuator elements are connected both in parallel and in series. While the parallel arrangement is commonly used, the configuration in series is rarely employed, because it is perceived to be less efficient.

However, there is one fault that is difficult to deal with in a parallel arrangement: the locking up of one actuator element. Because parallel actuator elements always have the same extension, one locked-up element can render the whole assembly useless. It is possible to mitigate this by guarding the elements against locking or by limiting the force exerted by a single element. But these measures reduce both the effectiveness of the system and introduce new points of failure.

The analysis of the serial configuration shows that it remains operational when one element is locked-up. This fact is important for the High Redundancy Actuator, as fault tolerance is required for different fault types. The goal of the HRA project is to use parallel and serial actuator elements to accommodate both the blocking and the inactivity (loss of force) of an element.

Available Technology

The basic idea of High Redundancy Actuation is technology agnostic: it should be applicable to a wide range of actuator technology, including different kinds of linear actuators and rotational actuators.

However, initial experiments are performed with electric actuators, especially with electromechanical and electromagnetic technology. Compared to pneumatic actuators, the electrical drive allow a much finer control of position and force.

Chapter 8

Fault-Tolerant System

Fault-tolerance or **graceful degradation** is the property that enables a system (often computer-based) to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naïvely-designed system in which even a small failure can cause total breakdown. Fault-tolerance is particularly sought-after in high-availability or life-critical systems.

Fault-tolerance is not just a property of individual machines; it may also characterise the rules by which they interact. For example, the Transmission Control Protocol (TCP) is designed to allow reliable two-way communication in a packet-switched network, even in the presence of communications links which are imperfect or overloaded. It does this by requiring the endpoints of the communication to *expect* packet loss, duplication, reordering and corruption, so that these conditions do not damage data integrity, and only reduce throughput by a proportional amount.

An example of graceful degradation by design in an image with transparency. The top two images are each the result of viewing the composite image in a viewer that recognises transparency. The bottom two images are the result in a viewer with no support for transparency. Because the transparency mask (centre bottom) is discarded, only the overlay (centre top) remains; the image on the left has been designed to degrade gracefully, hence is still meaningful without its transparency information.

Data formats may also be designed to degrade gracefully. HTML for example, is designed to be forward compatible, allowing new HTML entities to be ignored by Web browsers which do not understand them without causing the document to be unusable.

Recovery from errors in fault-tolerant systems can be characterised as either **roll-forward** or **roll-back**. When the system detects that it has made an error, roll-forward recovery takes the system state at that time and corrects it, to be able to move forward. Roll-back recovery reverts the system state back to some earlier, correct version, for example using checkpointing, and moves forward from there. Roll-back recovery requires that the operations between the checkpoint and the detected erroneous state can be made idempotent. Some systems make use of both roll-forward and roll-back recovery for different errors or different parts of one error.

Within the scope of an *individual* system, fault-tolerance can be achieved by anticipating exceptional conditions and building the system to cope with them, and, in general, aiming for self-stabilization so that the system converges towards an error-free state. However, if the consequences of a system failure are catastrophic, or the cost of making it sufficiently reliable is very high, a better solution may be to use some form of duplication. In any case, if the consequence of a system failure is catastrophic, the system must be able to use reversion to fall back to a safe mode. This is similar to roll-back recovery but can be a human action if humans are present in the loop.

Fault tolerance requirements

The basic characteristics of fault tolerance require:

1. No single point of failure
2. Fault isolation to the failing component
3. Fault containment to prevent propagation of the failure
4. Availability of reversion modes

In addition, fault tolerant systems are characterized in terms of both planned service outages and unplanned service outages. These are usually measured at the application level and not just at a hardware level. The figure of merit is called availability and is expressed as a percentage. For example, a five nines system would statistically provide 99.999% availability.

Fault-tolerant systems are typically based on the concept of redundancy.

Fault-tolerance by replication

Spare components addresses the first fundamental characteristic of fault-tolerance in three ways:

- Replication: Providing multiple identical instances of the same system or subsystem, directing tasks or requests to all of them in parallel, and choosing the correct result on the basis of a quorum;
- Redundancy: Providing multiple identical instances of the same system and switching to one of the remaining instances in case of a failure (failover);
- Diversity: Providing multiple *different* implementations of the same specification, and using them like replicated systems to cope with errors in a specific implementation.

All implementations of RAID, redundant array of independent disks, except RAID 0 are examples of a fault-tolerant storage device that uses data redundancy.

A lockstep fault-tolerant machine uses replicated elements operating in parallel. At any time, all the replications of each element should be in the same state. The same inputs are provided to each replication, and the same outputs are expected. The outputs of the replications are compared using a voting circuit. A machine with two replications of each element is termed Dual Modular Redundant (DMR). The voting circuit can then only detect a mismatch and recovery relies on other methods. A machine with three replications of each element is termed Triple Modular Redundancy (TMR). The voting circuit can determine which replication is in error when a two-to-one vote is observed. In this case, the voting circuit can output the correct result, and discard the erroneous version. After this, the internal state of the erroneous replication is assumed to be different from that of the other two, and the voting circuit can switch to a DMR mode. This model can be applied to any larger number of replications.

Lockstep fault tolerant machines are most easily made fully synchronous, with each gate of each replication making the same state transition on the same edge of the clock, and the clocks to the replications being exactly in phase. However, it is possible to build lockstep systems without this requirement.

Bringing the replications into synchrony requires making their internal stored states the same. They can be started from a fixed initial state, such as the reset state. Alternatively, the internal state of one replica can be copied to another replica.

One variant of DMR is **pair-and-spare**. Two replicated elements operate in lockstep as a pair, with a voting circuit that detects any mismatch between their operations and outputs a signal indicating that there is an error. Another pair operates exactly the same way. A final circuit selects the output of the pair that does not proclaim that it is in error. Pair-and-spare requires four replicas rather than the three of TMR, but has been used commercially.

No single point of repair

If a system experiences a failure, it must continue to operate without interruption during the repair process.

Fault isolation to the failing component

When a failure occurs, the system must be able to isolate the failure to the offending component. This requires the addition of dedicated failure detection mechanisms that exist only for the purpose of fault isolation.

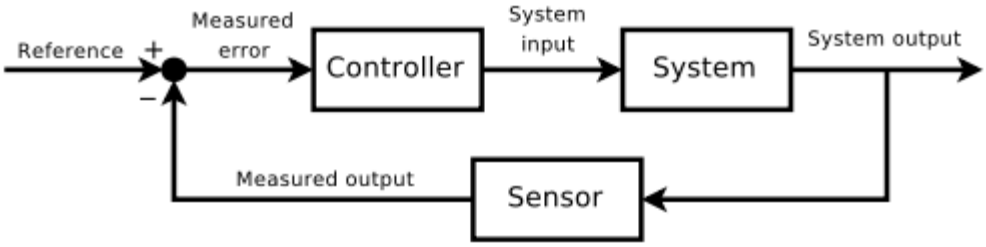
Recovery from a fault condition requires classifying the fault or failing component. The National Institute of Standards and Technology (NIST) categorizes faults based on Locality, Cause, Duration and Effect.

Fault containment

Some failure mechanisms can cause a system to fail by propagating the failure to the rest of the system. An example of this kind of failure is the "Rogue transmitter" which can swamp legitimate communication in a system and cause overall system failure. Mechanisms that isolate a rogue transmitter or failing component to protect the system are required.

Chapter 9

Control Theory



The concept of the feedback loop to control the dynamic behavior of the system: this is negative feedback, because the sensed value is subtracted from the desired value to create the error signal which is amplified by the controller.

Control theory is an interdisciplinary branch of engineering and mathematics, that deals with the behavior of dynamical systems. The desired output of a system is called the *reference*. When one or more output variables of a system need to follow a certain reference over time, a controller manipulates the inputs to a system to obtain the desired effect on the output of the system.

Overview

Control theory is

- a theory that deals with influencing the behavior of dynamical systems
- an interdisciplinary subfield of science, which originated in engineering and mathematics, and evolved into use by the social sciences, like psychology, sociology and criminology.

An example

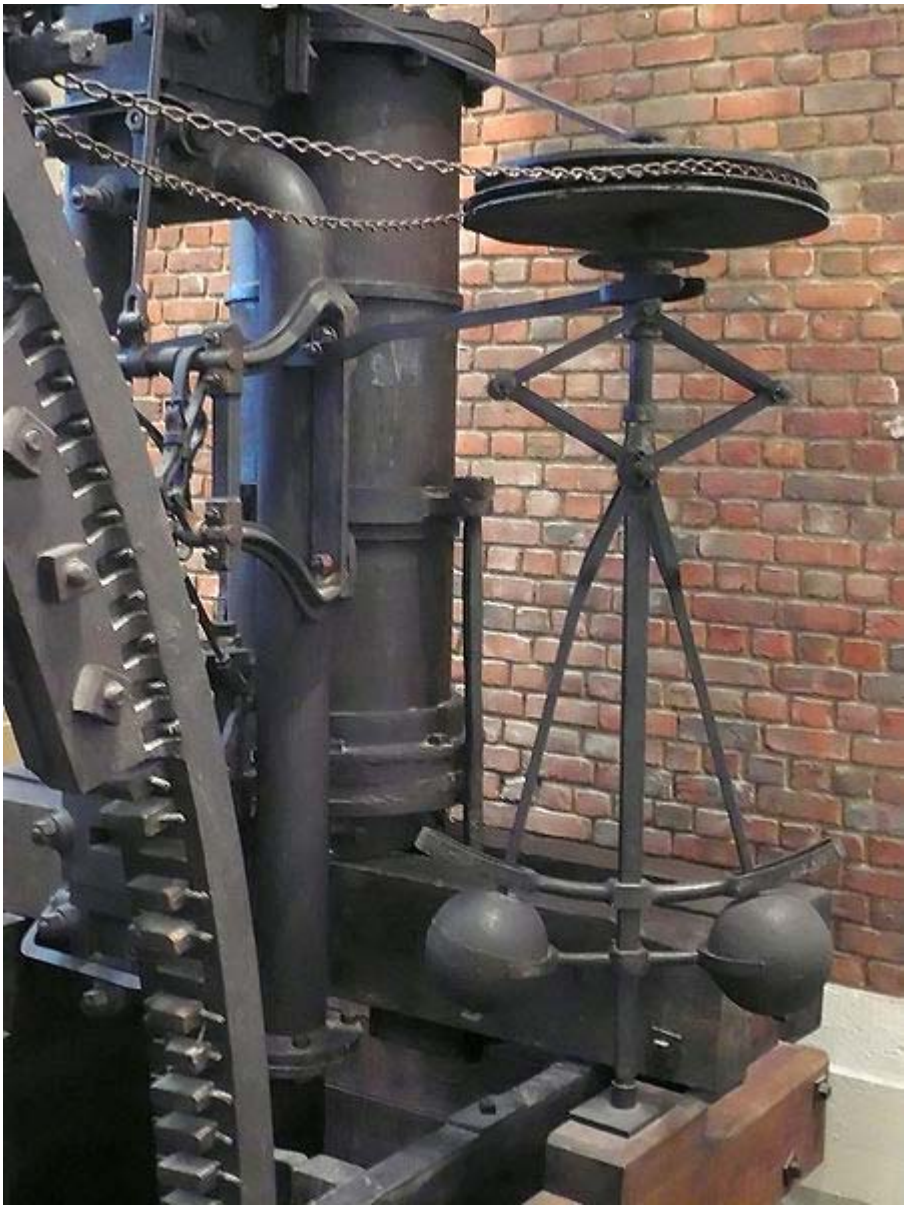
Consider a car's cruise control, which is a device designed to maintain vehicle speed at a constant *desired* or *reference* speed provided by the driver. The *controller* is the cruise

control, the *plant* is the car, and the *system* is the car and the cruise control. The system output is the car's speed, and the control itself is the engine's throttle position which determines how much power the engine generates.

A primitive way to implement cruise control is simply to lock the throttle position when the driver engages cruise control. However, if the cruise control is engaged on a stretch of flat road, then the car will travel slower going uphill and faster when going downhill. This type of controller is called an open-loop controller because no measurement of the system output (the car's speed) is used to alter the control (the throttle position.) As a result, the controller can not compensate for changes acting on the car, like a change in the slope of the road.

In a **closed-loop control system**, a sensor monitors the system output (the car's speed) and feeds the data to a controller which adjusts the control (the throttle position) as necessary to maintain the desired system output (match the car's speed to the reference speed.) Now when the car goes uphill the decrease in speed is measured, and the throttle position changed to increase engine power, speeding the vehicle. Feedback from measuring the car's speed has allowed the controller to dynamically compensate for changes to the car's speed. It is from this feedback that the paradigm of the control *loop* arises: the control affects the system output, which in turn is measured and looped back to alter the control.

