Advanced Control Engineering

Malvina Israel

First Edition, 2012

ISBN 978-81-323-4476-6

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

Table of Contents

Chapter- 1

Control Engineering

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

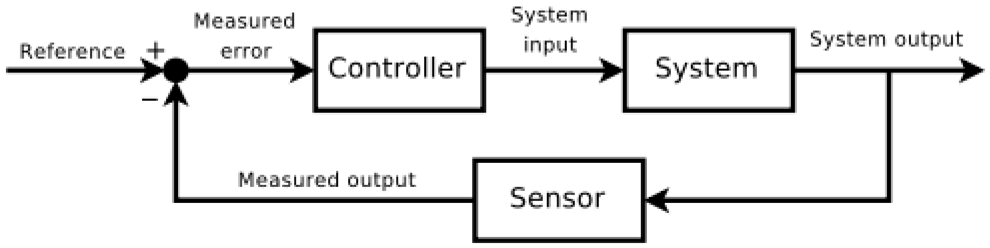
Originally, control engineering was all about continuous systems. Development of computer control tools posed a requirement of discrete control system engineering because the communications between the computer-based digital controller and the physical system are governed by a computer clock. The equivalent 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- 2

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.

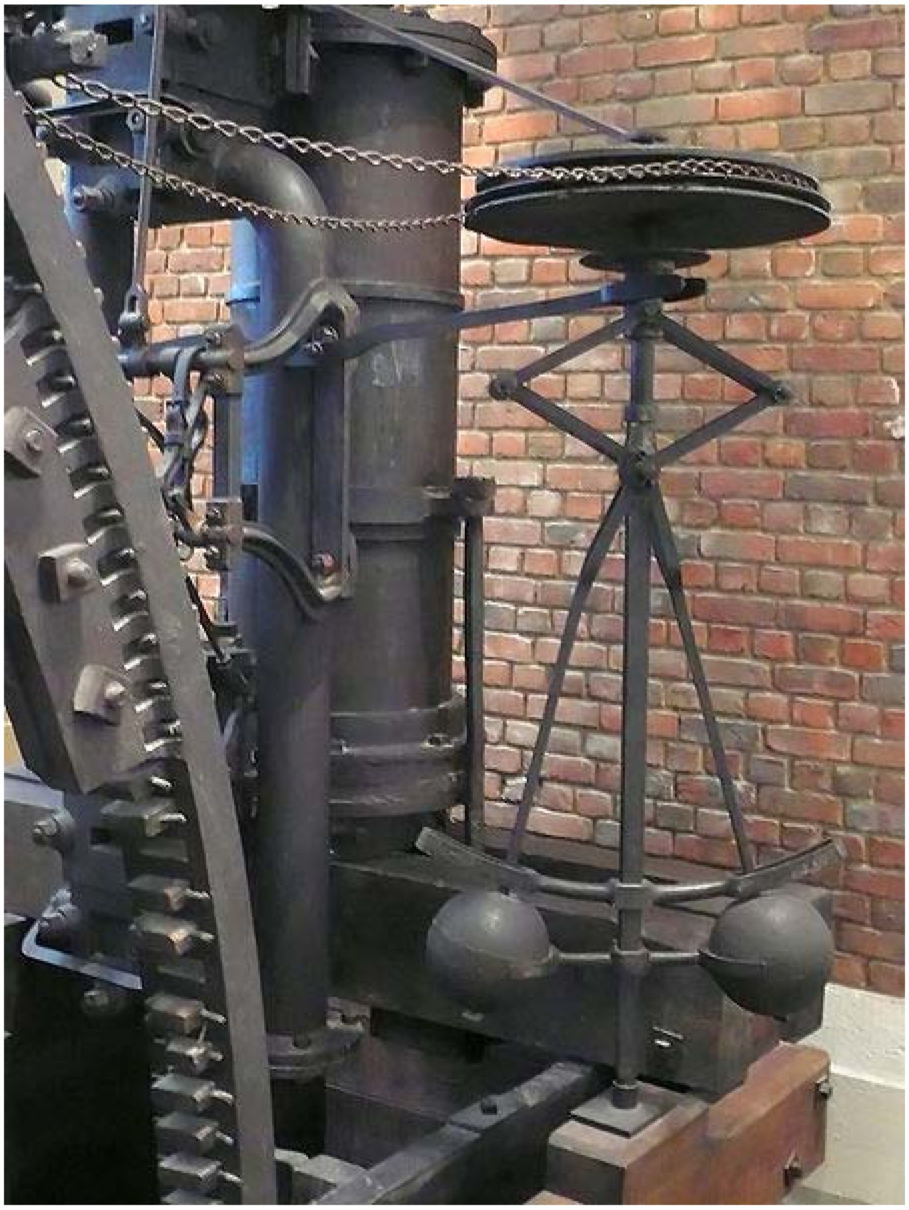
An example

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. 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

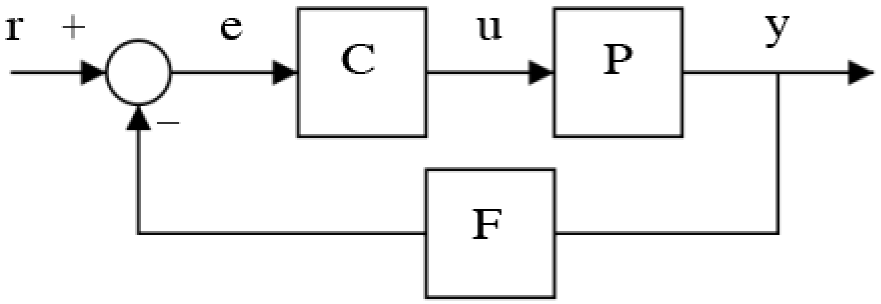
In some systems, closed-loop and open-loop control are used simultaneously. In such systems, the open-loop control is termed feedforward and serves to further improve reference tracking performance.

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.

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. 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, 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

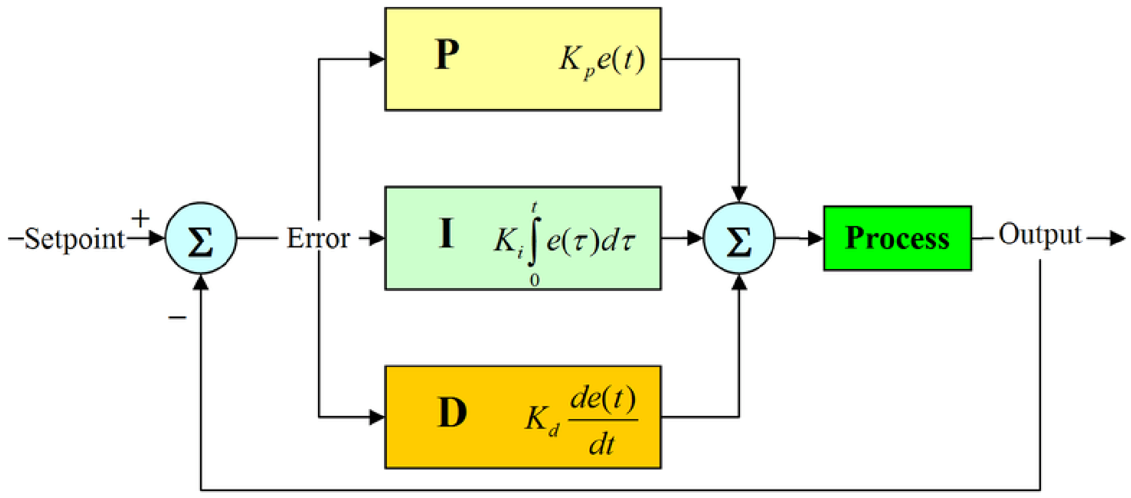
Robust control deals explicitly with uncertainty in its approach to controller design. Controllers designed using *robust control* methods tend to be able to cope with small differences between the true system and the nominal model used for design. The early methods of Bode and others were fairly robust; 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- 3

PID Controller



A block diagram of a PID controller

A **proportional–integral–derivative controller** (**PID controller**) is a generic control loop feedback mechanism (controller) widely used in industrial control systems – a PID is the most commonly used feedback controller. A PID controller calculates an "error" value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the error by adjusting the process control inputs.

The PID controller calculation (algorithm) involves three separate parameters, and is accordingly sometimes called **three-term control**: the proportional, the integral and derivative values, denoted *P, I,* and *D.* Heuristically, these values can be interpreted in terms of time: *P* depends on the *present* error, *I* on the accumulation of *past* errors, and *D* is a prediction of *future* errors, based on current rate of change. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element.

In the absence of knowledge of the underlying process, a PID controller is the best controller. By tuning the three constants in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

Some applications may require using only one or two modes to provide the appropriate system control. This is achieved by setting the gain of undesired control outputs to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are fairly common, since derivative action is sensitive to measurement noise, whereas the absence of an integral value may prevent the system from reaching its target value due to the control action.

Control loop basics

A familiar example of a control loop is the action taken when adjusting hot and cold faucet valves to maintain the faucet water at the desired temperature. This typically involves the mixing of two process streams, the hot and cold water. The person touches the water to sense or measure its temperature. Based on this feedback they perform a control action to adjust the hot and cold water valves until the process temperature stabilizes at the desired value.

Sensing water temperature is analogous to taking a measurement of the process value or process variable (PV). The desired temperature is called the setpoint (SP). The input to the process (the water valve position) is called the manipulated variable (MV). The difference between the temperature measurement and the setpoint is the error (e) and quantifies whether the water is too hot or too cold and by how much.

After measuring the temperature (PV), and then calculating the error, the controller decides when to change the tap position (MV) and by how much. When the controller first turns the valve on, it may turn the hot valve only slightly if warm water is desired, or it may open the valve all the way if very hot water is desired. This is an example of a simple **proportional** control. In the event that hot water does not arrive quickly, the controller may try to speed-up the process by opening up the hot water valve more-and-more as time goes by. This is an example of an **integral** control.

Making a change that is too large when the error is small is equivalent to a high gain controller and will lead to overshoot. If the controller were to repeatedly make changes that were too large and repeatedly overshoot the target, the output would oscillate around the setpoint in either a constant, growing, or decaying sinusoid. If the oscillations increase with time then the system is unstable, whereas if they decrease the system is stable. If the oscillations remain at a constant magnitude the system is marginally stable.

In the interest of achieving a gradual convergence at the desired temperature (SP), the controller may wish to damp the anticipated future oscillations. So in order to compensate for this effect, the controller may elect to temper their adjustments. This can be thought of as a **derivative** control method.

If a controller starts from a stable state at zero error (PV = SP), then further changes by the controller will be in response to changes in other measured or unmeasured inputs to the process that impact on the process, and hence on the PV. Variables that impact on the process other than the MV are known as disturbances. Generally controllers are used to reject disturbances and/or implement setpoint changes. Changes in feedwater temperature constitute a disturbance to the faucet temperature control process.

In theory, a controller can be used to control any process which has a measurable output (PV), a known ideal value for that output (SP) and an input to the process (MV) that will affect the relevant PV. Controllers are used in industry to regulate temperature, pressure, flow rate, chemical composition, speed and practically every other variable for which a measurement exists.

PID controller theory

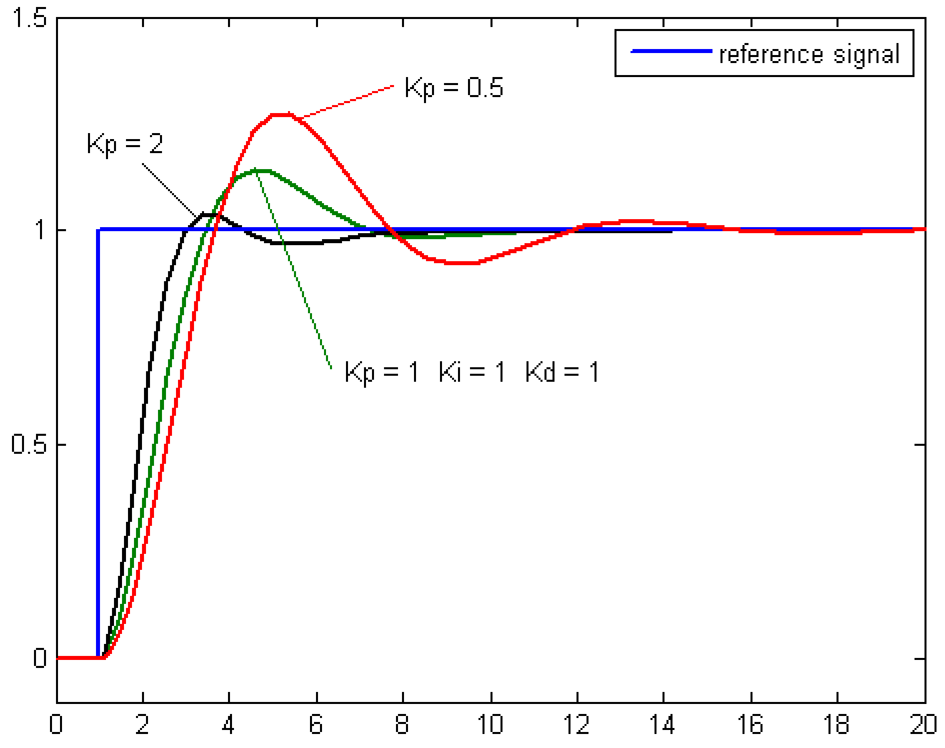
The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). Hence:

$$\mathrm{MV(t)} = P_{\mathrm{out}} + I_{\mathrm{out}} + D_{\mathrm{out}}$$

where

P_{out}, I_{out}, and D_{out} are the contributions to the output from the PID controller from each of the three terms, as defined below.

Proportional term



Plot of PV vs time, for three values of K_p (K_i and K_d held constant)

The proportional term (sometimes called *gain*) makes a change to the output that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain.

The proportional term is given by:

$$P_{\text{out}} = K_p\, e(t)$$

where

P_{out}: Proportional term of output
K_p: Proportional gain, a tuning parameter
SP: Setpoint, the desired value
PV: Process value (or process variable), the measured value
e: Error = $SP - PV$
t: Time or instantaneous time (the present)

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive (or sensitive) controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances.

Droop

A pure proportional controller will not always settle at its target value, but may retain a steady-state error. Specifically, the process gain - drift in the absence of control, such as cooling of a furnace towards room temperature, biases a pure proportional controller. If the process gain is down, as in cooling, then the bias will be *below* the set point, hence the term "droop".

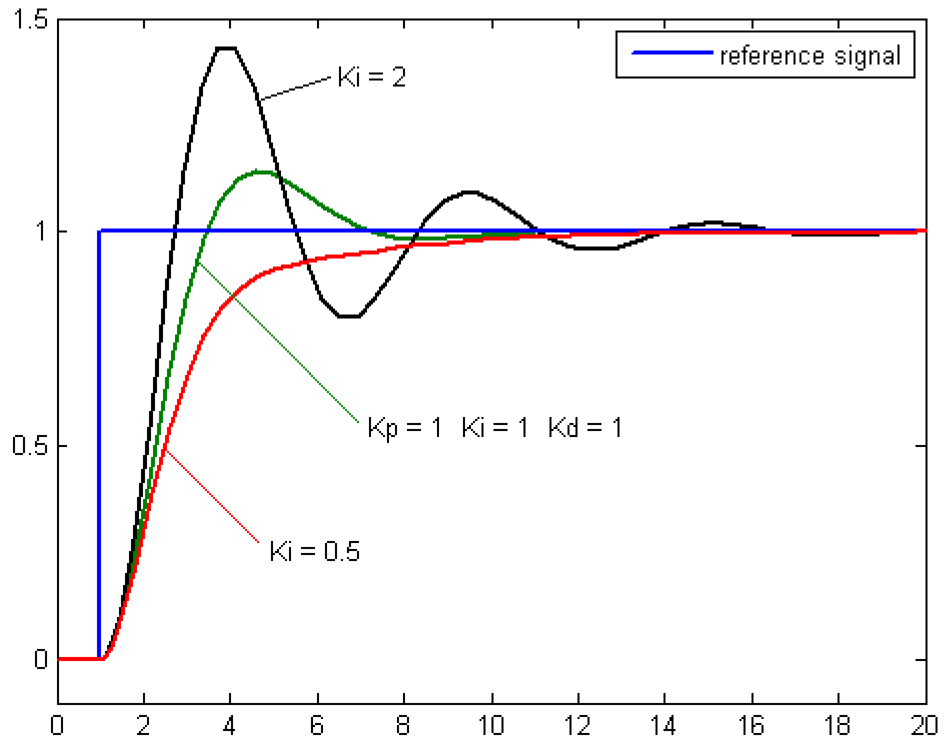
Droop is proportional to process gain and inversely proportional to proportional gain. Specifically the steady-state error is given by:

$$e = G / K_p$$

Droop is an inherent defect of purely proportional control. Droop may be mitigated by adding a compensating *bias* term (setting the setpoint above the true desired value), or corrected by adding an integration term (in a PI or PID controller), which effectively computes a bias adaptively.

Despite droop, both tuning theory and industrial practice indicate that it is the proportional term that should contribute the bulk of the output change.

Integral term



Plot of PV vs time, for three values of K_i (K_p and K_d held constant)

The contribution from the integral term (sometimes called *reset*) is proportional to both the magnitude of the error and the duration of the error. Summing the instantaneous error over time (integrating the error) gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain and added to the controller output. The magnitude of the contribution of the integral term to the overall control action is determined by the integral gain, K_i.

The integral term is given by:

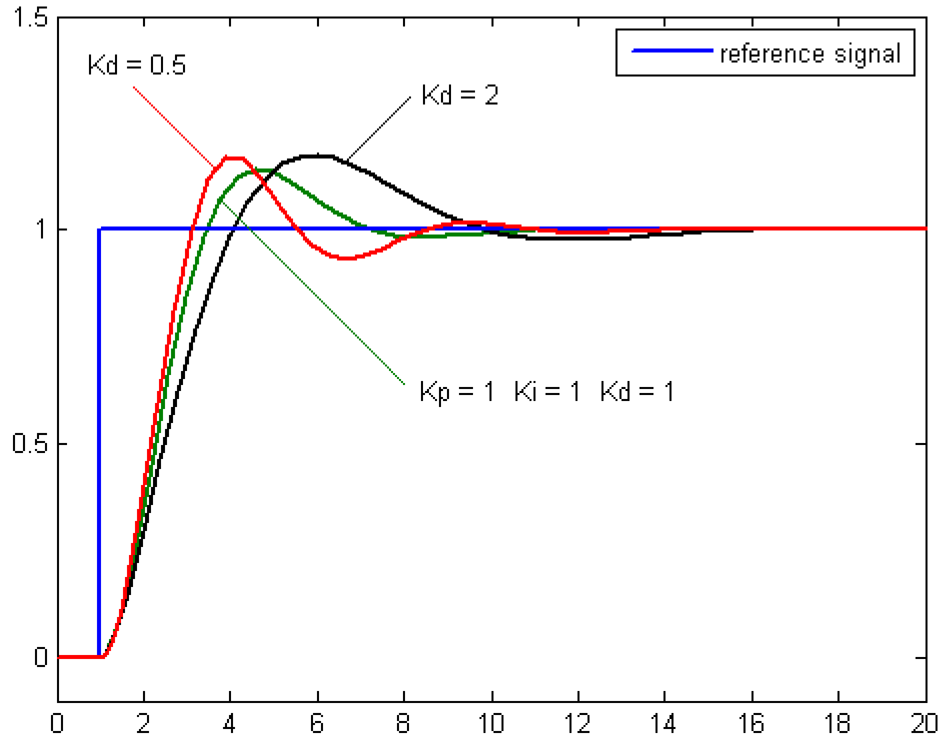
$$I_{\text{out}} = K_i \int_0^t e(\tau)\, d\tau$$

where

I_{out}: Integral term of output
K_i: Integral gain, a tuning parameter
SP: Setpoint, the desired value
PV: Process value (or process variable), the measured value
e: Error = $SP - PV$
t: Time or instantaneous time (the present)
τ: a dummy integration variable

The integral term (when added to the proportional term) accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a proportional only controller. However, since the integral term is responding to accumulated errors from the past, it can cause the present value to overshoot the setpoint value (cross over the setpoint and then create a deviation in the other direction).

Derivative term



Plot of PV vs time, for three values of K_d (K_p and K_i held constant)

The rate of change of the process error is calculated by determining the slope of the error over time (i.e., its first derivative with respect to time) and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term (sometimes called *rate*) to the overall control action is termed the derivative gain, K_d.

The derivative term is given by:

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

where

D_{out}: Derivative term of output
K_d: Derivative gain, a tuning parameter
SP: Setpoint, the desired value
PV: Process value (or process variable), the measured value
e: Error = $SP - PV$

t: Time or instantaneous time (the present)

The derivative term slows the rate of change of the controller output and this effect is most noticeable close to the controller setpoint. Hence, derivative control is used to reduce the magnitude of the overshoot produced by the integral component and improve the combined controller-process stability. However, differentiation of a signal amplifies noise and thus this term in the controller is highly sensitive to noise in the error term, and can cause a process to become unstable if the noise and the derivative gain are sufficiently large. Hence an approximation to a differentiator with a limited bandwidth is more commonly used. Such a circuit is known as a Phase-Lead compensator.

Summary

The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining *u*(*t*) as the controller output, the final form of the PID algorithm is:

$$\mathrm{u(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

where the tuning parameters are:

Proportional gain, K_p
: Larger values typically mean faster response since the larger the error, the larger the proportional term compensation. An excessively large proportional gain will lead to process instability and oscillation.

Integral gain, K_i
: Larger values imply steady state errors are eliminated more quickly. The trade-off is larger overshoot: any negative error integrated during transient response must be integrated away by positive error before reaching steady state.

Derivative gain, K_d
: Larger values decrease overshoot, but slow down transient response and may lead to instability due to signal noise amplification in the differentiation of the error

Loop tuning

Tuning a control loop is the adjustment of its control parameters (gain/proportional band, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. Stability (bounded oscillation) is a basic requirement, but beyond that, different systems have different behavior, different applications have different requirements, and requirements may conflict with one another.

Some processes have a degree of non-linearity and so parameters that work well at full-load conditions don't work when the process is starting up from no-load; this can be corrected by gain scheduling (using different parameters in different operating regions).

PID controllers often provide acceptable control using default tunings, but performance can generally be improved by careful tuning, and performance may be unacceptable with poor tuning.

PID tuning is a difficult problem, even though there are only three parameters and in principle is simple to describe, because it must satisfy complex criteria within the limitations of PID control. There are accordingly various methods for loop tuning, and more sophisticated techniques are the subject of patents; this section describes some traditional manual methods for loop tuning.

Stability

If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable, i.e. its output diverges, with or without oscillation, and is limited only by saturation or mechanical breakage. Instability is caused by *excess* gain, particularly in the presence of significant lag.

Generally, stability of response (the reverse of instability) is required and the process must not oscillate for any combination of process conditions and setpoints, though sometimes marginal stability (bounded oscillation) is acceptable or desired.

Optimum behavior

The optimum behavior on a process change or setpoint change varies depending on the application.

Two basic requirements are *regulation* (disturbance rejection – staying at a given setpoint) and *command tracking* (implementing setpoint changes) – these refer to how well the controlled variable tracks the desired value. Specific criteria for command tracking include rise time and settling time. Some processes must not allow an overshoot of the process variable beyond the setpoint if, for example, this would be unsafe. Other processes must minimize the energy expended in reaching a new setpoint.

Overview of methods

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively inefficient, particularly if the loops have response times on the order of minutes or longer.

The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Choosing a Tuning Method

Method	**Advantages**	**Disadvantages**
Manual Tuning	No math required. Online method.	Requires experienced personnel.
Ziegler–Nichols	Proven Method. Online method.	Process upset, some trial-and-error, very aggressive tuning.
Software Tools	Consistent tuning. Online or offline method. May include valve and sensor analysis. Allow simulation before downloading. Can support Non-Steady State (NSS) Tuning.	Some cost and training involved.
Cohen-Coon	Good process models.	Some math. Offline method. Only good for first-order processes.

Manual tuning

If the system must remain online, one tuning method is to first set K_i and K_d values to zero. Increase the K_p until the output of the loop oscillates, then the K_p should be set to approximately half of that value for a "quarter amplitude decay" type response. Then increase K_i until any offset is correct in sufficient time for the process. However, too much K_i will cause instability. Finally, increase K_d, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much K_d will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an *over-damped* closed-loop system is required, which will require a K_p setting significantly less than half that of the K_p setting causing oscillation.

Effects of *increasing* a parameter independently

Parameter	**Rise time**	**Overshoot**	**Settling time**	**Steady-state error**	**Stability**
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Decrease significantly	Degrade
K_d	Minor decrease	Minor decrease	Minor decrease	No effect in theory	Improve if K_d small

Ziegler–Nichols method

Another heuristic tuning method is formally known as the Ziegler–Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols in the 1940s. As in the method above, the K_i and K_d gains are first set to zero. The P gain is increased until it reaches the ultimate gain, K_u, at which the output of the loop starts to oscillate. K_u and the oscillation period P_u are used to set the gains as shown:

Ziegler–Nichols method

Control Type	K_p	K_i	K_d
P	$0.50K_u$	-	-
PI	$0.45K_u$	$1.2K_p / P_u$	-
PID	$0.60K_u$	$2K_p / P_u$	$K_pP_u / 8$

These gains apply to the ideal, parallel form of the PID controller. When applied to the standard PID form, the integral and derivative time parameters T_i and T_d are only dependent on the oscillation period P_u.