History



Centrifugal governor in a Boulton & Watt engine of 1788

Although control systems of various types date back to antiquity, a more formal analysis of the field began with a dynamics analysis of the centrifugal governor, conducted by the physicist James Clerk Maxwell in 1868 entitled *On Governors*. This described and analyzed the phenomenon of "hunting", in which lags in the system can lead to overcompensation and unstable behavior. This generated a flurry of interest in the topic, during which Maxwell's classmate Edward John Routh generalized the results of Maxwell for the general class of linear systems. Independently, Adolf Hurwitz analyzed system stability using differential equations in 1877, resulting in what is now known as the Routh-Hurwitz theorem.

A notable application of dynamic control was in the area of manned flight. The Wright brothers made their first successful test flights on December 17, 1903 and were distinguished by their ability to control their flights for substantial periods (more so than the ability to produce lift from an airfoil, which was known). Control of the airplane was necessary for safe flight.

By World War II, control theory was an important part of fire-control systems, guidance systems and electronics.

Sometimes mechanical methods are used to improve the stability of systems. For example, ship stabilizers are fins mounted beneath the waterline and emerging laterally. In contemporary vessels, they may be gyroscopically controlled active fins, which have the capacity to change their angle of attack to counteract roll caused by wind or waves acting on the ship.

The Sidewinder missile uses small control surfaces placed at the rear of the missile with spinning disks on their outer surface; these are known as rollerons. Airflow over the disk spins them to a high speed. If the missile starts to roll, the gyroscopic force of the disk drives the control surface into the airflow, cancelling the motion. Thus the Sidewinder team replaced a potentially complex control system with a simple mechanical solution.

The Space Race also depended on accurate spacecraft control. However, control theory also saw an increasing use in fields such as economics.

People in systems and control

Many active and historical figures made significant contribution to control theory, including, for example:

- Alexander Lyapunov (1857–1918) in the 1890s marks the beginning of stability theory.
- Harold S. Black (1898–1983), invented the concept of negative feedback amplifiers in 1927. He managed to develop stable negative feedback amplifiers in the 1930s.
- Harry Nyquist (1889–1976), developed the Nyquist stability criterion for feedback systems in the 1930s.
- Richard Bellman (1920–1984), developed dynamic programming since the 1940s.
- Andrey Kolmogorov (1903–1987) co-developed the Wiener-Kolmogorov filter (1941).
- Norbert Wiener (1894–1964) co-developed the Wiener-Kolmogorov filter and coined the term cybernetics in the 1940s.
- John R. Ragazzini (1912–1988) introduced digital control and the z-transform in the 1950s.
- Lev Pontryagin (1908–1988) introduced the maximum principle and the bang-bang principle.

Classical control theory

To avoid the problems of the open-loop controller, control theory introduces feedback. A closed-loop controller uses feedback to control states or outputs of a dynamical system. Its name comes from the information path in the system: process inputs (e.g. voltage applied to an electric motor) have an effect on the process outputs (e.g. velocity or torque of the motor), which is measured with sensors and processed by the controller; the result (the control signal) is used as input to the process, closing the loop.

Closed-loop controllers have the following advantages over open-loop controllers:

- disturbance rejection (such as unmeasured friction in a motor)
- guaranteed performance even with model uncertainties, when the model structure does not match perfectly the real process and the model parameters are not exact
- unstable processes can be stabilized
- reduced sensitivity to parameter variations
- improved reference tracking performance

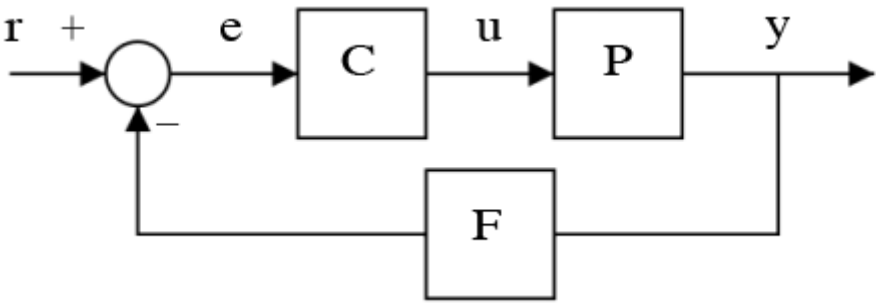
In some systems, closed-loop and open-loop control are used simultaneously. In such systems, the open-loop control is termed feedforward and serves to further improve reference tracking performance.

A common closed-loop controller architecture is the PID controller.

Closed-loop transfer function

The output of the system *y(t)* is fed back through a sensor measurement *F* to the reference value *r(t)*. The controller *C* then takes the error *e* (difference) between the reference and the output to change the inputs *u* to the system under control *P*. This is shown in the figure. This kind of controller is a closed-loop controller or feedback controller.

This is called a single-input-single-output (*SISO*) control system; *MIMO* (i.e. Multi-Input-Multi-Output) systems, with more than one input/output, are common. In such cases variables are represented through vectors instead of simple scalar values. For some distributed parameter systems the vectors may be infinite-dimensional (typically functions).



If we assume the controller *C*, the plant *P*, and the sensor *F* are linear and time-invariant (i.e.: elements of their transfer function *C(s)*, *P(s)*, and *F(s)* do not depend on time), the systems above can be analysed using the Laplace transform on the variables. This gives the following relations:

$$\begin{aligned} Y(s) &= P(s)U(s) \\ U(s) &= C(s)E(s) \\ E(s) &= R(s) - F(s)Y(s). \end{aligned}$$

Solving for $Y(s)$ in terms of $R(s)$ gives:

$$Y(s) = \left(\frac{P(s)C(s)}{1 + F(s)P(s)C(s)} \right) R(s) = H(s)R(s).$$

The expression $H(s) = \frac{P(s)C(s)}{1 + F(s)P(s)C(s)}$ is referred to as the *closed-loop transfer function* of the system. The numerator is the forward (open-loop) gain from *r* to *y*, and the denominator is one plus the gain in going around the feedback loop, the so-called loop gain. If $|P(s)C(s)| \gg 1$, i.e. it has a large norm with each value of *s*, and if $|F(s)| \approx 1$, then *Y(s)* is approximately equal to *R(s)* and the output closely tracks the reference input.

PID controller

The PID controller is probably the most-used feedback control design. *PID* is an acronym for *Proportional-Integral-Differential*, referring to the three terms operating on the error signal to produce a control signal. If *u(t)* is the control signal sent to the system, *y(t)* is the measured output and *r(t)* is the desired output, and tracking error $e(t) = r(t) - y(t)$, a PID controller has the general form

$$u(t) = K_P e(t) + K_I \int e(t)\mathrm{d}t + K_D \frac{\mathrm{d}}{\mathrm{d}t} e(t).$$

The desired closed loop dynamics is obtained by adjusting the three parameters K_P, K_I and K_D, often iteratively by "tuning" and without specific knowledge of a plant model. Stability can often be ensured using only the proportional term. The integral term permits the rejection of a step disturbance (often a striking specification in process control). The derivative term is used to provide damping or shaping of the response. PID controllers are the most well established class of control systems: however, they cannot be used in several more complicated cases, especially if MIMO systems are considered.

Applying Laplace transformation results in the transformed PID controller equation

$$u(s) = K_P e(s) + K_I \frac{1}{s} e(s) + K_D s e(s)$$
$$u(s) = (K_P + K_I \frac{1}{s} + K_D s) e(s)$$

with the PID controller transfer function

$$C(s) = (K_P + K_I \frac{1}{s} + K_D s).$$

Modern control theory

In contrast to the frequency domain analysis of the classical control theory, modern control theory utilizes the time-domain state space representation, a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations. To abstract from the number of inputs, outputs and states, the variables are expressed as vectors and the differential and algebraic equations are written in matrix form (the latter only being possible when the dynamical system is linear). The state space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyze systems with multiple inputs and outputs. With inputs and outputs, we would otherwise have to write down Laplace transforms to encode all the information about a system. Unlike the frequency domain approach, the use of the state space representation is not limited to systems with linear components and zero initial conditions. "State space" refers to the space whose axes are the state variables. The state of the system can be represented as a vector within that space.)

Topics in control theory

Stability

The *stability* of a general dynamical system with no input can be described with Lyapunov stability criteria. A linear system that takes an input is called bounded-input bounded-output (BIBO) stable if its output will stay bounded for any bounded input. Stability for nonlinear systems that take an input is input-to-state stability (ISS), which combines Lyapunov stability and a notion similar to BIBO stability. For simplicity, the following descriptions focus on continuous-time and discrete-time linear systems.

Mathematically, this means that for a causal linear system to be stable all of the poles of its transfer function must satisfy some criteria depending on whether a continuous or discrete time analysis is used:

- In continuous time, the Laplace transform is used to obtain the transfer function. A system is stable if the poles of this transfer function lie strictly in the open left half of the complex plane (i.e. the real part of all the poles is less than zero).

- In discrete time the Z-transform is used. A system is stable if the poles of this transfer function lie strictly inside the unit circle. i.e. the magnitude of the poles is less than one).

When the appropriate conditions above are satisfied a system is said to be asymptotically stable: the variables of an asymptotically stable control system always decrease from their initial value and do not show permanent oscillations. Permanent oscillations occur when a pole has a real part exactly equal to zero (in the continuous time case) or a modulus equal to one (in the discrete time case). If a simply stable system response neither decays nor grows over time, and has no oscillations, it is marginally stable: in this case the system transfer function has non-repeated poles at complex plane origin (i.e. their real and complex component is zero in the continuous time case). Oscillations are present when poles with real part equal to zero have an imaginary part not equal to zero.

Differences between the two cases are not a contradiction. The Laplace transform is in Cartesian coordinates and the Z-transform is in circular coordinates, and it can be shown that:

- the negative-real part in the Laplace domain can map onto the interior of the unit circle
- the positive-real part in the Laplace domain can map onto the exterior of the unit circle

If a system in question has an impulse response of

$$x[n] = 0.5^n u[n]$$

then the Z-transform (see this example), is given by

$$X(z) = \frac{1}{1 - 0.5z^{-1}}$$

which has a pole in $z = 0.5$ (zero imaginary part). This system is BIBO (asymptotically) stable since the pole is *inside* the unit circle.

However, if the impulse response was

$$x[n] = 1.5^n u[n]$$

then the Z-transform is

$$X(z) = \frac{1}{1 - 1.5z^{-1}}$$

which has a pole at $z = 1.5$ and is not BIBO stable since the pole has a modulus strictly greater than one.

Numerous tools exist for the analysis of the poles of a system. These include graphical systems like the root locus, Bode plots or the Nyquist plots.

Mechanical changes can make equipment (and control systems) more stable. Sailors add ballast to improve the stability of ships. Cruise ships use antiroll fins that extend transversely from the side of the ship for perhaps 30 feet (10 m) and are continuously rotated about their axes to develop forces that oppose the roll.

Controllability and observability

Controllability and observability are main issues in the analysis of a system before deciding the best control strategy to be applied, or whether it is even possible to control or stabilize the system. Controllability is related to the possibility of forcing the system into a particular state by using an appropriate control signal. If a state is not controllable, then no signal will ever be able to control the state. If a state is not controllable, but its dynamics are stable, then the state is termed Stabilizable. Observability instead is related to the possibility of "observing", through output measurements, the state of a system. If a state is not observable, the controller will never be able to determine the behaviour of an unobservable state and hence cannot use it to stabilize the system. However, similar to the stabilizability condition above, if a state cannot be observed it might still be detectable.

From a geometrical point of view, looking at the states of each variable of the system to be controlled, every "bad" state of these variables must be controllable and observable to ensure a good behaviour in the closed-loop system. That is, if one of the eigenvalues of the system is not both controllable and observable, this part of the dynamics will remain untouched in the closed-loop system. If such an eigenvalue is not stable, the dynamics of this eigenvalue will be present in the closed-loop system which therefore will be unstable. Unobservable poles are not present in the transfer function realization of a state-space representation, which is why sometimes the latter is preferred in dynamical systems analysis.

Solutions to problems of uncontrollable or unobservable system include adding actuators and sensors.

Control specification

Several different control strategies have been devised in the past years. These vary from extremely general ones (PID controller), to others devoted to very particular classes of systems (especially robotics or aircraft cruise control).