PID tuning software

Most modern industrial facilities no longer tune loops using the manual calculation methods shown above. Instead, PID tuning and loop optimization software are used to ensure consistent results. These software packages will gather the data, develop process models, and suggest optimal tuning. Some software packages can even develop tuning by gathering data from reference changes.

Mathematical PID loop tuning induces an impulse in the system, and then uses the controlled system's frequency response to design the PID loop values. In loops with response times of several minutes, mathematical loop tuning is recommended, because trial and error can literally take days just to find a stable set of loop values. Optimal values are harder to find. Some digital loop controllers offer a self-tuning feature in which very small setpoint changes are sent to the process, allowing the controller itself to calculate optimal tuning values.

Other formulas are available to tune the loop according to different performance criteria. Many patented formulas are now embedded within PID tuning software and hardware modules.

Advances in automated PID Loop Tuning software also deliver algorithms for tuning PID Loops in a dynamic or Non-Steady State (NSS) scenario. The software will model the dynamics of a process, through a disturbance, and calculate PID control parameters in response.

Modifications to the PID algorithm

The basic PID algorithm presents some challenges in control applications that have been addressed by minor modifications to the PID form.

Integral windup

One common problem resulting from the ideal PID implementations is integral windup, where a large change in setpoint occurs (say a positive change) and the integral term accumulates an error larger than the maximal value for the the regulation variable (windup), thus the system overshoots and continues to increase as this accumulated error is unwound. This problem can be addressed by:

- Initializing the controller integral to a desired value
- Increasing the setpoint in a suitable ramp
- Disabling the integral function until the PV has entered the controllable region
- Limiting the time period over which the integral error is calculated
- Preventing the integral term from accumulating above or below pre-determined bounds

Overshooting from known disturbances

For example, a PID loop is used to control the temperature of an electric resistance furnace, the system has stabilized. Now the door is opened and something cold is put into the furnace the temperature drops below the setpoint. The integral function of the controller tends to compensate this error by introducing another error in the positive direction. This overshoot can be avoided by freezing of the integral function after the opening of the door for the time the control loop typically needs to reheat the furnace.

Replacing the integral function by a model based part

Often the time-response of the system is approximately known. Then it is an advantage to simulate this time-response with a model and to calculate some unknown parameter from the actual response of the system. If for instance the system is an electrical furnace the response of the difference between furnace temperature and ambient temperature to changes of the electrical power will be similar to that of a simple RC low-pass filter multiplied by an unknown proportional coefficient. The actual electrical power supplied to the furnace is delayed by a low-pass filter to simulate the response of the temperature of the furnace and then the actual temperature minus the ambient temperature is divided by this low-pass filtered electrical power. Then, the result is stabilized by another low-pass filter leading to an estimation of the proportional coefficient. With this estimation, it is possible to calculate the required electrical power by dividing the set-point of the temperature minus the ambient temperature by this coefficient. The result can then be used instead of the integral function. This also achieves a control error of zero in the steady-state, but avoids integral windup and can give a significantly improved control action compared to an optimized PID controller. This type of controller does work properly in an open loop situation which causes integral windup with an integral function. This is an advantage if, for example, the heating of a furnace has to be reduced for some time because of the failure of a heating element, or if the controller is used as an advisory system to a human operator who may not switch it to closed-loop operation. It may also be useful if the controller is inside a branch of a complex control system that may be temporarily inactive.

Many PID loops control a mechanical device (for example, a valve). Mechanical maintenance can be a major cost and wear leads to control degradation in the form of either stiction or a deadband in the mechanical response to an input signal. The rate of mechanical wear is mainly a function of how often a device is activated to make a change. Where wear is a significant concern, the PID loop may have an output deadband to reduce the frequency of activation of the output (valve). This is accomplished by modifying the controller to hold its output steady if the change would be small (within the defined deadband range). The calculated output must leave the deadband before the actual output will change.

The proportional and derivative terms can produce excessive movement in the output when a system is subjected to an instantaneous step increase in the error, such as a large setpoint change. In the case of the derivative term, this is due to taking the derivative of the error, which is very large in the case of an instantaneous step change. As a result, some PID algorithms incorporate the following modifications:

Derivative of output

In this case the PID controller measures the derivative of the output quantity, rather than the derivative of the error. The output is always continuous (i.e., never has a step change). For this to be effective, the derivative of the output must have the same sign as the derivative of the error.

Setpoint ramping

In this modification, the setpoint is gradually moved from its old value to a newly specified value using a linear or first order differential ramp function. This avoids the discontinuity present in a simple step change.

Setpoint weighting

Setpoint weighting uses different multipliers for the error depending on which element of the controller it is used in. The error in the integral term must be the true control error to avoid steady-state control errors. This affects the controller's setpoint response. These parameters do not affect the response to load disturbances and measurement noise.

History

PID theory developed by observing the action of helmsmen.

PID controllers date to 1890s governor design. PID controllers were subsequently developed in automatic ship steering. One of the earliest examples of a PID-type controller was developed by Elmer Sperry in 1911, while the first published theoretical analysis of a PID controller was by Russian American engineer Nicolas Minorsky, in (Minorsky 1922). Minorsky was designing automatic steering systems for the US Navy, and based his analysis on observations of a helmsman, observing that the helmsman controlled the ship not only based on the current error, but also on past error and current rate of change; this was then made mathematical by Minorsky. His goal was stability, not general control, which significantly simplified the problem. While proportional control provides stability against small disturbances, it was insufficient for dealing with a steady disturbance, notably a stiff gale (due to droop), which required adding the integral term. Finally, the derivative term was added to improve control.

Trials were carried out on the USS *New Mexico*, with the controller controlling the *angular velocity* (not angle) of the rudder. PI control yielded sustained yaw (angular error) of ±2°, while adding D yielded yaw of ±1/6°, better than most helmsmen could achieve.

The Navy ultimately did not adopt the system, due to resistance by personnel. Similar work was carried out and published by several others in the 1930s.

Limitations of PID control

While PID controllers are applicable to many control problems, and often perform satisfactorily without any improvements or even tuning, they can perform poorly in some applications, and do not in general provide *optimal* control. The fundamental difficulty with PID control is that it is a feed*back* system, with *constant* parameters, and no direct knowledge of the process, and thus overall performance is reactive and a compromise – while PID control is the best controller with no model of the process, better performance can be obtained by incorporating a model of the process.

The most significant improvement is to incorporate feed-forward control with knowledge about the system, and using the PID only to control error. Alternatively, PIDs can be modified in more minor ways, such as by changing the parameters (either gain scheduling in different use cases or adaptively modifying them based on performance), improving measurement (higher sampling rate, precision, and accuracy, and low-pass filtering if necessary), or cascading multiple PID controllers.

PID controllers, when used alone, can give poor performance when the PID loop gains must be reduced so that the control system does not overshoot, oscillate or *hunt* about the control setpoint value. They also have difficulties in the presence of non-linearities, may trade off regulation versus response time, do not react to changing process behavior (say, the process changes after it has warmed up), and have lag in responding to large disturbances.

Linearity

Another problem faced with PID controllers is that they are linear, and in particular symmetric. Thus, performance of PID controllers in non-linear systems (such as HVAC systems) is variable. For example, in temperature control, a common use case is active heating (via a heating element) but passive cooling (heating off, but no cooling), so overshoot can only be corrected slowly – it cannot be forced downward. In this case the PID should be tuned to be overdamped, to prevent or reduce overshoot, though this reduces performance (it increases settling time).

Noise in derivative

A problem with the derivative term is that small amounts of measurement or process noise can cause large amounts of change in the output. It is often helpful to filter the measurements with a low-pass filter in order to remove higher-frequency noise components. However, low-pass filtering and derivative control can cancel each other out, so reducing noise by instrumentation means is a much better choice. Alternatively, a nonlinear median filter may be used, which improves the filtering efficiency and practical performance. In some case, the differential band can be turned off in many systems with little loss of control. This is equivalent to using the PID controller as a *PI* controller.

Improvements

Feed-forward

The control system performance can be improved by combining the feedback (or closed-loop) control of a PID controller with feed-forward (or open-loop) control. Knowledge about the system (such as the desired acceleration and inertia) can be fed forward and combined with the PID output to improve the overall system performance. The feed-forward value alone can often provide the major portion of the controller output. The PID controller can be used primarily to respond to whatever difference or *error* remains between the setpoint (SP) and the actual value of the process variable (PV). Since the feed-forward output is not affected by the process feedback, it can never cause the control system to oscillate, thus improving the system response and stability.

For example, in most motion control systems, in order to accelerate a mechanical load under control, more force or torque is required from the prime mover, motor, or actuator. If a velocity loop PID controller is being used to control the speed of the load and command the force or torque being applied by the prime mover, then it is beneficial to take the instantaneous acceleration desired for the load, scale that value appropriately and add it to the output of the PID velocity loop controller. This means that whenever the load is being accelerated or decelerated, a proportional amount of force is commanded from the prime mover regardless of the feedback value. The PID loop in this situation uses the feedback information to change the combined output to reduce the remaining difference between the process setpoint and the feedback value. Working together, the combined open-loop feed-forward controller and closed-loop PID controller can provide a more responsive, stable and reliable control system.

Other improvements

In addition to feed-forward, PID controllers are often enhanced through methods such as PID gain scheduling (changing parameters in different operating conditions), fuzzy logic or computational verb logic . Further practical application issues can arise from instrumentation connected to the controller. A high enough sampling rate, measurement precision, and measurement accuracy are required to achieve adequate control performance.

Cascade control

One distinctive advantage of PID controllers is that two PID controllers can be used together to yield better dynamic performance. This is called cascaded PID control. In cascade control there are two PIDs arranged with one PID controlling the set point of another. A PID controller acts as outer loop controller, which controls the primary physical parameter, such as fluid level or velocity. The other controller acts as inner loop controller, which reads the output of outer loop controller as set point, usually controlling a more rapid changing parameter, flowrate or acceleration. It can be mathematically proven that the working frequency of the controller is increased and the time constant of the object is reduced by using cascaded PID controller.

Physical implementation of PID control

In the early history of automatic process control the PID controller was implemented as a mechanical device. These mechanical controllers used a lever, spring and a mass and were often energized by compressed air. These pneumatic controllers were once the industry standard.

Electronic analog controllers can be made from a solid-state or tube amplifier, a capacitor and a resistance. Electronic analog PID control loops were often found within more complex electronic systems, for example, the head positioning of a disk drive, the power conditioning of a power supply, or even the movement-detection circuit of a modern seismometer. Nowadays, electronic controllers have largely been replaced by digital controllers implemented with microcontrollers or FPGAs.

Most modern PID controllers in industry are implemented in programmable logic controllers (PLCs) or as a panel-mounted digital controller. Software implementations have the advantages that they are relatively cheap and are flexible with respect to the implementation of the PID algorithm.

Variable voltages may be applied by the time proportioning form of Pulse-width modulation (PWM) – a cycle time is fixed, and variation is achieved by varying the proportion of the time during this cycle that the controller outputs +1 (or −1) instead of 0. On a digital system the possible proportions are discrete – e.g., increments of .1 second within a 2 second cycle time yields 20 possible steps: percentage increments of 5% – so there is a discretization error, but for high enough time resolution this yields satisfactory performance.

Alternative nomenclature and PID forms

Ideal versus standard PID form

The form of the PID controller most often encountered in industry, and the one most relevant to tuning algorithms is the *standard form*. In this form the K_p gain is applied to the I_{out}, and D_{out} terms, yielding:

$$\mathrm{MV(t)} = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{d}{dt} e(t) \right)$$

where

T_i is the *integral time*
T_d is the *derivative time*

In this standard form, the parameters have a clear physical meaning. In particular, the inner summation produces a new single error value which is compensated for future and past errors. The addition of the proportional and derivative components effectively predicts the error value at T_d seconds (or samples) in the future, assuming that the loop control remains unchanged. The integral component adjusts the error value to compensate for the sum of all past errors, with the intention of completely eliminating them in T_i seconds (or samples). The resulting compensated single error value is scaled by the single gain K_p.

In the ideal parallel form, shown in the controller theory section

$$\mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

the gain parameters are related to the parameters of the standard form through $K_i = \frac{K_p}{T_i}$ and $K_d = K_p T_d$. This parallel form, where the parameters are treated as simple gains, is the most general and flexible form. However, it is also the form where the parameters have the least physical interpretation and is generally reserved for theoretical treatment of the PID controller. The standard form, despite being slightly more complex mathematically, is more common in industry.

Laplace form of the PID controller

Sometimes it is useful to write the PID regulator in Laplace transform form:

$$G(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

Having the PID controller written in Laplace form and having the transfer function of the controlled system makes it easy to determine the closed-loop transfer function of the system.

PID Pole Zero Cancellation

The PID equation can be written in this form:

$$G(s) = K_d \frac{s^2 + \frac{K_p}{K_d} s + \frac{K_i}{K_d}}{s}$$

When this form is used it is easy to determine the closed loop transfer function.

$$H(s) = \frac{1}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$$

If

$$\frac{K_i}{K_d} = \omega_0^2$$
$$\frac{K_p}{K_d} = 2\zeta\omega_0$$

Then

$$G(s)H(s) = \frac{K_d}{s}$$

This can be very useful to remove unstable poles

Series/interacting form

Another representation of the PID controller is the series, or *interacting* form

$$G(s) = K_c \frac{(\tau_i s + 1)}{\tau_i s}(\tau_d s + 1)$$

where the parameters are related to the parameters of the standard form through

$K_p = K_c \cdot \alpha$, $T_i = \tau_i \cdot \alpha$, and
$T_d = \frac{\tau_d}{\alpha}$

with

$$\alpha = 1 + \frac{\tau_d}{\tau_i}.$$

This form essentially consists of a PD and PI controller in series, and it made early (analog) controllers easier to build. When the controllers later became digital, many kept using the interacting form.

Discrete implementation

The analysis for designing a digital implementation of a PID controller in a Microcontroller (MCU) or FPGA device requires the standard form of the PID controller to be *discretised* . Approximations for first-order derivatives are made by backward finite differences. The integral term is discretised, with a sampling time Δt,as follows,

$$\int_0^{t_k} e(\tau)\,d\tau = \sum_{i=1}^{k} e(t_i)\Delta t$$

The derivative term is approximated as,

$$\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

Thus, a *velocity algorithm* for implementation of the discretised PID controller in a MCU is obtained by differentiating $u(t)$, using the numerical definitions of the first and second derivative and solving for $u(t_k)$ and finally obtaining:

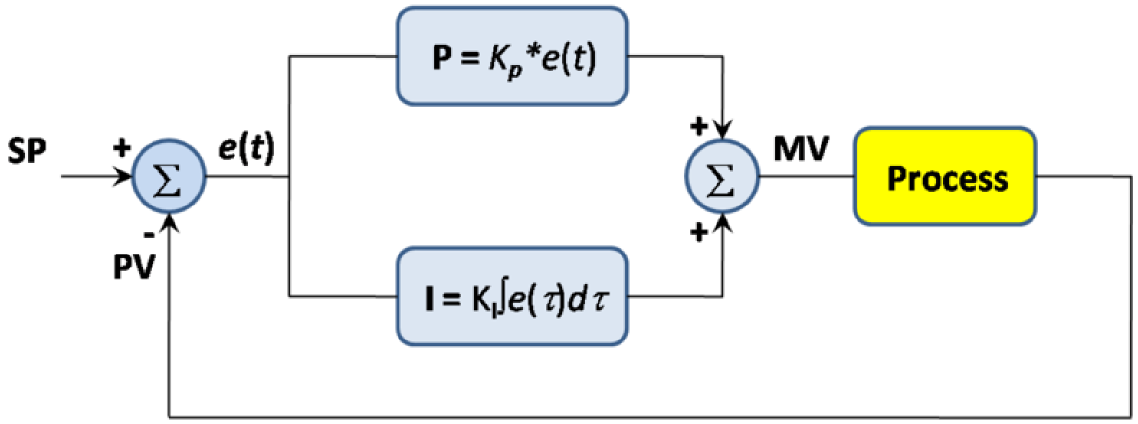
$$u(t_k) = u(t_{k-1}) + K_p \left[\left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t}\right) e(t_k) + \left(-1 - \frac{2T_d}{\Delta t}\right) e(t_{k-1}) + \frac{T_d}{\Delta t} e(t_{k-2}) \right]$$

Pseudocode

Here is a simple software loop that implements the PID algorithm in its 'ideal, parallel' form:

```
previous_error = 0
integral = 0
start:
  error = setpoint - actual_position
  integral = integral + (error*dt)
  derivative = (error - previous_error)/dt
  output = (Kp*error) + (Ki*integral) + (Kd*derivative)
  previous_error = error
  wait(dt)
  goto start
```

PI controller



Basic block of a PI controller.

A **PI Controller** (proportional-integral controller) is a special case of the PID controller in which the derivative (D) of the error is not used.

The controller output is given by

$$K_P\Delta + K_I \int \Delta \, dt$$

where Δ is the error or deviation of actual measured value (***PV***) from the set-point (***SP***).

Δ = **SP - PV**.

A PI controller can be modelled easily in software such as Simulink using a "flow chart" box involving Laplace operators:

$$C = \frac{G(1+\tau s)}{\tau s}$$

where

$G = K_P$ = proportional gain
$G / \tau = K_I$ = integral gain

Setting a value for G is often a trade off between decreasing overshoot and increasing settling time.

The lack of derivative action may make the system more steady in the steady state in the case of noisy data. This is because derivative action is more sensitive to higher-frequency terms in the inputs.

Without derivative action, a PI-controlled system is less responsive to real (non-noise) and relatively fast alterations in state and so the system will be slower to reach setpoint and slower to respond to perturbations than a well-tuned PID system may be.

Chapter- 4

Controllability & Observability

Controllability

Controllability is an important property of a control system, and the controllability property plays a crucial role in many control problems, such as stabilization of unstable systems by feedback, or optimal control.

Controllability and observability are dual aspects of the same problem.

Roughly, the concept of controllability denotes the ability to move a system around in its entire configuration space using only certain admissible manipulations. The exact definition varies slightly within the framework or the type of models applied.

The following are examples of variations of controllability notions which have been introduced in the systems and control literature:

- State controllability
- Output controllability
- Controllability in the behavioural framework

State controllability

The state of a system, which is a collection of system's variables values, completely describes the system at any given time. In particular, no information on the past of a system will help in predicting the future, if the states at the present time are known.

Complete state controllability (or simply *controllability* if no other context is given) describes the ability of an external input to move the internal state of a system from any initial state to any other final state in a finite time interval. [:737]

Continuous Linear Systems

Consider the continuous linear time-variant system

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$
$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t).$$

There exists a control u from state x_0 at time t_0 to state x_1 at time $t_1 > t_0$ if and only if $x_0 - \varphi(t_0,t_1)x_1$ is in the column space of

$$W(t_0,t_1) = \int_{t_0}^{t_1} \phi(t_0,t)B(t)B(t)^T\phi(t_0,t)^T dt$$

where φ is the state-transition matrix.

In fact, if η_0 is a solution to $W(t_0,t_1)\eta = x_0 - \varphi(t_0,t_1)x_1$ then a control given by $u(t) = -B(t)^T\varphi(t_0,t_1)^T\eta_0$ would make the desired transfer.

Note that the matrix W defined as above has the following properties:

- $W(t_0,t_1)$ is symmetric
- $W(t_0,t_1)$ is positive semidefinite for $t_1 \geq t_0$
- $W(t_0,t_1)$ satisfies the linear matrix differential equation

$$\frac{d}{dt}W(t,t_1) = A(t)W(t,t_1) + W(t,t_1)A(t)^T - B(t)B(t)^T,\ W(t_1,t_1) = 0$$

- $W(t_0,t_1)$ satisfies the equation

  $W(t_0,t_1) = W(t_0,t) + \varphi(t_0,t)W(t,t_1)\varphi(t_0,t)^T$

Continuous Linear Time-Invariant (LTI) Systems

Consider the continuous linear time-invariant system

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

where

$\mathbf{x}$ is the $n \times 1$ "state vector",
$\mathbf{y}$ is the $m \times 1$ "output vector",
$\mathbf{u}$ is the $r \times 1$ "input (or control) vector",
A is the $n \times n$ "state matrix",

B is the $n \times r$"input matrix",
C is the $m \times n$"output matrix",
D is the $m \times r$"feedthrough (or feedforward) matrix".

The $n \times nr$controllability matrix is given by

$$R = \begin{bmatrix} B & AB & A^2B & ... & A^{n-1}B \end{bmatrix}$$

The system is controllable if the controllability matrix has full rank (i.e. $\operatorname{rank}(R) = n$).

Discrete Linear Time-Invariant (LTI) Systems

For a discrete-time linear state-space system (i.e. time variable $k \in \mathbb{Z}$) the state equation is

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$$

Where *A* is an $n \times n$matrix and *B* is a $n \times r$matrix (i.e. $\mathbf{u}$is *r* inputs collected in a $r \times 1$vector. The test for controllability is that the $n \times nr$matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & ... & A^{n-1}B \end{bmatrix}$$

has full row rank (i.e., *rank*(*C*) = *n*). That is, if the system is controllable, *C* will have *n* columns that are linearly independent; if *n* columns of *C* are linearly independent, each of the *n* states is reachable giving the system proper inputs through the variable *u*(*k*).

Example

For example, consider the case when *n* = 2 and *r* = 1 (i.e. only one control input). Thus, *B* and *AB* are $n \times 1$vectors. If $\begin{bmatrix} B & AB \end{bmatrix}$ has rank 2 (full rank), and so *B* and *AB* are linearly independent and span the entire plane. If the rank is 1, then *B* and *AB* are collinear and do not span the plane.

Assume that the initial state is zero.

At time *k* = 0: $x(1) = A\mathbf{x}(0) + B\mathbf{u}(0) = B\mathbf{u}(0)$

At time *k* = 1: $x(2) = A\mathbf{x}(1) + B\mathbf{u}(1) = AB\mathbf{u}(0) + B\mathbf{u}(1)$

At time *k* = 0 all of the reachable states are on the line formed by the vector *B*. At time *k* = 1 all of the reachable states are linear combinations of *AB* and *B*. If the system is controllable then these two vectors can span the entire plane and can be done so for time *k* = 2. The assumption made that the initial state is zero is merely for convenience. Clearly if all states can be reached from the origin then any state can be reached from another state (merely a shift in coordinates).

This example holds for all positive *n*, but the case of *n* = 2 is easier to visualize.

Analogy for example of *n* = 2

Consider an analogy to the previous example system. You are sitting in your car on an infinite, flat plane and facing north. The goal is to reach any point in the plane by driving a distance in a straight line, come to a full stop, turn, and driving another distance, again, in a straight line. If your car has no steering then you can only drive straight, which means you can only drive on a line (in this case the north-south line since you started facing north). The lack of steering case would be analogous to when the rank of *C* is 1 (the two distances you drove are on the same line).

Now, if your car did have steering then you could easily drive to any point in the plane and this would be the analogous case to when the rank of *C* is 2.

If you change this example to *n* = 3 then the analogy would be flying in space to reach any position in 3D space (ignoring the orientation of the aircraft). You are allowed to:

- fly in a straight line
- turn left or right by any amount (Yaw)
- direct the plane upwards or downwards by any amount (Pitch)

Although the 3-dimensional case is harder to visualize, the concept of controllability is still analogous.

Nonlinear Systems

Nonlinear systems in the control-affine form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{g}_i(\mathbf{x}) u_i$$

is locally accessible about x_0 if the accessibility distribution *R* spans *n* space, when *n* equals the rank of *x* and R is given by :

$$R = \begin{bmatrix} \mathbf{g}_1 & \cdots & \mathbf{g}_m & [\mathrm{ad}^k_{\mathbf{g_i}} \mathbf{g_j}] & \cdots & [\mathrm{ad}^k_{\mathbf{f}} \mathbf{g_i}] \end{bmatrix}.$$

Here, $[\mathrm{ad}^k_{\mathbf{f}} \mathbf{g}]$is the repeated Lie bracket operation defined by

$$[\mathrm{ad}_{\mathbf{f}}^{k}\mathbf{g}] = \begin{bmatrix} \mathbf{f} & \cdots & j & \cdots & [\mathbf{f}, \mathbf{g}] \end{bmatrix}.$$

The controllability matrix for linear systems in the previous section can in fact be derived from this equation.

Output controllability

Output controllability is the related notion for the output of the system; the output controllability describes the ability of an external input to move the output from any initial condition to any final condition in a finite time interval. A controllable system is not necessarily output controllable, and an output controllable system is not necessarily controllable.

For a linear continuous-time system, like the example above, described by matrices *A*, *B*, *C*, and *D*, the $m \times (n+1)r$ *output controllability matrix*

$$\begin{bmatrix} CB & CAB & CA^2B & \cdots & CA^{n-1}B & D \end{bmatrix}$$

must have full row rank (i.e. rank *m*) if and only if the system is output controllable. :742

Controllability in the behavioural framework

In the so-called behavioral system theoretic approach due to Willems, models considered do not directly define an input–output structure. In this framework systems are described by admissible trajectories of a collection of variables, some of which might be interpreted as inputs or outputs.