A control problem can have several specifications. Stability, of course, is always present: the controller must ensure that the closed-loop system is stable, regardless of the open-loop stability. A poor choice of controller can even worsen the stability of the open-loop system, which must normally be avoided. Sometimes it would be desired to obtain

particular dynamics in the closed loop: i.e. that the poles have $Re[\lambda] < -\overline{\lambda}$, where $\overline{\lambda}$ is a fixed value strictly greater than zero, instead of simply asking that $Re[\lambda] < 0$.

Another typical specification is the rejection of a step disturbance; including an integrator in the open-loop chain (i.e. directly before the system under control) easily achieves this. Other classes of disturbances need different types of sub-systems to be included.

Other "classical" control theory specifications regard the time-response of the closed-loop system: these include the rise time (the time needed by the control system to reach the desired value after a perturbation), peak overshoot (the highest value reached by the response before reaching the desired value) and others (settling time, quarter-decay). Frequency domain specifications are usually related to robustness (see after).

Modern performance assessments use some variation of integrated tracking error (IAE,ISA,CQI).

Model identification and robustness

A control system must always have some robustness property. A robust controller is such that its properties do not change much if applied to a system slightly different from the mathematical one used for its synthesis. This specification is important: no real physical system truly behaves like the series of differential equations used to represent it mathematically. Typically a simpler mathematical model is chosen in order to simplify calculations, otherwise the true system dynamics can be so complicated that a complete model is impossible.

System identification

The process of determining the equations that govern the model's dynamics is called system identification. This can be done off-line: for example, executing a series of measures from which to calculate an approximated mathematical model, typically its transfer function or matrix. Such identification from the output, however, cannot take account of unobservable dynamics. Sometimes the model is built directly starting from known physical equations: for example, in the case of a mass-spring-damper system we know that $m\ddot{x}(t) = -Kx(t) - B\dot{x}(t)$. Even assuming that a "complete" model is used in designing the controller, all the parameters included in these equations (called "nominal parameters") are never known with absolute precision; the control system will have to behave correctly even when connected to physical system with true parameter values away from nominal.

Some advanced control techniques include an "on-line" identification process (see later). The parameters of the model are calculated ("identified") while the controller itself is running: in this way, if a drastic variation of the parameters ensues (for example, if the robot's arm releases a weight), the controller will adjust itself consequently in order to ensure the correct performance.

Analysis

Analysis of the robustness of a SISO control system can be performed in the frequency domain, considering the system's transfer function and using Nyquist and Bode diagrams. Topics include gain and phase margin and amplitude margin. For MIMO and, in general, more complicated control systems one must consider the theoretical results devised for each control technique: i.e., if particular robustness qualities are needed, the engineer must shift his attention to a control technique including them in its properties.

Constraints

A particular robustness issue is the requirement for a control system to perform properly in the presence of input and state constraints. In the physical world every signal is limited. It could happen that a controller will send control signals that cannot be followed by the physical system: for example, trying to rotate a valve at excessive speed. This can produce undesired behavior of the closed-loop system, or even break actuators or other subsystems. Specific control techniques are available to solve the problem: model predictive control (see later), and anti-wind up systems. The latter consists of an additional control block that ensures that the control signal never exceeds a given threshold.

System classifications

Linear Systems control

For MIMO systems, pole placement can be performed mathematically using a state space representation of the open-loop system and calculating a feedback matrix assigning poles in the desired positions. In complicated systems this can require computer-assisted calculation capabilities, and cannot always ensure robustness. Furthermore, all system states are not in general measured and so observers must be included and incorporated in pole placement design.

Nonlinear Systems control

Processes in industries like robotics and the aerospace industry typically have strong nonlinear dynamics. In control theory it is sometimes possible to linearize such classes of systems and apply linear techniques, but in many cases it can be necessary to devise from scratch theories permitting control of nonlinear systems. These, e.g., feedback linearization, backstepping, sliding mode control, trajectory linearization control normally take advantage of results based on Lyapunov's theory. Differential geometry has been widely used as a tool for generalizing well-known linear control concepts to the non-linear case, as well as showing the subtleties that make it a more challenging problem.

Decentralized Systems

When the system is controlled by multiple controllers, the problem is one of decentralized control. Decentralization is helpful in many ways, for instance, it helps control systems operate over a larger geographical area. The agents in decentralized control systems can interact using communication channels and coordinate their actions.

Main control strategies

Every control system must guarantee first the stability of the closed-loop behavior. For linear systems, this can be obtained by directly placing the poles. Non-linear control systems use specific theories (normally based on Aleksandr Lyapunov's Theory) to ensure stability without regard to the inner dynamics of the system. The possibility to fulfill different specifications varies from the model considered and the control strategy chosen. Here a summary list of the main control techniques is shown:

Adaptive control
: Adaptive control uses on-line identification of the process parameters, or modification of controller gains, thereby obtaining strong robustness properties. Adaptive controls were applied for the first time in the aerospace industry in the 1950s, and have found particular success in that field.

Hierarchical control
: A Hierarchical control system is a type of Control System in which a set of devices and governing software is arranged in a hierarchical tree. When the links in the tree are implemented by a computer network, then that hierarchical control system is also a form of Networked control system.

Intelligent control
: Intelligent control uses various AI computing approaches like neural networks, Bayesian probability, fuzzy logic, machine learning, evolutionary computation and genetic algorithms to control a dynamic system.

Optimal control
: Optimal control is a particular control technique in which the control signal optimizes a certain "cost index": for example, in the case of a satellite, the jet thrusts needed to bring it to desired trajectory that consume the least amount of fuel. Two optimal control design methods have been widely used in industrial applications, as it has been shown they can guarantee closed-loop stability. These are Model Predictive Control (MPC) and Linear-Quadratic-Gaussian control (LQG). The first can more explicitly take into account constraints on the signals in the system, which is an important feature in many industrial processes. However, the "optimal control" structure in MPC is only a means to achieve such a result, as it does not optimize a true performance index of the closed-loop control system. Together with PID controllers, MPC systems are the most widely used control technique in process control.

Robust control
: Robust control deals explicitly with uncertainty in its approach to controller design. Controllers designed using *robust control* methods tend to be able to cope

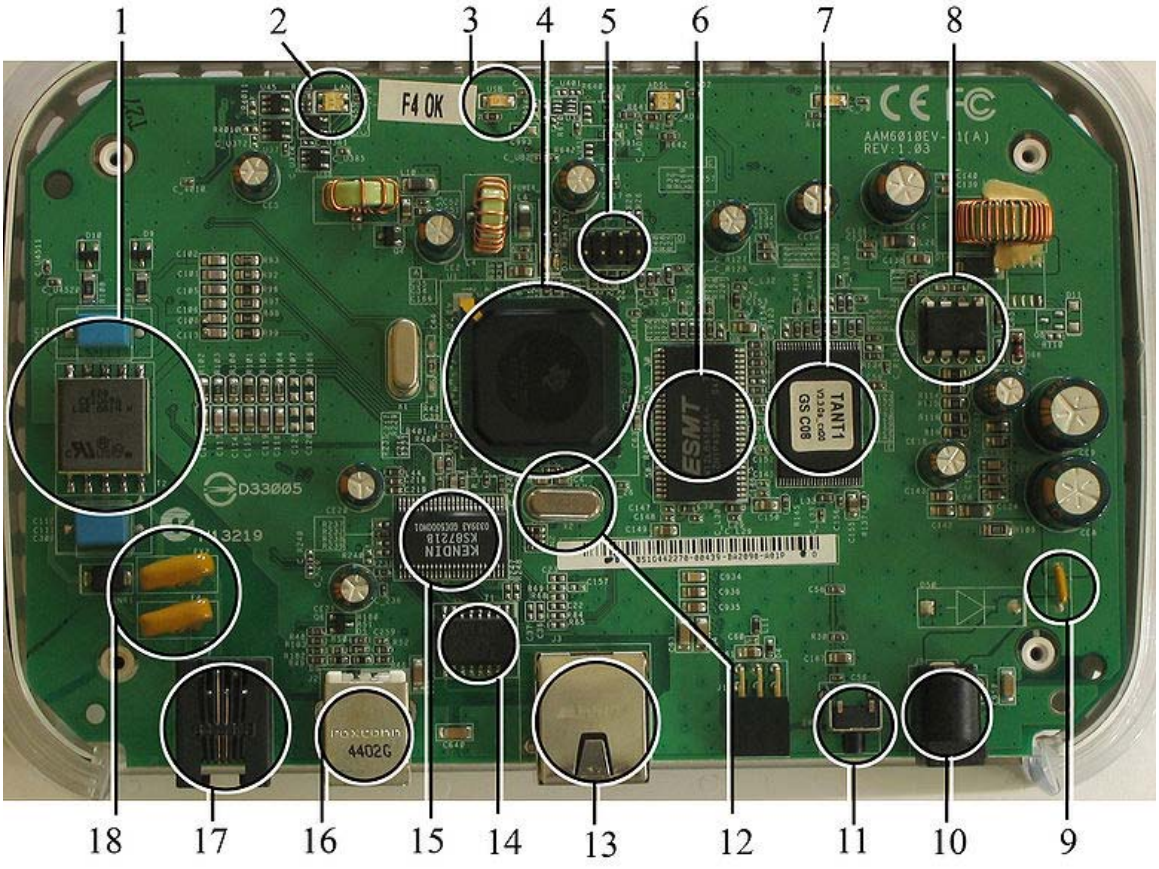
with small differences between the true system and the nominal model used for design. The early methods of Bode and others were fairly robust; the state-space methods invented in the 1960s and 1970s were sometimes found to lack robustness. A modern example of a robust control technique is H-infinity loop-shaping developed by Duncan McFarlane and Keith Glover of Cambridge University, United Kingdom. Robust methods aim to achieve robust performance and/or stability in the presence of small modeling errors.

Stochastic control

Stochastic control deals with control design with uncertainty in the model. In typical stochastic control problems, it is assumed that there exist random noise and disturbances in the model and the controller, and the control design must take into account these random deviations.

Chapter 10

Embedded System



Picture of the internals of an ADSL modem/router. A modern example of an embedded system. Labelled parts include a microprocessor (4), RAM (6), and flash memory (7).

An **embedded system** is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is *embedded* as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

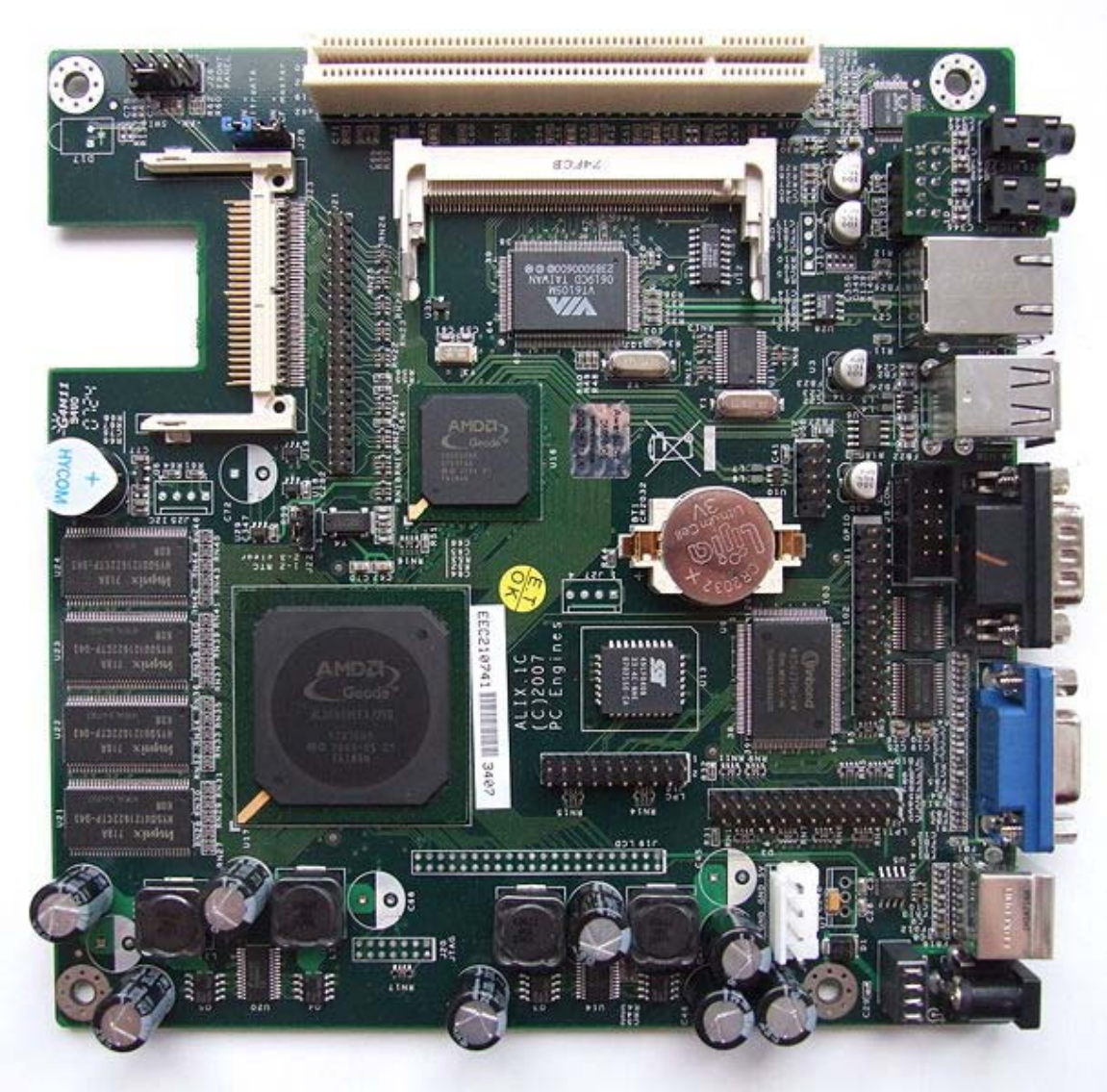
Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites (each radar probably includes one or more embedded systems of its own).