A system is then defined to be controllable in this setting, if any past part of a behavior (state trajectory) can be concatenated with any future part of a behavior with which it shares the current state in such a way that the concatenation is contained in the behavior, i.e. is part of the admissible system behavior.

Stabilizability

A slightly weaker notion than controllability is that of **Stabilizability**. A system is determined to be stabilizable when all uncontrollable states have stable dynamics. Thus, even though some of the states cannot be controlled (as determined by the controllability test above) all the states will still remain bounded during the system's behavior.

Observability

Observability, in control theory, is a measure for how well internal states of a system can be inferred by knowledge of its external outputs. The observability and controllability of a system are mathematical duals. The concept of observability was introduced by American-Hungarian scientist Rudolf E. Kalman for linear dynamic systems , .

Definition

Formally, a system is said to be **observable** if, for any possible sequence of state and control vectors, the current state can be determined in finite time using only the outputs (this definition is slanted towards the state space representation). Less formally, this means that from the system's outputs it is possible to determine the behaviour of the entire system. If a system is not observable, this means the current values of some of its states cannot be determined through output sensors: this implies that their value is unknown to the controller and, consequently, that it will be unable to fulfil the control specifications referred to these outputs.

For time-invariant linear systems in the state space representation, a convenient test to check if a system is observable exists. Consider a SISO system with *n* states, if the rank of the following *observability matrix*

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

is equal to *n*, then the system is observable. The rationale for this test is that if *n* rows are linearly independent, then each of the *n* states is viewable through linear combinations of the output variables *y*(*k*).

A module designed to estimate the state of a system from measurements of the outputs is called a state observer or simply an observer for that system.

Observability index

The Observability index *v* of a linear time-invariant discrete system is the smallest natural number for which is satisfied that $rank(O_v) = rank(O_{v+1})$, where

$$\mathcal{O}_v = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{v-1} \end{bmatrix}.$$

Detectability

A slightly weaker notion is Detectability. A system is detectable if and only if all of its unobservable modes are stable. Thus even though not all system modes are observable, the ones that are not observable do not require stabilization.

Continuous Time-Varying System

Consider the Consider the continuous linear time-variant system

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t) \\ \mathbf{y}(t) &= C(t)\mathbf{x}(t).\end{aligned}$$

Suppose that the matrices *A*,*B*, and *C* are given as well as inputs and outputs *u* and *y* for all $t \in [t_0, t_1]$then it is possible to determine $x(t_0)$ to within an additive constant vector which lies in the null space of $M(t_0,t_1)$ defined by

$$M(t_0, t_1) = \int_{t_0}^{t_1} \phi(t, t_0)^T C(t)^T C(t) \phi(t, t_0) dt$$

where φ is the state-transition matrix.

It is possible to determine a unique $x(t_0)$ if $M(t_0,t_1)$ is nonsingular. In fact, it is not possible to distinguish the initial state for x_1 from that of x_2 if $x_1 - x_2$ is in the null space of $M(t_0,t_1)$.

Note that the matrix *M* defined as above has the following properties:

- $M(t_0,t_1)$ is symmetric
- $M(t_0,t_1)$ is positive semidefinite for $t_1 \geq t_0$
- $M(t_0,t_1)$ satisfies the linear matrix differential equation

$$\frac{d}{dt}M(t,t_1) = -A(t)^T M(t,t_1) - M(t,t_1)A(t) - C(t)^T C(t),\ M(t_1,t_1) = 0$$

- $M(t_0,t_1)$ satisfies the equation

$M(t_0,t_1) = M(t_0,t) + \varphi(t,t_0)^T M(t,t_1)\varphi(t,t_0)$

Nonlinear case

Given the system $\dot{x} = f(x) + \sum_{j=1}^{m} g_j(x)u_j$, $y_i = h_i(x), i \in p$. Where $x \in \mathbb{R}^n$ the state vector, $u \in \mathbb{R}^m$ the input vector and $y \in \mathbb{R}^p$ the output vector. *f*,*g*,*h* are to be smooth vectorfields.

Now define the observation space $\mathcal{O}_s$ to be the space containing all repeated Lie derivatives. Now the system is observable in x_0 if and only if $\dim(d\mathcal{O}_s(x_0)) = n$.

Note:
$d\mathcal{O}_s(x_0) = \operatorname{span}(dh_1(x_0), ..., dh_p(x_0), dL_{v_i}L_{v_{i-1}}, ..., L_{v1}h_j(x_0)), j \in p, k = 1, 2, ...$
.

Early criteria for observability in nonlinear dynamic systems were discovered by Griffith and Kumar , Kou, Elliot and Tarn , and Singh .

Static systems and general topological spaces

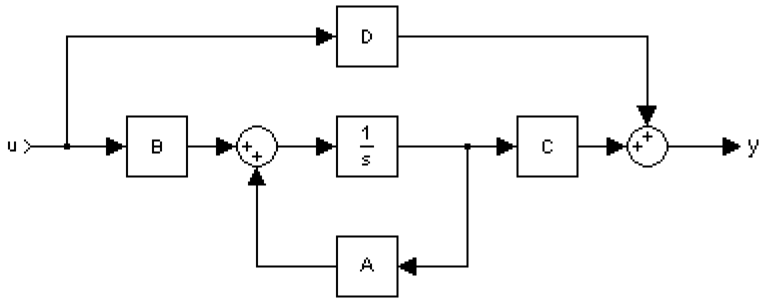
Observability may also be characterized for steady state systems (systems typically defined in terms of algebraic equations and inequalities), or more generally, for sets in $\mathbb{R}^n$, . Just as observability criteria are used to predict the behavior of Kalman filters or other observers in the dynamic system case, observability criteria for sets in $\mathbb{R}^n$ are used to predict the behavior of data reconciliation and other static estimators. In the nonlinear case, observability can be characterized for individual variables, and also for local estimator behavior rather than just global behavior.

Chapter- 5

State Space (Controls)

In control engineering, a **state space representation** is 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 last one can be done when the dynamical system is linear and time invariant). 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 *p* inputs and *q* outputs, we would otherwise have to write down $q \times p$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.

State variables



Block diagram representation of the state space equations

The internal state variables are the smallest possible subset of system variables that can represent the entire state of the system at any given time. The minimum number of state variables required to represent a given system, *n*, is usually equal to the order of the system's defining differential equation. If the system is represented in transfer function form, the minimum number of state variables is equal to the order of the transfer function's denominator after it has been reduced to a proper fraction. It is important to understand that converting a state space realization to a transfer function form may lose some internal information about the system, and may provide a description of a system which is stable, when the state-space realization is unstable at certain points. In electric circuits, the number of state variables is often, though not always, the same as the number of energy storage elements in the circuit such as capacitors and inductors.

Linear systems

The most general state-space representation of a linear system with *p* inputs, *q* outputs and *n* state variables is written in the following form:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)\\ \mathbf{y}(t) &= C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)\end{aligned}$$

where:

$\mathbf{x}(\cdot)$ is called the "state vector", $\mathbf{x}(t) \in \mathbb{R}^n$;
$\mathbf{y}(\cdot)$ is called the "output vector", $\mathbf{y}(t) \in \mathbb{R}^q$;
$\mathbf{u}(\cdot)$ is called the "input (or control) vector", $\mathbf{u}(t) \in \mathbb{R}^p$;
$A(\cdot)$ is the "state matrix", $\dim[A(\cdot)] = n \times n$,
$B(\cdot)$ is the "input matrix", $\dim[B(\cdot)] = n \times p$,
$C(\cdot)$ is the "output matrix", $\dim[C(\cdot)] = q \times n$,
$D(\cdot)$ is the "feedthrough (or feedforward) matrix" (in cases where the system model does not have a direct feedthrough, $D(\cdot)$ is the zero matrix), $\dim[D(\cdot)] = q \times p$,
$\dot{\mathbf{x}}(t) := \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t)$.

In this general formulation, all matrices are allowed to be time-variant (i.e., their elements can depend on time); however, in the common LTI case, matrices will be time invariant. The time variable *t* can be a "continuous" (e.g., $t \in \mathbb{R}$) or discrete (e.g., $t \in \mathbb{Z}$). In the latter case, the time variable is usually indicated as *k*. Hybrid systems allow for time domains that have both continuous and discrete parts. Depending on the assumptions taken, the state-space model representation can assume the following forms:

System type	**State-space model**
Continuous time-invariant	$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$ $\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$
Continuous time-variant	$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$ $\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t)$
Discrete time-invariant	$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$

$$\mathbf{y}(k) = C\mathbf{x}(k) + D\mathbf{u}(k)$$

Discrete time-variant

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k)$$
$$\mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{D}(k)\mathbf{u}(k)$$

Laplace domain of continuous time-invariant

$$s\mathbf{X}(s) = A\mathbf{X}(s) + B\mathbf{U}(s)$$
$$\mathbf{Y}(s) = C\mathbf{X}(s) + D\mathbf{U}(s)$$

Z-domain of discrete time-invariant

$$z\mathbf{X}(z) = A\mathbf{X}(z) + B\mathbf{U}(z)$$
$$\mathbf{Y}(z) = C\mathbf{X}(z) + D\mathbf{U}(z)$$

Example: Continuous-time LTI case

Stability and natural response characteristics of a continuous-time LTI system (i.e., linear with matrices that are constant with respect to time) can be studied from the eigenvalues of the matrix **A**. The stability of a time-invariant state-space model can be determined by looking at the system's transfer function in factored form. It will then look something like this:

$$\mathbf{G}(s) = k\frac{(s-z_1)(s-z_2)(s-z_3)}{(s-p_1)(s-p_2)(s-p_3)(s-p_4)}$$

The denominator of the transfer function is equal to the characteristic polynomial found by taking the determinant of $sI - A$,

$$\lambda(s) = |sI - A|.$$

The roots of this polynomial (the eigenvalues) are the system transfer function's poles (i.e., the singularities where the transfer function's magnitude is unbounded). These poles can be used to analyze whether the system is asymptotically stable or marginally stable. An alternative approach to determining stability, which does not involve calculating eigenvalues, is to analyze the system's Lyapunov stability.

The zeros found in the numerator of $\mathbf{G}(s)$ can similarly be used to determine whether the system is minimum phase.

The system may still be **input–output stable** even though it is not internally stable. This may be the case if unstable poles are canceled out by zeros (i.e., if those singularities in the transfer function are removable).

Controllability

Thus, state controllability condition implies that it is possible – by admissible inputs – to steer the states from any initial value to any final value within some finite time window. A continuous time-invariant linear state-space model is **controllable** if and only if

$$\text{rank}\begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} = n$$

(Rank is the number of linearly independent rows in a matrix.)

Observability

Observability is a measure for how well internal states of a system can be inferred by knowledge of its external outputs. The observability and controllability of a system are mathematical duals (i.e., as controllability provides that an input is available that brings any initial state to any desired final state, observability provides that knowing an output trajectory provides enough information to predict the initial state of the system).

A continuous time-invariant linear state-space model is **observable** if and only if

$$\text{rank}\begin{bmatrix} C \\ CA \\ \dots \\ CA^{n-1} \end{bmatrix} = n$$

Transfer function

The "transfer function" of a continuous time-invariant linear state-space model can be derived in the following way:

First, taking the Laplace transform of

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

yields

$$s\mathbf{X}(s) = A\mathbf{X}(s) + B\mathbf{U}(s)$$

Next, we simplify for $\mathbf{X}(s)$, giving

$$(s\mathbf{I} - A)\mathbf{X}(s) = B\mathbf{U}(s)$$
$$\mathbf{X}(s) = (s\mathbf{I} - A)^{-1}B\mathbf{U}(s)$$

this is substituted for $\mathbf{X}(s)$in the output equation

$$\mathbf{Y}(s) = C\mathbf{X}(s) + D\mathbf{U}(s)$$, giving
$$\mathbf{Y}(s) = C((s\mathbf{I} - A)^{-1}B\mathbf{U}(s)) + D\mathbf{U}(s)$$

Because the transfer function $\mathbf{G}(s)$ is defined as the ratio of the output to the input of a system, we take

$$\mathbf{G}(s) = \mathbf{Y}(s)/\mathbf{U}(s)$$

and substitute the previous expression for $\mathbf{Y}(s)$ with respect to $\mathbf{U}(s)$, giving

$$\mathbf{G}(s) = C(s\mathbf{I} - A)^{-1}B + D$$

Clearly $\mathbf{G}(s)$ must have *q* by *p* dimensionality, and thus has a total of *qp* elements. So for every input there are *q* transfer functions with one for each output. This is why the state-space representation can easily be the preferred choice for multiple-input, multiple-output (MIMO) systems.