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

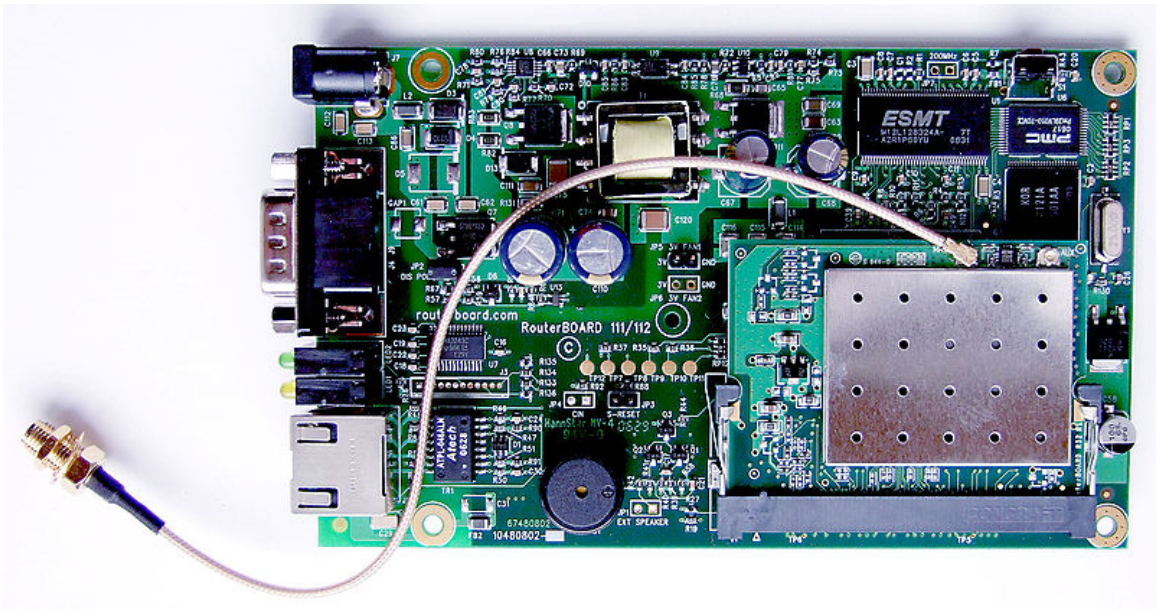
Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, "embedded system" is not a strictly definable term, as most systems have some element of extensibility or programmability. For example, handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which do not expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus appropriate to call "embedded".

Variety of embedded systems



PC Engines' ALIX.1C Mini-ITX embedded board with an x86 AMD Geode LX 800 together with Compact Flash, miniPCI and PCI slots, 44-pin IDE interface, audio, USB and 256MB RAM



An embedded RouterBoard 112 with U.FL-RSMA pigtail and R52 miniPCI Wi-Fi card widely used by wireless Internet service providers (WISPs) in the Czech Republic.

Embedded systems span all aspects of modern life and there are many examples of their use.

Telecommunications systems employ numerous embedded systems from telephone switches for the network to mobile phones at the end-user. Computer networking uses dedicated routers and network bridges to route data.

Consumer electronics include personal digital assistants (PDAs), mp3 players, mobile phones, videogame consoles, digital cameras, DVD players, GPS receivers, and printers. Many household appliances, such as microwave ovens, washing machines and dishwashers, are including embedded systems to provide flexibility, efficiency and features. Advanced HVAC systems use networked thermostats to more accurately and efficiently control temperature that can change by time of day and season. Home automation uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded devices for sensing and controlling.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers that also have considerable safety requirements. Various electric motors — brushless DC motors, induction motors and DC motors — are using electric/electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles are increasingly using embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive.

Medical equipment is continuing to advance with more embedded systems for vital signs monitoring, electronic stethoscopes for amplifying sounds, and various medical imaging (PET, SPECT, CT, MRI) for non-invasive internal inspections.

Embedded systems are especially suited for use in transportation, fire safety, safety and security, medical applications and life critical systems as these systems can be isolated from hacking and thus be more reliable. For fire safety, the systems can be designed to be have greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self sufficient and be able to deal with cut electrical and communication systems.

In addition to commonly described embedded systems based on small computers, a new class of miniature wireless devices called motes are quickly gaining popularity as the field of wireless sensor networking rises. Wireless sensor networking, WSN, makes use of miniaturization made possible by advanced IC design to couple full wireless subsystems to sophisticated sensors, enabling people and companies to measure a myriad of things in the physical world and act on this information through IT monitoring and control systems. These motes are completely self contained, and will typically run off a battery source for many years before the batteries need to be changed or charged.

History

One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961. It was built from transistor logic and had a hard disk for main memory. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits. This program alone reduced prices on quad nand gate ICs from $1000/each to $3/each, permitting their use in commercial products.

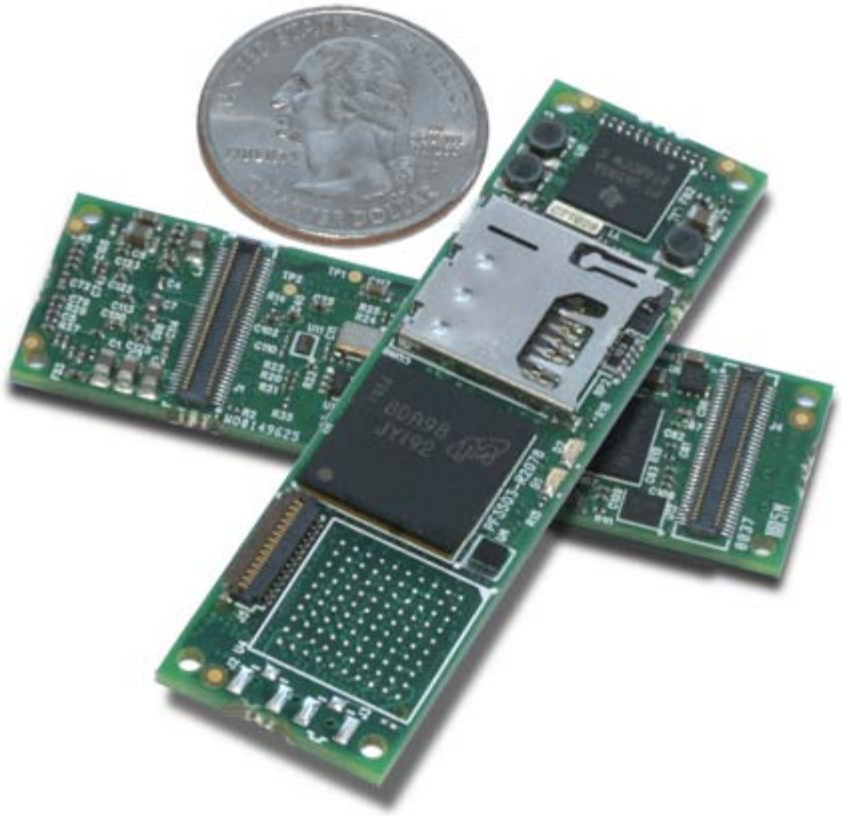
Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. The first microprocessor for example, the Intel 4004, was designed for calculators and other small systems but still required many external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer-based controllers, such as single board computers, numerical, and event-based controllers.

As the cost of microprocessors and microcontrollers fell it became feasible to replace expensive knob-based analog components such as potentiometers and variable capacitors with up/down buttons or knobs read out by a microprocessor even in some consumer products. By the mid-1980s, most of the common previously external system components had been integrated into the same chip as the processor and this modern form of the

microcontroller allowed an even more widespread use, which by the end of the decade were the norm rather than the exception for almost all electronics devices.

The integration of microcontrollers has further increased the applications for which embedded systems are used into areas where traditionally a computer would not have been considered. A general purpose and comparatively low-cost microcontroller may often be programmed to fulfill the same role as a large number of separate components. Although in this context an embedded system is usually more complex than a traditional solution, most of the complexity is contained within the microcontroller itself. Very few additional components may be needed and most of the design effort is in the software. The intangible nature of software makes it much easier to prototype and test new revisions compared with the design and construction of a new circuit not using an embedded processor.

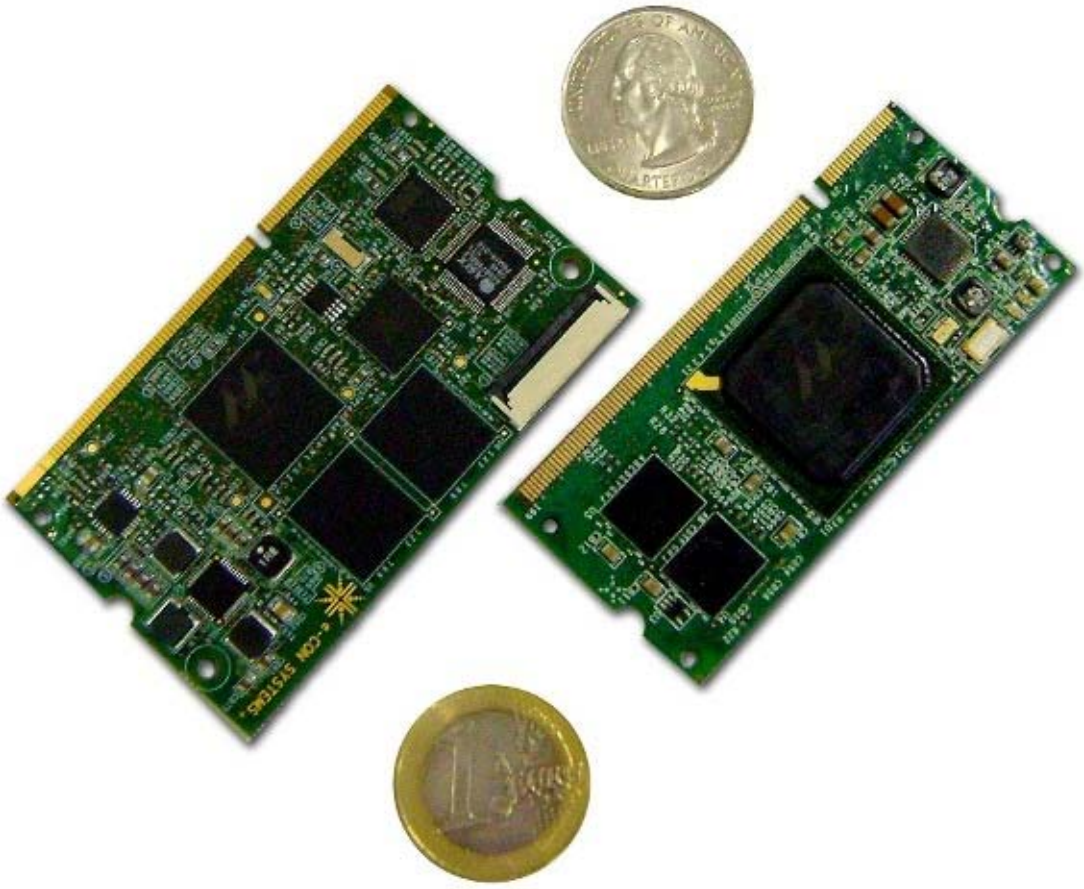
Characteristics



Gumstix Overo COM, a tiny, OMAP-based embedded computer-on-module with Wifi and Bluetooth.

1. Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

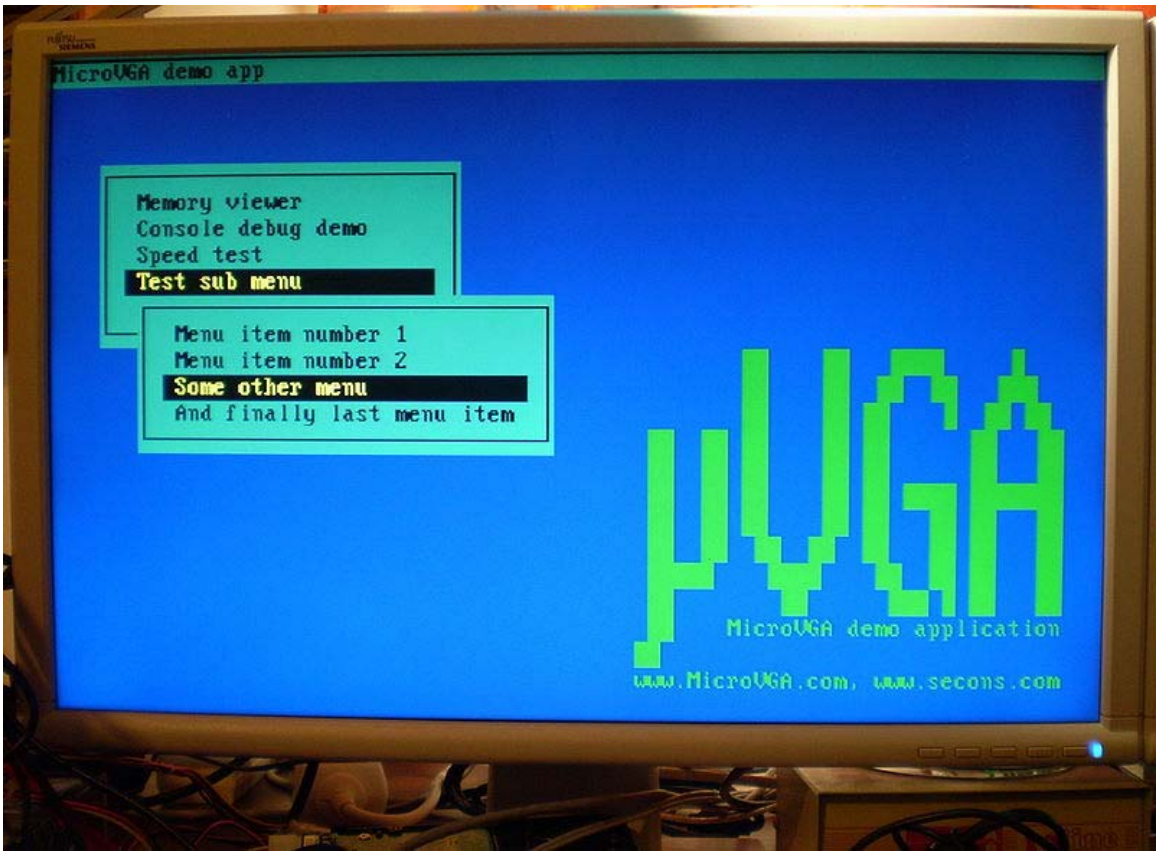
2. Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. For example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an automobile provides a specific function as a subsystem of the car itself.



e-con Systems eSOM270 & eSOM300 Computer on Modules

3. The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard and/or screen.

User interface



Embedded system text user interface using MicroVGA

Embedded systems range from no user interface at all — dedicated only to one task — to complex graphical user interfaces that resemble modern computer desktop operating systems. Simple embedded devices use buttons, LEDs, graphic or character LCDs (for example popular HD44780 LCD) with a simple menu system.

More sophisticated devices use graphical screen with touch sensing or screen-edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what's desired. Handheld systems often have a screen with a "joystick button" for a pointing device.

Some systems provide user interface remotely with the help of a serial (e.g. RS-232, USB, I²C, etc.) or network (e.g. Ethernet) connection. In spite of the potentially necessary proprietary client software and/or specialist cables that are needed, this approach usually gives a lot of advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies BSP, allows to build rich user interface on the PC. A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network routers. The user interface is displayed in a web

browser on a PC connected to the device, therefore needing no bespoke software to be installed.

Processors in embedded systems

Secondly, Embedded processors can be broken into two broad categories: ordinary microprocessors (μP) and microcontrollers (μC), which have many more peripherals on chip, reducing cost and size. Contrasting to the personal computer and server markets, a fairly large number of basic CPU architectures are used; there are Von Neumann as well as various degrees of Harvard architectures, RISC as well as non-RISC and VLIW; word lengths vary from 4-bit to 64-bits and beyond (mainly in DSP processors) although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

A long but still not exhaustive list of common architectures are: 65816, 65C02, 68HC08, 68HC11, 68k, 78K0R/78K0, 8051, ARM, AVR, AVR32, Blackfin, C167, Coldfire, COP8, Cortus APS3, eZ8, eZ80, FR-V, H8, HT48, M16C, M32C, MIPS, MSP430, PIC, PowerPC, R8C, RL78, SHARC, SPARC, ST6, SuperH, TLCS-47, TLCS-870, TLCS-900, Tricore, V850, x86, XE8000, Z80, AsAP etc.

Ready made computer boards

PC/104 and PC/104+ are examples of standards for *ready made* computer boards intended for small, low-volume embedded and ruggedized systems, mostly x86-based. These are often physically small compared to a standard PC, although still quite large compared to most simple (8/16-bit) embedded systems. They often use MSDOS, Linux, NetBSD, or an embedded real-time operating system such as MicroC/OS-II, QNX or VxWorks. Sometimes these boards use non-x86 processors.

In certain applications, where small size or power efficiency are not primary concerns, the components used may be compatible with those used in general purpose x86 personal computers. Boards such as the VIA EPIA range help to bridge the gap by being PC-compatible but highly integrated, physically smaller or have other attributes making them attractive to embedded engineers. The advantage of this approach is that low-cost commodity components may be used along with the same software development tools used for general software development. Systems built in this way are still regarded as embedded since they are integrated into larger devices and fulfill a single role. Examples of devices that may adopt this approach are ATMs and arcade machines, which contain code specific to the application.

However, most ready-made embedded systems boards are not PC-centered and do not use the ISA or PCI busses. When a System-on-a-chip processor is involved, there may be little benefit to having a standarized bus connecting discrete components, and the environment for both hardware and software tools may be very different.

One common design style uses a small system module, perhaps the size of a business card, holding high density BGA chips such as an ARM-based System-on-a-chip processor and peripherals, external flash memory for storage, and DRAM for runtime memory. The module vendor will usually provide boot software and make sure there is a selection of operating systems, usually including Linux and some real time choices. These modules can be manufactured in high volume, by organizations familiar with their specialized testing issues, and combined with much lower volume custom mainboards with application-specific external peripherals. Gumstix product lines are a Linux-centric example of this model.

ASIC and FPGA solutions

A common array of n configuration for very-high-volume embedded systems is the system on a chip (SoC) which contains a complete system consisting of multiple processors, multipliers, caches and interfaces on a single chip. SoCs can be implemented as an application-specific integrated circuit (ASIC) or using a field-programmable gate array (FPGA).

Peripherals

Embedded Systems talk with the outside world via peripherals, such as:

- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485 etc.
- Synchronous Serial Communication Interface: I2C, SPI, SSC and ESSI (Enhanced Synchronous Serial Interface)
- Universal Serial Bus (USB)
- Multi Media Cards (SD Cards, Compact Flash etc.)
- Networks: Ethernet, LonWorks, etc.
- Fieldbuses: CAN-Bus, LIN-Bus, PROFIBUS, etc.
- Timers: PLL(s), Capture/Compare and Time Processing Units
- Discrete IO: aka General Purpose Input/Output (GPIO)
- Analog to Digital/Digital to Analog (ADC/DAC)
- Debugging: JTAG, ISP, ICSP, BDM Port, BITP, and DP9 ports.

Tools

As with other software, embedded system designers use compilers, assemblers, and debuggers to develop embedded system software. However, they may also use some more specific tools:

- In circuit debuggers or emulators.
- Utilities to add a checksum or CRC to a program, so the embedded system can check if the program is valid.
- For systems using digital signal processing, developers may use a math workbench such as Scilab / Scicos, MATLAB / Simulink, EICASLAB, MathCad, Mathematica,or FlowStone DSP to simulate the mathematics. They might also use

libraries for both the host and target which eliminates developing DSP routines as done in DSPnano RTOS and Unison Operating System.
- Custom compilers and linkers may be used to improve optimisation for the particular hardware.
- An embedded system may have its own special language or design tool, or add enhancements to an existing language such as Forth or Basic.
- Another alternative is to add a real-time operating system or embedded operating system, which may have DSP capabilities like DSPnano RTOS.
- Modeling and code generating tools often based on state machines

Software tools can come from several sources:

- Software companies that specialize in the embedded market
- Ported from the GNU software development tools
- Sometimes, development tools for a personal computer can be used if the embedded processor is a close relative to a common PC processor

As the complexity of embedded systems grows, higher level tools and operating systems are migrating into machinery where it makes sense. For example, cellphones, personal digital assistants and other consumer computers often need significant software that is purchased or provided by a person other than the manufacturer of the electronics. In these systems, an open programming environment such as Linux, NetBSD, OSGi or Embedded Java is required so that the third-party software provider can sell to a large market.

Debugging

Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most sophisticated they can be roughly grouped into the following areas:

- Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)
- External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multicore systems.
- An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
- An in-circuit emulator (ICE) replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
- A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified, and allowing debugging on a normal PC.

Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as HLL source-code, assembly code or mixture of both.

Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software- (and microprocessor-) centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, co-processor). An increasing number of embedded systems today use more than one single processor core. A common problem with multi-core development is the proper synchronization of software execution. In such a case, the embedded system design may wish to check the data traffic on the busses between the processor cores, which requires very low-level debugging, at signal/bus level, with a logic analyzer, for instance.

Tracing Real-time operating systems (RTOS) often supports tracing of operating system events. A graphical view is presented by a host PC tool, based on a recording of the system behavior. The trace recording can be performed in software, by the RTOS, or by special tracing hardware. RTOS tracing allows developers to understand timing and performance issues of the software system and gives a good understanding of the high-level system behavior. A good example is RTXCview, for RTXC Quadros by Quadros Systems, Inc..

Reliability

Embedded systems often reside in machines that are expected to run continuously for years without errors, and in some cases recover by themselves if an error occurs. Therefore the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided.

Specific reliability issues may include:

1. The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
2. The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals, engines on single-engine aircraft.
3. The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.

A variety of techniques are used, sometimes in combination, to recover from errors—both software bugs such as memory leaks, and also soft errors in the hardware:

- watchdog timer that resets the computer unless the software periodically notifies the watchdog
- subsystems with redundant spares that can be switched over to
- software "limp modes" that provide partial function
- Designing with a Trusted Computing Base (TCB) architecture ensures a highly secure & reliable system environment
- An Embedded Hypervisor is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.
- Immunity Aware Programming

High vs low volume

For high volume systems such as portable music players or mobile phones, minimizing cost is usually the primary design consideration. Engineers typically select hardware that is just “good enough” to implement the necessary functions.

For low-volume or prototype embedded systems, general purpose computers may be adapted by limiting the programs or by replacing the operating system with a real-time operating system.

Embedded software architectures

There are several different types of software architecture in common use.

Simple control loop

In this design, the software simply has a loop. The loop calls subroutines, each of which manages a part of the hardware or software.

Interrupt controlled system

Some embedded systems are predominantly interrupt controlled. This means that tasks performed by the system are triggered by different kinds of events. An interrupt could be generated for example by a timer in a predefined frequency, or by a serial port controller receiving a byte.

These kinds of systems are used if event handlers need low latency and the event handlers are short and simple.

Usually these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays.

Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

Cooperative multitasking

A nonpreemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to “run” in. When a task is idle, it calls an idle routine, usually called “pause”, “wait”, “yield”, “nop” (stands for *no operation*), etc.

The advantages and disadvantages are very similar to the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue-interpreter.

Preemptive multitasking or multi-threading

In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non-blocking synchronization scheme.