Canonical realizations

Any given transfer function which is strictly proper can easily be transferred into state-space by the following approach (this example is for a 4-dimensional, single-input, single-output system)):

Given a transfer function, expand it to reveal all coefficients in both the numerator and denominator. This should result in the following form:

$$\mathbf{G}(s) = \frac{n_1 s^3 + n_2 s^2 + n_3 s + n_4}{s^4 + d_1 s^3 + d_2 s^2 + d_3 s + d_4}.$$

The coefficients can now be inserted directly into the state-space model by the following approach:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -d_1 & -d_2 & -d_3 & -d_4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix} n_1 & n_2 & n_3 & n_4 \end{bmatrix} \mathbf{x}(t).$$

This state-space realization is called **controllable canonical form** because the resulting model is guaranteed to be controllable (i.e., because the control enters a chain of integrators, it has the ability to move every state).

The transfer function coefficients can also be used to construct another type of canonical form

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -d_1 & 1 & 0 & 0 \\ -d_2 & 0 & 1 & 0 \\ -d_3 & 0 & 0 & 1 \\ -d_4 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix} \mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t).$$

This state-space realization is called **observable canonical form** because the resulting model is guaranteed to be observable (i.e., because the output exits from a chain of integrators, every state has an effect on the output).

Proper transfer functions

Transfer functions which are only proper (and not strictly proper) can also be realised quite easily. The trick here is to separate the transfer function into two parts: a strictly proper part and a constant.

$$\mathbf{G}(s) = \mathbf{G}_{SP}(s) + \mathbf{G}(\infty)$$

The strictly proper transfer function can then be transformed into a canonical state space realization using techniques shown above. The state space realization of the constant is trivially $\mathbf{y}(t) = \mathbf{G}(\infty)\mathbf{u}(t)$. Together we then get a state space realization with matrices *A*,*B* and *C* determined by the strictly proper part, and matrix *D* determined by the constant.

Here is an example to clear things up a bit:

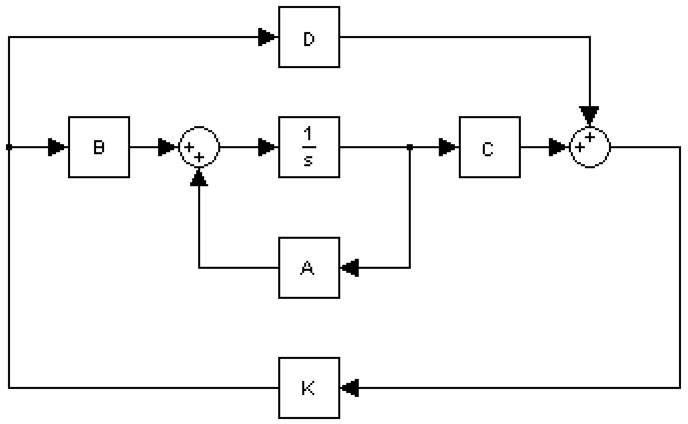
$$\mathbf{G}(s) = \frac{s^2 + 3s + 3}{s^2 + 2s + 1} = \frac{s + 2}{s^2 + 2s + 1} + 1$$

which yields the following controllable realization

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix} 1 & 2 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \end{bmatrix} \mathbf{u}(t)$$

Notice how the output also depends directly on the input. This is due to the $\mathbf{G}(\infty)$constant in the transfer function.

Feedback



Typical state space model with feedback

A common method for feedback is to multiply the output by a matrix *K* and setting this as the input to the system: $\mathbf{u}(t) = K\mathbf{y}(t)$. Since the values of *K* are unrestricted the values can easily be negated for negative feedback. The presence of a negative sign (the common notation) is merely a notational one and its absence has no impact on the end results.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

becomes

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + BK\mathbf{y}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + DK\mathbf{y}(t)$$

solving the output equation for $\mathbf{y}(t)$and substituting in the state equation results in

$$\dot{\mathbf{x}}(t) = \left(A + BK\left(I - DK\right)^{-1}C\right)\mathbf{x}(t)$$
$$\mathbf{y}(t) = \left(I - DK\right)^{-1}C\mathbf{x}(t)$$

The advantage of this is that the eigenvalues of *A* can be controlled by setting *K* appropriately through eigendecomposition of $\left(A + BK\left(I - DK\right)^{-1}C\right)$. This assumes that the open-loop system is controllable or that the unstable eigenvalues of *A* can be made stable through appropriate choice of *K*.

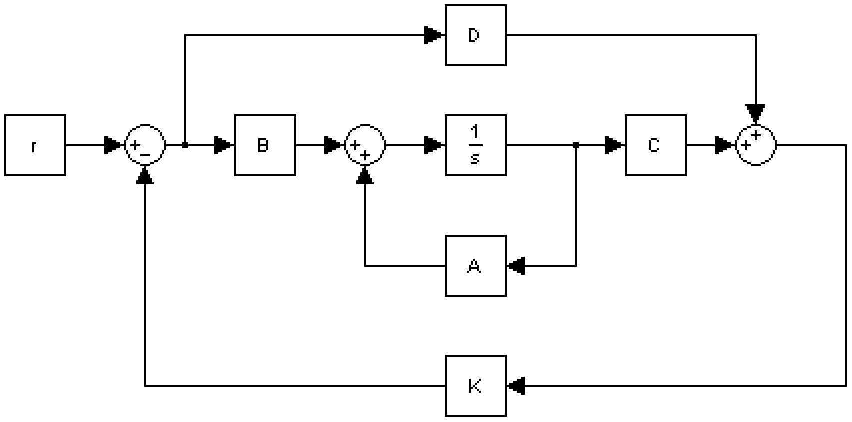
One fairly common simplification to this system is removing *D* and setting *C* to identity, which reduces the equations to

$$\dot{\mathbf{x}}(t) = \left(A + BK\right)\mathbf{x}(t)$$

$$\mathbf{y}(t) = \mathbf{x}(t)$$

This reduces the necessary eigendecomposition to just *A* + *BK*.

Feedback with setpoint (reference) input



Output feedback with set point

In addition to feedback, an input, *r*(*t*), can be added such that $\mathbf{u}(t) = -K\mathbf{y}(t) + \mathbf{r}(t)$.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

becomes

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) - BK\mathbf{y}(t) + B\mathbf{r}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) - DK\mathbf{y}(t) + D\mathbf{r}(t)$$

solving the output equation for $\mathbf{y}(t)$ and substituting in the state equation results in

$$\dot{\mathbf{x}}(t) = \left(A - BK\left(I + DK\right)^{-1}C\right)\mathbf{x}(t) + B\left(I - K\left(I + DK\right)^{-1}D\right)\mathbf{r}(t)$$
$$\mathbf{y}(t) = \left(I + DK\right)^{-1}C\mathbf{x}(t) + \left(I + DK\right)^{-1}D\mathbf{r}(t)$$

One fairly common simplification to this system is removing *D*, which reduces the equations to

$$\dot{\mathbf{x}}(t) = \left(A - BKC\right)\mathbf{x}(t) + B\mathbf{r}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t)$$

Moving object example

A classical linear system is that of one-dimensional movement of an object. The Newton's laws of motion for an object moving horizontally on a plane and attached to a wall with a spring

$$m\ddot{y}(t) = u(t) - k_1\dot{y}(t) - k_2 y(t)$$

where

- $y(t)$ is position; $\dot{y}(t)$ is velocity; $\ddot{y}(t)$ is acceleration
- $u(t)$ is an applied force
- k_1 is the viscous friction coefficient
- k_2 is the spring constant
- m is the mass of the object

The state equation would then become

$$\begin{bmatrix}\dot{\mathbf{x}}_{\mathbf{1}}(t)\\ \dot{\mathbf{x}}_{\mathbf{2}}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\frac{k2}{m} & -\frac{k1}{m}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathbf{1}}(t)\\ \mathbf{x}_{\mathbf{2}}(t)\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathbf{1}}(t)\\ \mathbf{x}_{\mathbf{2}}(t)\end{bmatrix}$$

where

- $x_1(t)$ represents the position of the object
- $x_2(t) = \dot{x}_1(t)$ is the velocity of the object
- $\dot{x_2}(t) = \ddot{x}_1(t)$ is the acceleration of the object
- the output $\mathbf{y}(t)$ is the position of the object

The controllability test is then

$$\begin{bmatrix}B & AB\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix} & \begin{bmatrix}0 & 1\\ -\frac{k2}{m} & -\frac{k1}{m}\end{bmatrix}\begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0 & \frac{1}{m}\\ \frac{1}{m} & \frac{k1}{m^2}\end{bmatrix}$$

which has full rank for all k_1 and m.

The observability test is then

$$\begin{bmatrix}C\\ CA\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}\\ \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}0 & 1\\ -\frac{k2}{m} & -\frac{k1}{m}\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

which also has full rank. Therefore, this system is both controllable and observable.

Nonlinear systems

The more general form of a state space model can be written as two functions.

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}(t, x(t), u(t)) \\ \mathbf{y}(t) &= \mathbf{h}(t, x(t), u(t))\end{aligned}$$

The first is the state equation and the latter is the output equation. If the function $f(\cdot,\cdot,\cdot)$ is a linear combination of states and inputs then the equations can be written in matrix notation like above. The *u*(*t*) argument to the functions can be dropped if the system is unforced (i.e., it has no inputs).

Pendulum example

A classic nonlinear system is a simple unforced pendulum

$$ml\ddot{\theta}(t) = -mg\sin\theta(t) - kl\dot{\theta}(t)$$

where

- θ(*t*) is the angle of the pendulum with respect to the direction of gravity
- *m* is the mass of the pendulum (pendulum rod's mass is assumed to be zero)
- *g* is the gravitational acceleration
- *k* is coefficient of friction at the pivot point
- *l* is the radius of the pendulum (to the center of gravity of the mass *m*)

The state equations are then

$$\begin{aligned}\dot{x_1}(t) &= x_2(t) \\ \dot{x_2}(t) &= -\frac{g}{l}\sin x_1(t) - \frac{k}{m}x_2(t)\end{aligned}$$

where

- $x_1(t)$ = θ(*t*) is the angle of the pendulum
- $x_2(t) = \dot{x_1}(t)$ is the rotational velocity of the pendulum
- $\dot{x_2} = \ddot{x}_1$ is the rotational acceleration of the pendulum

Instead, the state equation can be written in the general form

$$\dot{x}(t) = \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \mathbf{f}(t, x(t)) = \begin{pmatrix} x_2(t) \\ -\frac{g}{l} \sin x_1(t) - \frac{k}{m} x_2(t) \end{pmatrix}.$$

The equilibrium/stationary points of a system are when $\dot{x} = 0$and so the equilibrium points of a pendulum are those that satisfy

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} n\pi \\ 0 \end{pmatrix}$$

for integers *n*.

Chapter- 6

Nonlinear Control

Nonlinear control is the area of control engineering specifically involved with systems that are nonlinear, time-variant, or both. Many well-established analysis and design techniques exist for LTI systems (e.g., root-locus, Bode plot, Nyquist criterion, state-feedback, pole placement); however, one or both of the controller and the system under control in a general control system may not be an LTI system, and so these methods cannot necessarily be applied directly. Nonlinear control theory studies how to apply existing linear methods to these more general control systems. Additionally, it provides novel control methods that cannot be analyzed using LTI system theory. Even when LTI system theory can be used for the analysis and design of a controller, a nonlinear controller can have attractive characteristics (e.g., simpler implementation, increased speed, or decreased control energy); however, nonlinear control theory usually requires more rigorous mathematical analysis to justify its conclusions.

Properties of nonlinear systems

Some properties of nonlinear dynamic systems are

- They do not follow the principle of superposition (linearity and homogeneity).
- They may have multiple isolated equilibrium points.
- They may exhibit properties such as limit-cycle, bifurcation, chaos.
- Finite escape time: Solutions of nonlinear systems may not exist for all times.

Analysis and control of nonlinear systems

There are several well-developed techniques for analyzing nonlinear feedback systems:

- Describing function method
- Phase plane method
- Lyapunov stability analysis
- Singular perturbation method

- Popov criterion (described in *The Lur'e Problem* below)
- Center manifold theorem
- Small-gain theorem
- Passivity analysis

Control design techniques for nonlinear systems also exist. These can be subdivided into techniques which attempt to treat the system as a linear system in a limited range of operation and use (well-known) linear design techniques for each region:

- Gain scheduling

Those that attempt to introduce auxiliary nonlinear feedback in such a way that the system can be treated as linear for purposes of control design:

- Feedback linearization

And Lyapunov based methods:

- Lyapunov redesign
- Nonlinear damping
- Backstepping
- Sliding mode control

Nonlinear feedback analysis — The Lur'e problem

An early nonlinear feedback system analysis problem was formulated by A. I. Lur'e. Control systems described by the Lur'e problem have a forward path that is linear and time-invariant, and a feedback path that contains a memory-less, possibly time-varying, static nonlinearity.