Because of these complexities, it is common for organizations to use a real-time operating system (RTOS), allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a *generic* real time system, due to limitations regarding memory size, performance, and/or battery life.The choice that a RTOS is required brings in its own issues however as the selection must be done prior to starting to the application development process. This timing forces developers to choose the embedded operating system for their device based upon current requirements and so restricts future options to a large extent. The restriction of future options becomes more of an issue as product life decreases. Additionally the level of complexity is continuously growing as devices are required to manage many variables such as serial, USB, TCP/IP, Bluetooth, Wireless LAN, trunk radio, multiple channels, data and voice, enhanced graphics, multiple states, multiple threads, numerous wait states and so on. These trends are leading to the uptake of embedded middleware in addition to a real time operating system.

Microkernels and exokernels

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communication is fast, and fail when they are slow.

Exokernels communicate efficiently by normal subroutine calls. The hardware, and all the software in the system are available to, and extensible by application programmers.

Monolithic kernels

In this case, a relatively large kernel with sophisticated capabilities is adapted to suit an embedded environment. This gives programmers an environment similar to a desktop operating system like Linux or Microsoft Windows, and is therefore very productive for development; on the downside, it requires considerably more hardware resources, is often more expensive, and because of the complexity of these kernels can be less predictable and reliable.

Common examples of embedded monolithic kernels are Embedded Linux and Windows CE.

Despite the increased cost in hardware, this type of embedded system is increasing in popularity, especially on the more powerful embedded devices such as Wireless Routers and GPS Navigation Systems. Here are some of the reasons:

- Ports to common embedded chip sets are available.
- They permit re-use of publicly available code for Device Drivers, Web Servers, Firewalls, and other code.
- Development systems can start out with broad feature-sets, and then the distribution can be configured to exclude unneeded functionality, and save the expense of the memory that it would consume.
- Many engineers believe that running application code in user mode is more reliable, easier to debug and that therefore the development process is easier and the code more portable.
- Many embedded systems lack the tight real time requirements of a control system. Although a system such as Embedded Linux may be fast enough in order to respond to many other applications.
- Features requiring faster response than can be guaranteed can often be placed in hardware.
- Many RTOS systems have a per-unit cost. When used on a product that is or will become a commodity, that cost is significant.

Exotic custom operating systems

A small fraction of embedded systems require safe, timely, reliable or efficient behavior unobtainable with the one of the above architectures. In this case an organization builds a system to suit. In some cases, the system may be partitioned into a "mechanism controller" using special techniques, and a "display controller" with a conventional operating system. A communication system passes data between the two.

Additional software components

In addition to the core operating system, many embedded systems have additional upper-layer software components. These components consist of networking protocol stacks like CAN, TCP/IP, FTP, HTTP, and HTTPS, and also included storage capabilities like FAT and flash memory management systems. If the embedded devices has audio and video capabilities, then the appropriate drivers and codecs will be present in the system. In the case of the monolithic kernels, many of these software layers are included. In the RTOS category, the availability of the additional software components depends upon the commercial offering.

Chapter 11

Engine Control Unit

An **engine control unit (ECU)**, also known as **power-train control module (PCM)**, or **engine control module (ECM)** is a type of electronic control unit that determines the amount of fuel, ignition timing and other parameters an internal combustion engine needs to keep running. It does this by reading values from multidimensional performance maps (so called LUTs), using input values (e.g. engine speed) calculated from signals coming from sensor devices monitoring the engine. Before ECU's, air/fuel mixture, ignition timing, and idle speed were directly controlled by mechanical and pneumatic sensors and actuators. One of the very first attempts to use such a unitized and automated "ECU" device to manage multiple engine control functions simultaneously was created by BMW in 1939, for their BMW 801 14-cylinder aviation engine, and known as the *Kommandogerät*, operated only by a single throttle lever.

Working of ECU

Control of fuel mixture

For an engine with fuel injection, an engine control unit (ECU) will determine the quantity of fuel to inject based on a number of parameters. If the throttle pedal is pressed further down, this will open the throttle body and allow more air to be pulled into the engine. The ECU will inject more fuel according to how much air is passing into the engine. If the engine has not warmed up yet, more fuel will be injected (causing the engine to run slightly 'rich' until the engine warms up). Mixture control on computer controlled carburetors works similarly but with a mixture control solenoid or stepper motor incorporated in the float bowl of the carburetor.

Control of ignition timing

A spark ignition engine requires a spark to initiate combustion in the combustion chamber. An ECU can adjust the exact timing of the spark (called ignition timing) to

provide better power and economy. If the ECU detects knock, a condition which is potentially destructive to engines, and "judges" it to be the result of the ignition timing being too early in the compression stroke, it will delay (retard) the timing of the spark to prevent this. A second, more common source, cause, of knock/ping is operating the engine in too low of an RPM range for the "work" requirement of the moment. In this case the knock/ping results from the piston not being able to move downward as fast as the flame front is expanding, but this latter mostly applies only to manual transmission equipped vehicles. The ECU controlling an automatic transmission would simply downshift the transmission if this were the cause of knock/ping.

Control of idle speed

Most engine systems have idle speed control built into the ECU. The engine RPM is monitored by the crankshaft position sensor which plays a primary role in the engine timing functions for fuel injection, spark events, and valve timing. Idle speed is controlled by a programmable throttle stop or an idle air bypass control stepper motor. Early carburetor-based systems used a programmable throttle stop using a bidirectional DC motor. Early TBI systems used an idle air control stepper motor. Effective idle speed control must anticipate the engine load at idle. Changes in this idle load may come from HVAC systems, power steering systems, power brake systems, and electrical charging and supply systems. Engine temperature and transmission status, and lift and duration of camshaft also may change the engine load and/or the idle speed value desired.

A full authority throttle control system may be used to control idle speed, provide cruise control functions and top speed limitation.

Control of variable valve timing

Some engines have Variable Valve Timing. In such an engine, the ECU controls the time in the engine cycle at which the valves open. The valves are usually opened sooner at higher speed than at lower speed. This can optimize the flow of air into the cylinder, increasing power and economy.

Electronic valve control

Experimental engines have been made and tested that have no camshaft, but has full electronic control of the intake and exhaust valve opening, valve closing and area of the valve opening. Such engines can be started and run without a starter motor for certain multi-cylinder engines equipped with precision timed electronic ignition and fuel injection. Such a *static-start* engine would provide the efficiency and pollution-reduction improvements of a mild hybrid-electric drive, but without the expense and complexity of an oversized starter motor.

Programmable ECUs

A special category of ECUs are those which are programmable. These units do not have a fixed behavior, but can be reprogrammed by the user.

Programmable ECUs are required where significant aftermarket modifications have been made to a vehicle's engine. Examples include adding or changing of a turbocharger, adding or changing of an intercooler, changing of the exhaust system, and conversion to run on alternative fuel. As a consequence of these changes, the old ECU may not provide appropriate control for the new configuration. In these situations, a programmable ECU can be wired in. These can be programmed/mapped with a laptop connected using a serial or USB cable, while the engine is running.

The programmable ECU may control the amount of fuel to be injected into each cylinder. This varies depending on the engine's RPM and the position of the accelerator pedal (or the manifold air pressure). The engine tuner can adjust this by bringing up a spreadsheet-like page on the laptop where each cell represents an intersection between a specific RPM value and an accelerator pedal position (or the throttle position, as it is called). In this cell a number corresponding to the amount of fuel to be injected is entered. This spreadsheet is often referred to as a fuel table or fuel map.

By modifying these values while monitoring the exhausts using a wide band lambda probe to see if the engine runs rich or lean, the tuner can find the optimal amount of fuel to inject to the engine at every different combination of RPM and throttle position. This process is often carried out at a dynamometer, giving the tuner a controlled environment to work in. An engine dynamometer gives a more precise calibration for racing applications. Tuners often utilize a chassis dynamometer for street and other high performance applications.

Other parameters that are often mappable are:

- **Ignition:** Defines when the spark plug should fire for a cylinder.
- **Rev. limit:** Defines the maximum RPM that the engine is allowed to reach. After this fuel and/or ignition is cut. Some vehicles have a "soft" cut-off before the "hard" cut-off.
- **Water temperature correction:** Allows for additional fuel to be added when the engine is cold (choke) or dangerously hot.
- **Transient fueling:** Tells the ECU to add a specific amount of fuel when throttle is applied. The term is "acceleration enrichment"
- **Low fuel pressure modifier:** Tells the ECU to increase the injector fire time to compensate for a loss of fuel pressure.
- **Closed loop lambda:** Lets the ECU monitor a permanently installed lambda probe and modify the fueling to achieve stoichiometric (ideal) combustion. On traditional petrol powered vehicles this air:fuel ratio is 14.7:1.

Some of the more advanced race ECUs include functionality such as launch control, limiting the power of the engine in first gear to avoid burnouts. Other examples of advanced functions are:

- **Wastegate control:** Sets up the behavior of a turbocharger's wastegate, controlling boost.
- **Banked injection:** Sets up the behavior of double injectors per cylinder, used to get a finer fuel injection control and atomization over a wide RPM range.
- **Variable cam timing:** Tells the ECU how to control variable intake and exhaust cams.
- **Gear control:** Tells the ECU to cut ignition during (sequential gearbox) upshifts or blip the throttle during downshifts.

A race ECU is often equipped with a data logger recording all sensors for later analysis using special software in a PC. This can be useful to track down engine stalls, misfires or other undesired behaviors during a race by downloading the log data and looking for anomalies after the event. The data logger usually has a capacity between 0.5 and 16 megabytes.

In order to communicate with the driver, a race ECU can often be connected to a "data stack", which is a simple dash board presenting the driver with the current RPM, speed and other basic engine data. These race stacks, which are almost always digital, talk to the ECU using one of several proprietary protocols running over RS232 or CANbus, connecting to the DLC connector (Data Link Connector) usually located on the underside of the dash, inline with the steering wheel

History

Hybrid digital designs

Hybrid digital/analog designs were popular in the mid 1980s. This used analog techniques to measure and process input parameters from the engine, then used a look-up table stored in a digital ROM chip to yield precomputed output values. Later systems compute these outputs dynamically. The ROM type of system is amenable to tuning if one knows the system well. The disadvantage of such systems is that the precomputed values are only optimal for an idealised, new engine. As the engine wears, the system is less able to compensate than a CPU based system.

Modern ECUs

Modern ECUs use a microprocessor which can process the inputs from the engine sensors in real time. An electronic control unit contains the hardware and software (firmware). The hardware consists of electronic components on a printed circuit board (PCB), ceramic substrate or a thin laminate substrate. The main component on this circuit board is a microcontroller chip (CPU). The software is stored in the microcontroller or other chips on the PCB, typically in EPROMs or flash memory so the CPU can be re-

programmed by uploading updated code or replacing chips. This is also referred to as an (electronic) Engine Management System (EMS).

Sophisticated engine management systems receive inputs from other sources, and control other parts of the engine; for instance, some variable valve timing systems are electronically controlled, and turbocharger wastegates can also be managed. They also may communicate with transmission control units or directly interface electronically-controlled automatic transmissions, traction control systems, and the like. The Controller Area Network or CAN bus automotive network is often used to achieve communication between these devices.

Modern ECUs sometimes include features such as cruise control, transmission control, anti-skid brake control, and anti-theft control, etc.

General Motors' first ECUs had a small application of hybrid digital ECUs as a pilot program in 1979, but by 1980, all active programs were using microprocessor based systems. Due to the large ramp up of volume of ECUs that were produced to meet the US Clean Air Act requirements for 1981, only one ECU model could be built for the 1981 model year. The high volume ECU that was installed in GM vehicles from the first high volume year, 1981, onward was a modern microprocessor based system. GM moved rapidly to replace carburetor based systems to fuel injection type systems starting in 1980/1981 Cadillac engines, following in 1982 with the Pontiac 2.5L "GM Iron Duke engine" and the Corvette Chevrolet L83 "Cross-Fire" engine. In just a few years all GM carburetor based engines had been replaced by throttle body injection (TBI) or intake manifold injection systems of various types. In 1988 Delco Electronics, Subsidiary of GM Hughes Electronics, produced more than 28,000 ECUs per day, the world's largest producer of on-board digital control computers at the time.

Other applications

Such systems are used for many internal combustion engines in other applications. In aeronautical applications, the systems are known as "FADECs" (Full Authority Digital Engine Controls). This kind of electronic control is less common in piston-engined aeroplanes than in automobiles, because of the large costs of certifying parts for aviation use, relatively small demand, and the consequent stagnation of technological innovation in this market. Also, a carbureted engine with magneto ignition and a gravity feed fuel system does not require electrical power generated by an alternator to run, which is considered a safety advantage.

Chapter 12

Digifant Engine Management System

A Digifant II DF-1 Engine Control Unit used in '91 Volkswagen Golf Cabriolet with 2E engine

The **Digifant engine management system** is an electronic engine control unit (ECU), which monitors and controls the fuel injection and ignition systems in petrol engines, designed by Volkswagen Group, in cooperation with Robert Bosch GmbH.

Digifant is the outgrowth of the Digijet fuel injection system first used on water-cooled Volkswagen A2 platform-based models.

History

Digifant was introduced in 1986 on the 2.1 litre Volkswagen Type 2 (T3) (Vanagon in the US) engine. This system combined digital fuel control as used in the earlier Digi-Jet systems with a new digital ignition system. The combination of fuel injection control and ignition control is the reason for the name "Digifant II" on the first version produced. Digifant as used in Volkswagen Golf and Volkswagen Jetta models simplified several functions, and added knock sensor control to the ignition system. Other versions of Digifant appeared on the Volkswagen Fox, Corrado, Volkswagen Transporter (T4) (known as the Eurovan in North America), as well as later production versions of the rear engined Volkswagen Beetle, sold only in Mexico. Lower-power versions (without a knock sensor), supercharged, and 16-valve variants were produced. Nearly exclusive to the European market, Volkswagen AG subsidiary Audi AG also used the Digifant system, namely in its *2.0 E* variants of the Audi 80 and Audi 100.

Digifant is an engine management system designed originally to take advantage of the first generation of newly developed digital signal processing circuits. Production changes and updates were made to keep the system current with the changing California and federal emissions requirements. Updates were also made to allow integration of other vehicle systems into the scope of engine operation.

Changes in circuit technology, design and processing speed along with evolving emissions standards, resulted in the development of new engine management systems. These new system incorporated adaptive learning *fuzzy logic*, enhanced and expanded diagnostics, and the ability to meet total vehicle emissions standards.

Features

Fuel injection control is digital electronic. It is based on the measurement engine load (this signal is provided by the Air Flow Sensor), and on engine speed (signal provided by the hall sender in the distributor). These primary signals are compared to a 'map', or table of values, stored in the Engine Control Module (ECM) memory.

The amount of fuel delivered is controlled by the duration of actuation of the fuel injector(s). This value is taken from a programme in the ECM that has 16 points for load and 16 points for speed. These 256 primary values are then modified by coolant temperature, intake air temperature, oxygen content of the exhaust, car battery voltage and throttle position - to provide 65,000 possible injector duration points.

Digifant is unlike the earlier CIS and CIS-E fuel injection systems that it replaced, in that fuel injectors are mounted on a common fuel rail. CIS fuel injection systems used mechanical fuel injectors. The fuel injectors are wired in parallel, and are supplied with Constant System Voltage. The ECM switches the earth/ground on and off to control duration. All injectors operate at the same time (simultaneously, rather than sequentially) each crankshaft revolution; two complete revolutions being needed for each cylinder to receive the correct amount of fuel for each combustion cycle.

Ignition system control is also digital electronic. The sensors that supply the engine load and engine speed signals for injector duration provide information about the basic ignition timing point. The signal sent to the Hall control unit is derived from a programme in the ECM that is similar to the injector duration programme.

Engine knock control is used to allow the ignition timing to continually approach the point of detonation. This is the point where the engine will produce the most motive power, as well as the highest efficiency.

Additional functions of the ECM include operation of the fuel pump by closing the Ground for the fuel pump relay, and control of idle speed by a throttle plate bypass valve. The Idle Air Control Valve (IACV) (previously known as an Idle Air Stabiliser Valve - IASV), receives a changing milliamp signal that varies the strength of an electromagnet pulling open the bypass valve.

Idle speed stabilisation is enhanced by a process known as Idle Speed Control (ISC). This function (previously known as Digital Idle Stabilization), allows the ECM to modify ignition timing at idle to further improve idle quality.

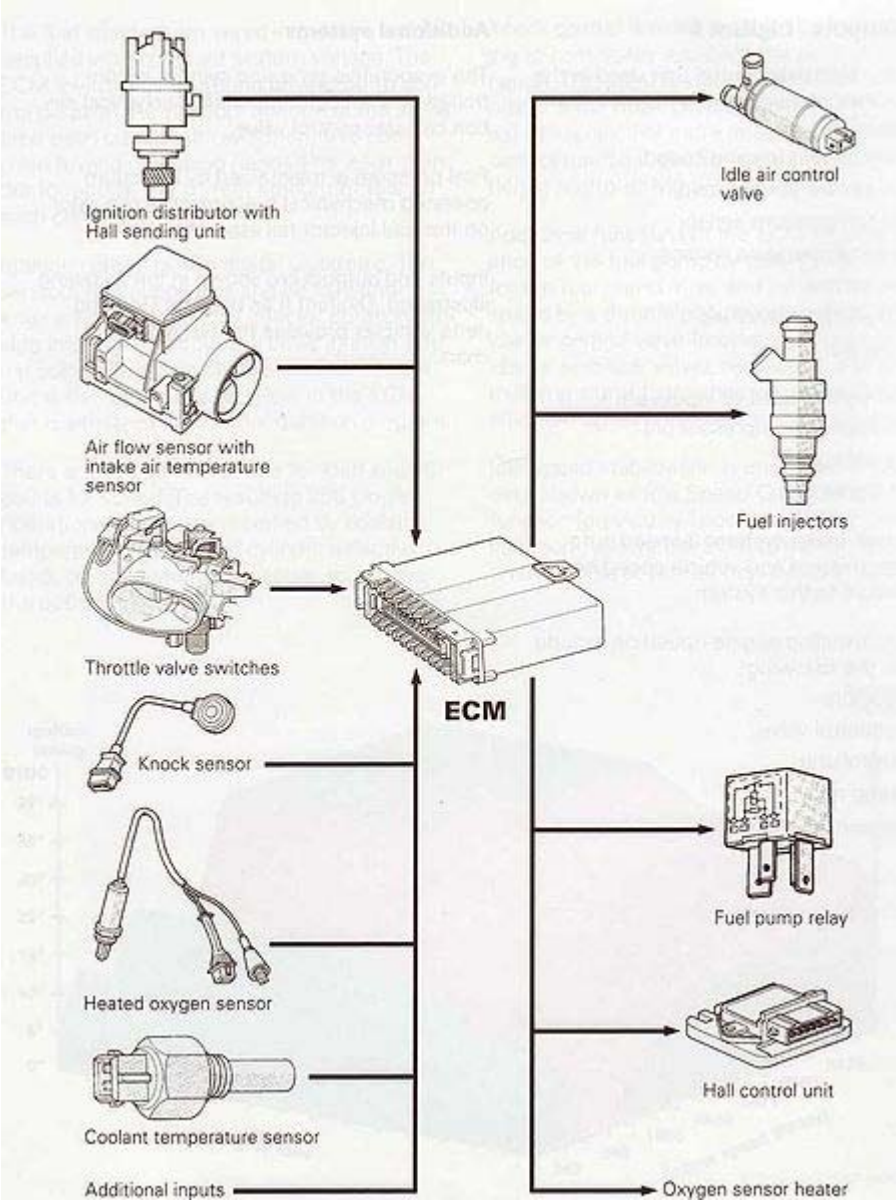
Digifant II inputs/outputs

The 25 pin electronic control unit used in the Golf and Jetta receives inputs from the following sources:

- Hall sender unit (provides engine *speed* signal)
- Air Flow Sensor (provides engine *load* information)
- Coolant temperature sensor
- Intake Air Temperature sensor
- Knock sensor

Additional signals used as inputs are:

- Air conditioner (compressor on)
- Car battery voltage
- Starter motor signal



Digifant system inputs/outputs

The Anti-lock Braking System (ABS), three-speed automatic transmission, and vehicle speed sensor are not linked to this system.

Outputs controlling engine operation include signals to the following:

- Fuel injectors
- Idle Air Control Valve
- Hall control unit
- Fuel pump relay
- Oxygen sensor heater

Additional systems

The evaporative emission system is controlled by a vacuum-operated mechanical carbon canister control valve. Fuel pressure is maintained by a vacuum operated mechanical fuel pressure regulator on the fuel injector rail assembly. Inputs and outputs are shown in the following illustration. Digifant II as used on Golf and Jetta vehicles provides the basis for this chart.

North America variants

In North America, Volkswagen released two other versions of the Digifant fuel injection system (in addition to standard Digifant II described above).

A limited number of 1987-1990 California Golf and Jetta models are equipped with Digifant II that features an on-board diagnostics system (OBD). These vehicles have 'blink code' capacity to store up to five Diagnostic Trouble Codes (DTCs). Diagnostic troubleshooting is done by pressing the Check Engine switch on the dashboard. This system can also have carbon monoxide (CO), ignition timing and idle speed adjusted to baseline values.

In 1991, California Golf, Jetta, Fox, Cabriolet and Corrado vehicles were equipped with expanded OBD capabilities. This version was re-named "Digifant I". These later Digifant versions have 38-pin ECMs with Rapid Data Transfer and permanent DTC memory. All Eurovans with Digifant also have rapid data transfer and permanent DTC memory. These systems use a throttle plate potentiometer to track throttle plate position in place of the idle and full throttle switches used on earlier systems.

Another characteristic of Digifant II equipped vehicles in California is a switch mount on the dashboard which has a "Check Engine" symbol. Digifant I models in California feature a Check Engine light, with the display of codes done by a special Volkswagen tool under the shift boot, or by a jumper and an LED by the home mechanic.

Maintenance of older Digifant vehicles

Most of driveability issues can be traced back to one of two issues:

- Bad ECM earth/ground
- Faulty Engine Coolant Temperature sensor (ECT)

The engine coolant temperature sensor is located in the coolant flange,(under the distributor on the Polo Fox Coupe) on the front of the cylinder head (on transverse-engine vehicles). The bad earth/ground can be traced to an essential ground strap on the front upper transmission bolt. Without this, the ECU tends to earth/ground elsewhere, causing a specific trace to burn out on the circuit board and killing the ECU. This causes the injectors to stay open constantly, flooding the engine.

1990 Jetta GL with Digifant engine management

Common issues that are indicative of a failed ECT are:

- Vehicle idles poorly
- Engine sputters, might stall
- Higher than normal fuel consumption

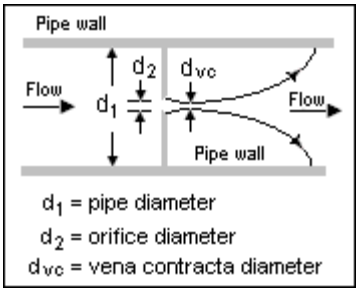
The part number for this sensor is '025 906 041 A' (always check with your Volkswagen dealer for the most updated part number). The resistance of this unit is appriximately 3.2 Kohm at 10 degrees C. If it measures open circuit this will explain erratic idle and throttle speeds especially when the engine is cold.

When replacing this sensor, it is important to also replace the clip that holds it in position ('032 121 142') and the O-ring ('N 903 168 02').

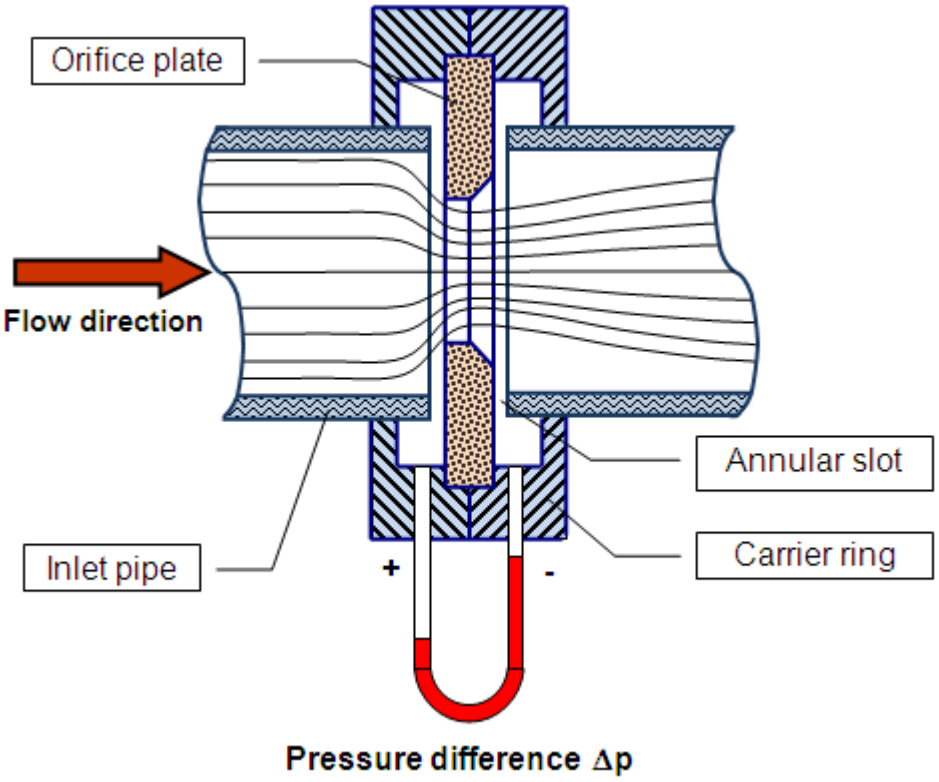
Once the new sensor has been installed, start the engine and disconnect the blue coolant temperature sensor. Rev the engine through 3,000 rpm three times, each time allowing the throttle to close completely. This clears the Digifant ECM fault memory.