Lur'e Problem Block Diagram

The linear part can be characterized by four matrices (A,B,C,D), while the nonlinear part is Φ(y) with $\frac{\Phi(y)}{y} \in [a, b], \quad a < b \quad \forall y$ (a sector nonlinearity).

Absolute stability problem

Consider:

1. (A,B) is controllable and (C,A) is observable
2. two real numbers a, b with a<b, defining a sector for function Φ

The problem is to derive conditions involving only the transfer matrix H(s) and {a,b} such that x=0 is a globally uniformly asymptotically stable equilibrium of the system. This is known as the Lur'e problem.

There are two main theorems concerning the problem:

- The Circle criterion
- The Popov criterion.

Popov criterion

The sub-class of Lur'e systems studied by Popov is described by:

$$\begin{aligned} \dot{x} &= Ax + bu \\ \dot{\xi} &= u \\ y &= cx + d\xi \quad (1) \end{aligned}$$

$$u = -\phi(y) \quad (2)$$

where x $\in$ R^n, ξ,u,y are scalars and A,b,c,d have commensurate dimensions. The nonlinear element Φ: R → R is a time-invariant nonlinearity belonging to *open sector* (0, ∞). This means that

$\Phi(0) = 0,\ y\,\Phi(y) > 0,\ \forall\ y \neq 0;$

The transfer function from u to y is given by

$$H(s) = \frac{d}{s} + c(sI - A)^{-1}b$$

Theorem: Consider the system (1)-(2) and suppose

1. A is Hurwitz
2. (A,b) is controllable
3. (A,c) is observable
4. d>0 and
5. Φ $\in$ (0,∞)

then the system is globally asymptotically stable if there exists a number r>0 such that $\inf_{\omega \in R} \mathrm{Re}[(1+j\omega r)h(j\omega)] > 0$.

Things to be noted:

- The Popov criterion is applicable only to autonomous systems

- The system studied by Popov has a pole at the origin and there is no direct pass-through from input to output
- The nonlinearity Φ must satisfy an open sector condition

Theoretical Results in Nonlinear Control

Frobenius Theorem

The Frobenius theorem is a deep result in Differential Geometry. When applied to Nonlinear Control, it says the following: Given a system of the form

$$\dot{x} = \sum_{i=1}^{k} f_i(x) u_i(t)$$

where $x \in R^n, f_1, \ldots, f_k$ are vector fields belonging to a distribution Δ and $u_i(t)$ are control functions, the integral curves of *x* are restricted to a manifold of dimension *m* if span(Δ) = *m* and Δ is an involutive distribution.

Chapter- 7

Adaptive Control & System Identification

Adaptive control

Adaptive control involves modifying the control law used by a controller to cope with the fact that the parameters of the system being controlled are slowly time-varying or uncertain. For example, as an aircraft flies, its mass will slowly decrease as a result of fuel consumption; we need a control law that adapts itself to such changing conditions. Adaptive control is different from robust control in the sense that it does not need a priori information about the bounds on these uncertain or time-varying parameters; robust control guarantees that if the changes are within given bounds the control law need not be changed, while adaptive control is precisely concerned with control law changes.

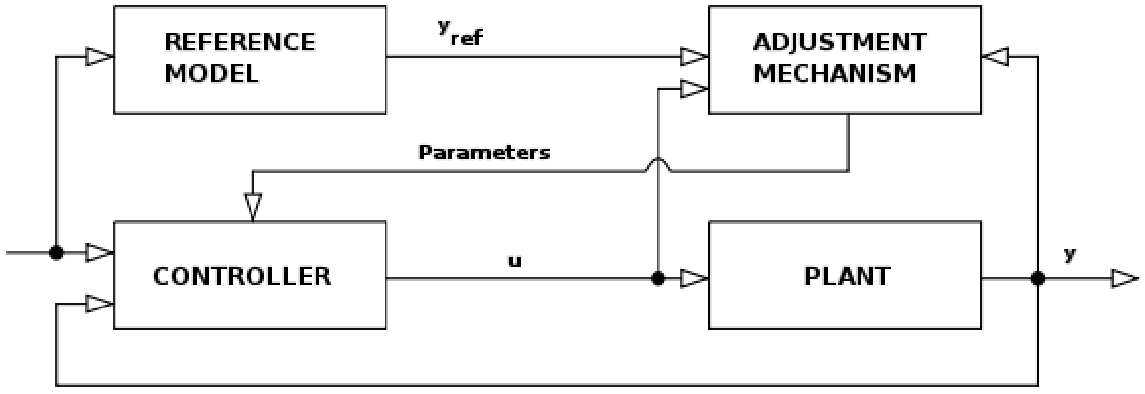
Classification of adaptive control techniques

In general one should distinguish between:

1. Feedforward Adaptive Control
2. Feedback Adaptive Control

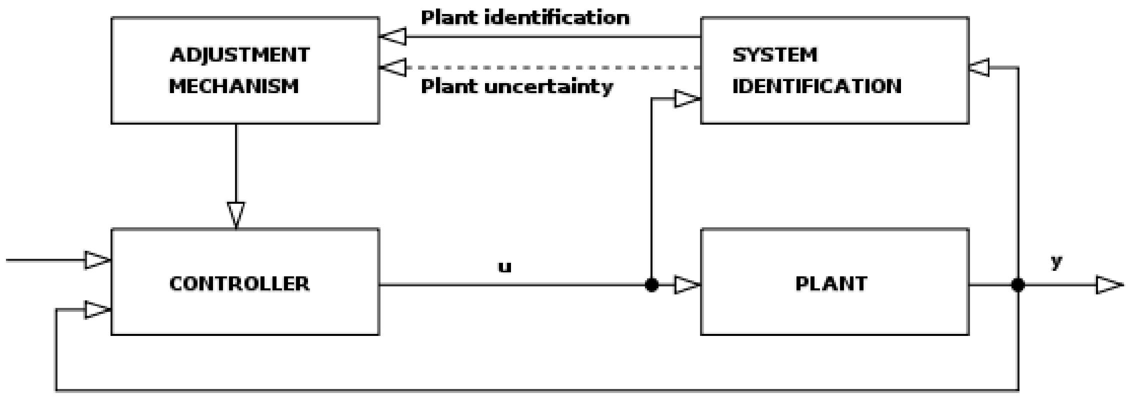
There are several broad categories of feedback adaptive control (classification can vary):

- Dual Adaptive Controllers [based on Dual control theory]
  - Optimal Dual Controllers [difficult to design]
  - Suboptimal Dual Controllers
- Nondual Adaptive Controllers
  - Gain scheduling
  - Model Reference Adaptive Controllers (MRACs) [incorporate a *reference model* defining desired closed loop performance]



MODEL REFERENCE ADAPTIVE CONTROL (MRAC)

MRAC



MODEL IDENTIFICATION ADAPTIVE CONTROL (MIAC)

MIAC

    - Gradient Optimization MRACs [use local rule for adjusting params when performance differs from reference. Ex.: "MIT rule".]
    - Stability Optimized MRACs
- Model Identification Adaptive Controllers (MIACs) [perform System identification while the system is running]
    - Cautious Adaptive Controllers [use current SI to modify control law, allowing for SI uncertainty]
    - Certainty Equivalent Adaptive Controllers [take current SI to be the true system, assume no uncertainty]
        - Nonparametric Adaptive Controllers
        - Parametric Adaptive Controllers
            - Explicit Parameter Adaptive Controllers
            - Implicit Parameter Adaptive Controllers

Some special topics in adaptive control can be introduced as well:

1. Adaptive Control Based on Discrete-Time Process Identification
2. Adaptive Control Based on the Model Reference Technique
3. Adaptive Control based on Continuous-Time Process Models
4. Adaptive Control of Multivariable Processes
5. Adaptive Control of Nonlinear Processes

Applications

When designing adaptive control systems, special consideration is necessary of convergence and robustness issues.

Typical applications of adaptive control are (in general):

- Self-tuning of subsequently fixed linear controllers during the implementation phase for one operating point;
- Self-tuning of subsequently fixed robust controllers during the implementation phase for whole range of operating points;
- Self-tuning of fixed controllers on request if the process behaviour changes due to ageing, drift, wear etc.;
- Adaptive control of linear controllers for nonlinear or time-varying processes;
- Adaptive control or self-tuning control of nonlinear controllers for nonlinear processes;
- Adaptive control or self-tuning control of multivariable controllers for multivariable processes (MIMO systems);

Usually these methods adapt the controllers to both the process statics and dynamics. In special cases the adaptation can be limited to the static behavior alone, leading to adaptive control based on characteristic curves for the steady-states or to extremum value control, optimizing the steady state. Hence, there are several ways to apply adaptive control algorithms.

System identification

In control engineering, the field of **system identification** uses statistical methods to build mathematical models of dynamical systems from measured data. System identification also includes the optimal design of experiments for efficiently generating informative data for fitting such models.

Overview

A dynamical mathematical model in this context is a mathematical description of the dynamic behavior of a system or process in either the time or frequency domain. Examples include:

- physical processes such as the movement of a falling body under the influence of gravity;
- economic processes such as stock markets that react to external influences.

One could build a so-called white-box model based on first principles, e.g. a model for a physical process from the Newton equations, but in many cases such models will be overly complex and possibly even impossible to obtain in reasonable time due to the complex nature of many systems and processes.

A much more common approach is therefore to start from measurements of the behavior of the system and the external influences (inputs to the system) and try to determine a mathematical relation between them without going into the details of what is actually happening inside the system. This approach is called system identification. Two types of models are common in the field of system identification:

- **grey box model:** although the peculiarities of what is going on inside the system are not entirely known, a certain model based on both insight into the system and experimental data is constructed. This model does however still have a number of unknown free parameters which can be estimated using system identification. One example, uses the monod saturation model for microbial growth. The model contains a simple hyperbolic relationship between substrate concentration and growth rate, but this can be justified by molecules binding to a substrate without going into detail on the types of molecules or types of binding. Grey box modeling is also known as semi-physical modeling.

- **black box model:** No prior model is available. Most system identification algorithms are of this type.

In the context of non-linear model identification Jin et al. describe greybox modeling as assuming a model structure a priori and then estimating the model parameters. This model structure can be specialized or more general so that it is applicable to a larger range of systems or devices. The parameter estimation is the tricky part and Jin et al. point out that the search for a good fit to experimental data tend to lead to an increasingly complex model. They then define a black-box model as a model which is very general and thus containing little a priori information on the problem at hand and at the same time being combined with an efficient method for parameter estimation. But as Nielsen and Madsen point out, the choice of parameter estimation can itself be problem-dependent.

Optimal design of experiments

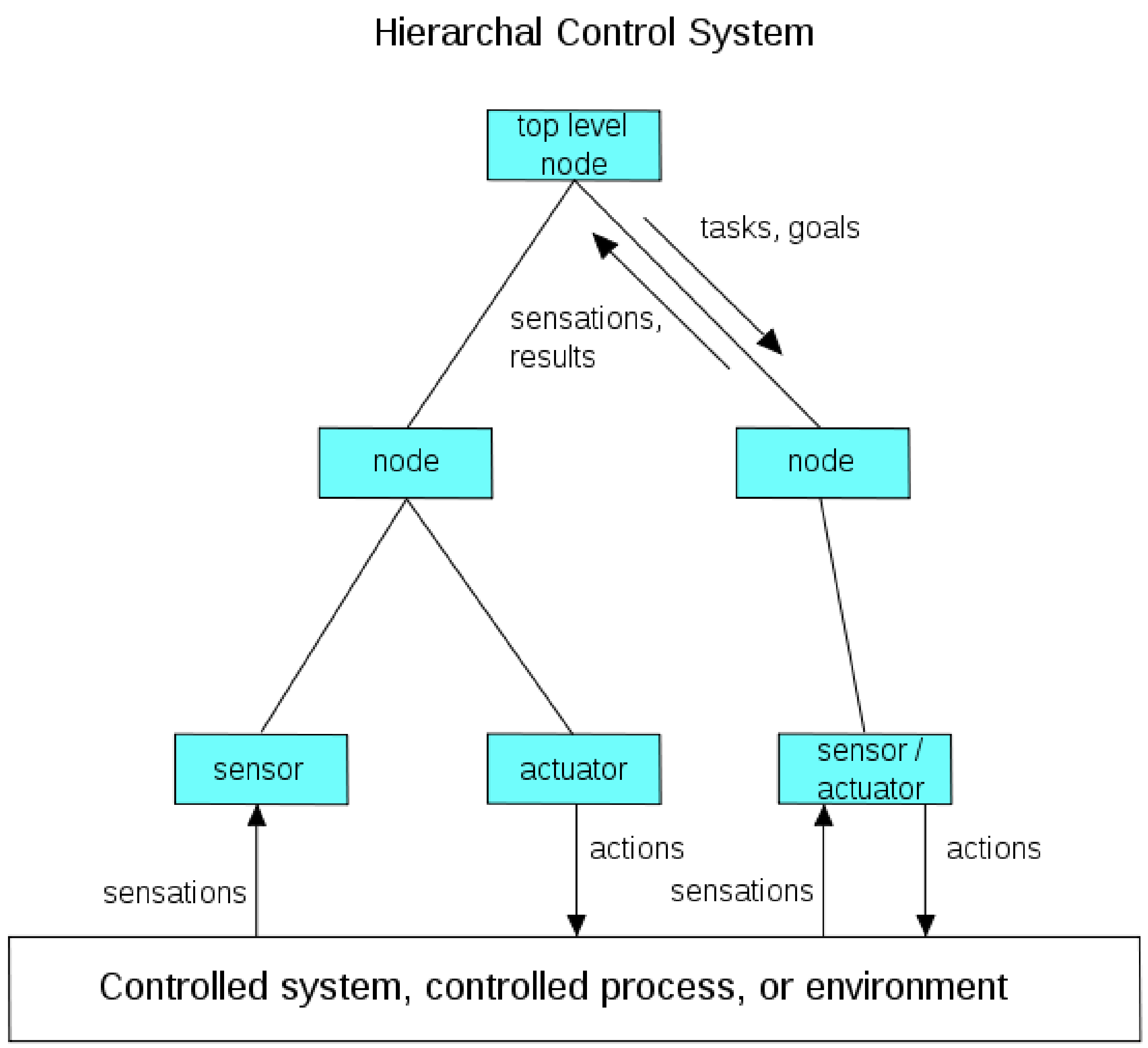
The quality of system identification depends on the quality of the inputs, which are under the control of the systems engineer. Therefore, systems engineers have long used the principles of the design of experiments. In recent decades, engineers have increasingly used the theory of optimal experimental design to specify inputs that yield maximally precise estimators.