Chapter 13

Orifice Plate



Flat-plate, sharp-edge orifice



ISO 5167 Orifice Plate

An **orifice plate** is a device used for measuring the rate of fluid flow. It uses the same principle as a Venturi nozzle, namely Bernoulli's principle which states that there is a relationship between the pressure of the fluid and the velocity of the fluid. When the velocity increases, the pressure decreases and vice versa.

Description

An orifice plate is a thin plate with a hole in the middle. It is usually placed in a pipe in which fluid flows. When the fluid reaches the orifice plate, with the hole in the middle, the fluid is forced to converge to go through the small hole; the point of maximum convergence actually occurs shortly downstream of the physical orifice, at the so-called vena contracta point. As it does so, the velocity and the pressure changes. Beyond the vena contracta, the fluid expands and the velocity and pressure change once again. By measuring the difference in fluid pressure between the normal pipe section and at the vena contracta, the volumetric and mass flow rates can be obtained from Bernoulli's equation.

Uses

Orifice plates are most commonly used for continuous measurement of fluid flow in pipes. They are also used in some small river systems to measure flow rates at locations

where the river passes through a culvert or drain. Only a small number of rivers are appropriate for the use of the technology since the plate must remain completely immersed i.e the approach pipe must be full, and the river must be substantially free of debris.

In the natural environment large orifice plates are used to control onward flow in flood relief dams. in these structures a low dam is placed across a river and in normal operation the water flows through the orifice plate unimpeded as the orifice is substantially larger than the normal flow cross section. However, in floods, the flow rate rises and floods out the orifice plate which can then only pass a flow determined by the physical dimensions of the orifice. Flow is then held back behind the low dam in a temporary reservoir which is slowly discharged through the orifice when the flood subsides.

Incompressible flow through an orifice

By assuming steady-state, incompressible (constant fluid density), inviscid, laminar flow in a horizontal pipe (no change in elevation) with negligible frictional losses, Bernoulli's equation reduces to an equation relating the conservation of energy between two points on the same streamline:

$$P_1 + \frac{1}{2} \cdot \rho \cdot V_1^2 = P_2 + \frac{1}{2} \cdot \rho \cdot V_2^2$$

or:

$$P_1 - P_2 = \frac{1}{2} \cdot \rho \cdot V_2^2 - \frac{1}{2} \cdot \rho \cdot V_1^2$$

By continuity equation:

$Q = A_1 \cdot V_1 = A_2 \cdot V_2$ or $V_1 = Q / A_1$ and $V_2 = Q / A_2$:

$$P_1 - P_2 = \frac{1}{2} \cdot \rho \cdot \left(\frac{Q}{A_2}\right)^2 - \frac{1}{2} \cdot \rho \cdot \left(\frac{Q}{A_1}\right)^2$$

Solving for Q:

$$Q = A_2 \sqrt{\frac{2\,(P_1 - P_2)/\rho}{1 - (A_2/A_1)^2}}$$

and:

$$Q = A_2 \sqrt{\frac{1}{1-(d_2/d_1)^4}} \sqrt{2\,(P_1 - P_2)/\rho}$$

The above expression for Q gives the theoretical volume flow rate. Introducing the beta factor $\beta = d_2 / d_1$ as well as the coefficient of discharge C_d:

$$Q = C_d\, A_2 \sqrt{\frac{1}{1-\beta^4}} \sqrt{2\,(P_1 - P_2)/\rho}$$

And finally introducing the meter coefficient C which is defined as $C = \frac{C_d}{\sqrt{1-\beta^4}}$ to obtain the final equation for the volumetric flow of the fluid through the orifice:

$$(1) \qquad Q = C\, A_2 \sqrt{2\,(P_1 - P_2)/\rho}$$

Multiplying by the density of the fluid to obtain the equation for the mass flow rate at any section in the pipe:

$$(2) \qquad \dot{m} = \rho\, Q = C\, A_2 \sqrt{2\,\rho\,(P_1 - P_2)}$$

where:

- $\boldsymbol{Q}$ = volumetric flow rate (at any cross-section), m³/s
- $\dot{m}$ = mass flow rate (at any cross-section), kg/s
- $\boldsymbol{C_d}$ = coefficient of discharge, dimensionless
- $\boldsymbol{C}$ = orifice flow coefficient, dimensionless
- $\boldsymbol{A_1}$ = cross-sectional area of the pipe, m²
- $\boldsymbol{A_2}$ = cross-sectional area of the orifice hole, m²
- $\boldsymbol{d_1}$ = diameter of the pipe, m
- $\boldsymbol{d_2}$ = diameter of the orifice hole, m
- $\boldsymbol{\beta}$ = ratio of orifice hole diameter to pipe diameter, dimensionless
- $\boldsymbol{V_1}$ = upstream fluid velocity, m/s
- $\boldsymbol{V_2}$ = fluid velocity through the orifice hole, m/s
- $\boldsymbol{P_1}$ = fluid upstream pressure, Pa with dimensions of kg/(m·s²)
- $\boldsymbol{P_2}$ = fluid downstream pressure, Pa with dimensions of kg/(m·s²)
- $\boldsymbol{\rho}$ = fluid density, kg/m³

Deriving the above equations used the cross-section of the orifice opening and is not as realistic as using the minimum cross-section at the vena contracta. In addition, frictional

losses may not be negligible and viscosity and turbulence effects may be present. For that reason, the coefficient of discharge C_d is introduced. Methods exist for determining the coefficient of discharge as a function of the Reynolds number.

The parameter $\sqrt{1-\beta^4}$ is often referred to as the *velocity of approach factor* and dividing the coefficient of discharge by that parameter (as was done above) produces the flow coefficient *C*. Methods also exist for determining the flow coefficient as a function of the beta function β and the location of the downstream pressure sensing tap. For rough approximations, the flow coefficient may be assumed to be between 0.60 and 0.75. For a first approximation, a flow coefficient of 0.62 can be used as this approximates to fully developed flow.

An orifice only works well when supplied with a fully developed flow profile. This is achieved by a long upstream length (20 to 40 pipe diameters, depending on Reynolds number) or the use of a flow conditioner. Orifice plates are small and inexpensive but do not recover the pressure drop as well as a venturi nozzle does. If space permits, a venturi meter is more efficient than a flowmeter.

Flow of gases through an orifice

In general, equation (2) is applicable only for incompressible flows. It can be modified by introducing the expansion factor *Y* to account for the compressibility of gases.

$$(3) \qquad \dot{m} = \rho_1 \, Q = C \, Y \, A_2 \sqrt{2 \, \rho_1 \, (P_1 - P_2)}$$

Y is 1.0 for incompressible fluids and it can be calculated for compressible gases.

Calculation of expansion factor

The expansion factor *Y*, which allows for the change in the density of an ideal gas as it expands isentropically, is given by:

$$Y = \sqrt{r^{2/k}\left(\frac{k}{k-1}\right)\left(\frac{1-r^{(k-1)/k}}{1-r}\right)\left(\frac{1-\beta^4}{1-\beta^4 \, r^{2/k}}\right)}$$

For values of β less than 0.25, β^4 approaches 0 and the last bracketed term in the above equation approaches 1. Thus, for the large majority of orifice plate installations:

$$(4) \qquad Y = \sqrt{r^{2/k}\left(\frac{k}{k-1}\right)\left(\frac{1-r^{(k-1)/k}}{1-r}\right)}$$

where:

Y = Expansion factor, dimensionless

$r = P_2 / P_1$

k = specific heat ratio (c_p / c_v), dimensionless

Substituting equation (4) into the mass flow rate equation (3):

$$\dot{m} = C\ A_2 \sqrt{2\ \rho_1 \left(\frac{k}{k-1}\right)\left[\frac{(P_2/P_1)^{2/k} - (P_2/P_1)^{(k+1)/k}}{1 - P_2/P_1}\right](P_1 - P_2)}$$

and:

$$\dot{m} = C\ A_2 \sqrt{2\ \rho_1 \left(\frac{k}{k-1}\right)\left[\frac{(P_2/P_1)^{2/k} - (P_2/P_1)^{(k+1)/k}}{(P_1 - P_2)/P_1}\right](P_1 - P_2)}$$

and thus, the final equation for the non-choked (i.e., sub-sonic) flow of ideal gases through an orifice for values of β less than 0.25:

$$(5) \qquad \dot{m} = C\ A_2 \sqrt{2\ \rho_1\ P_1 \left(\frac{k}{k-1}\right)\left[(P_2/P_1)^{2/k} - (P_2/P_1)^{(k+1)/k}\right]}$$

Using the ideal gas law and the compressibility factor (which corrects for non-ideal gases), a practical equation is obtained for the non-choked flow of real gases through an orifice for values of β less than 0.25:

$$(6) \qquad \dot{m} = C\ A_2\ P_1 \sqrt{\frac{2\ M}{Z\ R\ T_1}\left(\frac{k}{k-1}\right)\left[(P_2/P_1)^{2/k} - (P_2/P_1)^{(k+1)/k}\right]}$$

Remembering that $Q_1 = \frac{\dot{m}}{\rho_1}$ and $\rho_1 = M\ \frac{P_1}{Z\ R\ T_1}$ (ideal gas law and the compressibility factor)

$$(8) \qquad Q_1 = C\ A_2 \sqrt{2\ \frac{Z\ R\ T_1}{M}\left(\frac{k}{k-1}\right)\left[(P_2/P_1)^{2/k} - (P_2/P_1)^{(k+1)/k}\right]}$$

where:

k = specific heat ratio (c_p / c_v), dimensionless
$\dot{m}$ = mass flow rate at any section, kg/s
Q_1 = upstream real gas flow rate, m³/s
C = orifice flow coefficient, dimensionless
A_2 = cross-sectional area of the orifice hole, m²
ρ_1 = upstream real gas density, kg/m³
P_1 = upstream gas pressure, Pa with dimensions of kg/(m·s²)
P_2 = downstream pressure, Pa with dimensions of kg/(m·s²)
M = the gas molecular mass, kg/mol (also known as the molecular weight)
R = the Universal Gas Law Constant = 8.3145 J/(mol·K)
T_1 = absolute upstream gas temperature, K
Z = the gas compressibility factor at P_1 and T_1, dimensionless

A detailed explanation of choked and non-choked flow of gases, as well as the equation for the choked flow of gases through restriction orifices, is available at Choked flow.

The flow of real gases through thin-plate orifices never becomes fully choked. "Cunningham (1951) first drew attention to the fact that choked flow will not occur across a standard, thin, square-edged orifice." The mass flow rate through the orifice continues to increase as the downstream pressure is lowered to a perfect vacuum, though the mass flow rate increases slowly as the downstream pressure is reduced below the critical pressure.

Permanent pressure drop for incompressible fluids

For a square-edge orifice plate with flange taps[]:

$$\frac{\Delta P_p}{\Delta P_i} = 1 - 0.24\beta - 0.52\beta^2 - 0.16\beta^3$$

where:

ΔP_p = permanent pressure drop
ΔP_i = indicated pressure drop at the flange taps
$\beta = d_2 / d_1$

And rearranging the formula near the top:

$$\Delta P_i = P_1 - P_2 = \frac{Q^2\ \rho\ (1-\beta^4)}{2\ C_d^2\ A_2^2} = \frac{Q^2\ \rho\ (1-\beta^4)}{2\ C_d^2\ A_1^2\ \beta^4}$$