Chapter- 8

Hierarchical Control System, Intelligent Control, Robust Control & Stochastic Control

Hierarchical control system

A **Hierarchical control system** is a form 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.



A hierarchical control system takes the shape of a tree in which each node operates independently, performing tasks from its superior node, commanding tasks of its subordinate nodes, sending abstracted sensations to its superior node, and receiving sensations from its subordinate nodes. Leaf nodes are sensors or actuators.

Overview

A human-built system with complex behavior is often organized as a hierarchy. For example a command hierarchy has among its notable features the organizational chart of superiors, subordinates, and lines of organizational communication. Hierarchical control systems are organized similarly to divide the decision making responsibility.

Each element of the hierarchy is a linked node in the tree. Commands, tasks and goals to be achieved flow down the tree from superior nodes to subordinate nodes, whereas sensations and command results flow up the tree from subordinate to superior nodes. Nodes may also exchange messages with their siblings. The two distinguishing features of a hierarchical control system are related to its layers.

- Each higher layer of the tree operates with a longer interval of planning and execution time than its immediately lower layer.
- The lower layers have local tasks, goals, and sensations, and their activities are planned and coordinated by higher layers which do not generally override their decisions. The layers form a hybrid intelligent system in which the lowest, reactive layers are sub-symbolic. The higher layers, having relaxed time constraints, are capable of reasoning from an abstract world model and performing planning. A hierarchical task network is a good fit for planning in a hierarchical control system.

Besides artificial systems, an animal's control systems are proposed to be organized as a hierarchy. In perceptual control theory, which postulates that an organism's behavior is a means of controlling its perceptions, the organism's control systems are suggested to be organized in a hierarchical pattern as their perceptions are constructed so.

Applications

Manufacturing, robotics and vehicles

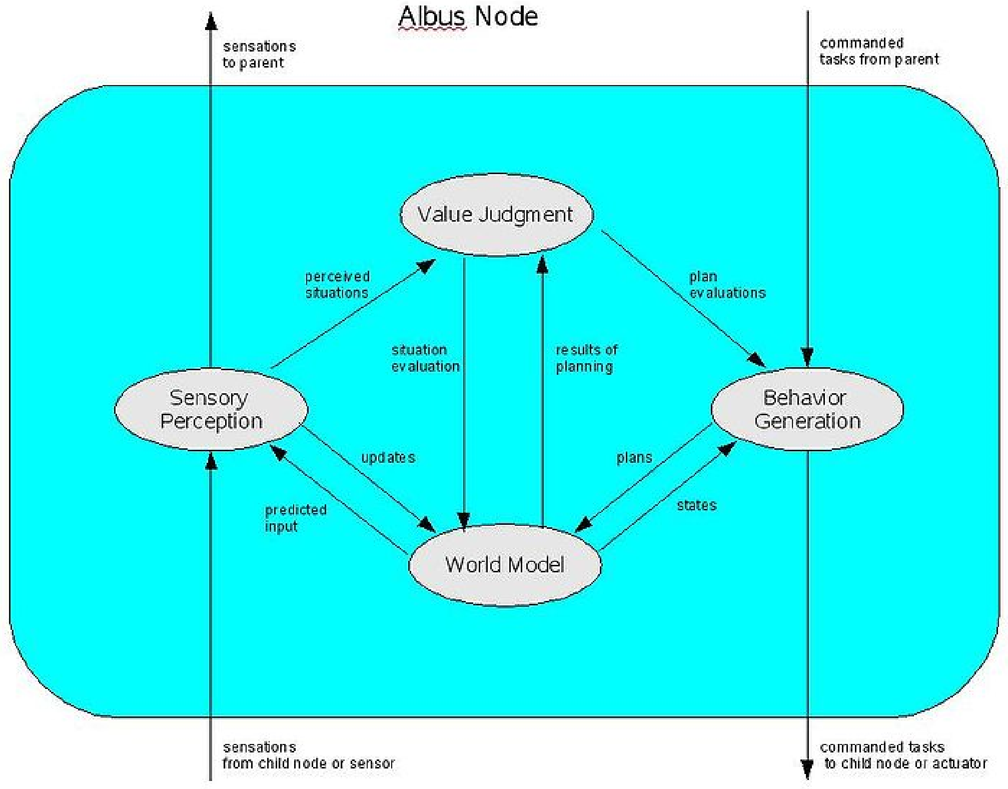
Among the robotic paradigms is the hierarchical paradigm in which a robot operates in a top-down fashion, heavy on planning, especially motion planning. Computer-aided production engineering has been a research focus at NIST since the 1980s. Its Automated Manufacturing Research Facility was used to develop a five layer production control model. In the early 1990s DARPA sponsored research to develop distributed (i.e. networked) intelligent control systems for applications such as military command and control systems. NIST built on earlier research to develop its Real-Time Control System (RCS) and Real-time Control System Software which is a generic hierarchical control system that has been used to operate a manufacturing cell, a robot crane, and an automated vehicle.

In November 2007, DARPA held the Urban Challenge. The winning entry, Tartan Racing employed a hierarchical control system, with layered mission planning, motion planning, behavior generation, perception, world modelling, and mechatronics.

Artificial intelligence

Subsumption architecture is a methodology for developing artificial intelligence that is heavily associated with behavior based robotics. This architecture is a way of decomposing complicated intelligent behavior into many "simple" behavior modules, which are in turn organized into layers. Each layer implements a particular goal of the software agent (i.e. system as a whole), and higher layers are increasingly more abstract. Each layer's goal subsumes that of the underlying layers, e.g. the decision to move forward by the eat-food layer takes into account the decision of the lowest obstacle-avoidance layer. Behavior need not be planned by a superior layer, rather behaviors may be triggered by sensory inputs and so are only active under circumstances where they might be appropriate.

Reinforcement learning has been used to acquire behavior in a hierarchical control system in which each node can learn to improve its behavior with experience.



Constituents in a node from James Albus's Reference Model Architecture

James Albus, while at NIST, developed a theory for intelligent system design named the Reference Model Architecture (RMA), which is a hierarchical control system inspired by RCS. Albus defines each node to contain these components.

- *Behavior generation* is responsible for executing tasks received from the superior, parent node. It also plans for, and issues tasks to, the subordinate nodes.
- *Sensory perception* is responsible for receiving sensations from the subordinate nodes, then grouping, filtering, and otherwise processing them into higher level abstractions that update the local state and which form sensations that are sent to the superior node.
- *Value judgment* is responsible for evaluating the updated situation and evaluating alternative plans.
- *World Model* is the local state that provides a model for the controlled system, controlled process, or environment at the abstraction level of the subordinate nodes.

At its lowest levels, the RMA can be implemented as a subsumption architecture, in which the world model is mapped directly to the controlled process or real world, avoiding the need for a mathematical abstraction, and in which time-constrained reactive planning can be implemented as a finite state machine. Higher levels of the RMA however, may have sophisticated mathematical world models and behavior implemented by automated planning and scheduling. Planning is required when certain behaviors cannot be triggered by current sensations, but rather by predicted or anticipated sensations, especially those that come about as result of the node's actions.

Intelligent control

Intelligent control is a class of control techniques, that use various AI computing approaches like neural networks, Bayesian probability, fuzzy logic, machine learning, evolutionary computation and genetic algorithms.

Overview

Intelligent control can be divided into the following major sub-domains:

- Neural network control
- Bayesian control
- Fuzzy (logic) control
- Neuro-fuzzy control
- Expert Systems
- Genetic control
- Intelligent agents (Cognitive/Conscious control)

New control techniques are created continuously as new models of intelligent behavior are created and computational methods developed to support them.

Neural network controllers

Neural networks have been used to solve problems in almost all spheres of science and technology. Neural network control basically involves two steps:

- System identification
- Control

It has been shown that a feedforward network with nonlinear, continuous and differentiable activation functions have universal approximation capability. Recurrent networks have also been used for system identification. Given, a set of input-output data pairs, system identification aims to form a mapping among these data pairs. Such a network is supposed to capture the dynamics of a system.

Bayesian controllers

Bayesian probability has produced a number of algorithms that are in common use in many advanced control systems, serving as state space estimators of some variables that are used in the controller.

The Kalman filter and the Particle filter are two examples of popular Bayesian control components. The Bayesian approach to controller design requires often an important effort in deriving the so-called system model and measurement model, which are the mathematical relationships linking the state variables to the sensor measurements available in the controlled system. In this respect, it is very closely linked to the system-theoretic approach to control design.

Robust control

Robust control is a branch of control theory that explicitly deals with uncertainty in its approach to controller design. Robust control methods are designed to function properly so long as uncertain parameters or disturbances are within some (typically compact) set. Robust methods aim to achieve robust performance and/or stability in the presence of bounded modelling errors.

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 , prompting research to improve them. This was the start of the theory of Robust Control, which took shape in the 1980s and 1990s and is still active today.

In contrast with an adaptive control policy, a robust control policy is static; rather than adapting to measurements of variations, the controller is designed to work assuming that certain variables will be unknown but, for example, bounded.

When is a control method said to be robust?

Informally, a controller designed for a particular set of parameters is said to be robust if it would also work well under a different set of assumptions. High-gain feedback is a simple example of a robust control method; with sufficiently high gain, the effect of any parameter variations will be negligible. High-gain feedback is the principle that allows simplified models of operational amplifiers and emitter-degenerated bipolar transistors to be used in a variety of different settings. This idea was already well understood by Bode and Black in 1927.

The modern theory of robust control

The theory of robust control began in the late 1970s and early 1980s and soon developed a number of techniques for dealing with bounded system uncertainty. .

Probably the most important example of a robust control technique is H-infinity loop-shaping, which was developed by Duncan McFarlane and Keith Glover of Cambridge University; this method minimizes the sensitivity of a system over its frequency spectrum, and this guarantees that the system will not greatly deviate from expected trajectories when disturbances enter the system.

Another example is LQG/LTR, which was developed to overcome the robustness problems of LQG control.

Stochastic control

Stochastic control is a subfield of control theory which deals with the existence of uncertainty in the data. The designer assumes, in a Bayesian probability-driven fashion, that a random noise with known probability distribution affects the state evolution and the observation of the controllers. Stochastic control aims to design the optimal controller that performs the desired control task with minimum average cost despite the presence of these noises.

An extremely well-studied formulation in stochastic control is that of linear-quadratic-Gaussian problem. Here the model is linear, and the objective function is the expected value of a quadratic form, and the additive disturbances are distributed in a Gaussian manner. A basic result for discrete time centralized systems is the **certainty equivalence property**: that the optimal control solution in this case is the same as would be obtained in the absence of the additive disturbances. This property is applicable to all systems that are merely linear and quadratic (LQ), and the Gaussian assumption allows for the optimal control laws, that are based on the certainty-equivalence property, to be linear functions of the observations of the controllers.

This property fails to hold for decentralized control, as was demonstrated by Witsenhausen in the celebrated Witsenhausen's counterexample.

Any deviation from the above assumptions—a nonlinear state equation, a non-quadratic objective function, or noise in the multiplicative parameters of the model—would cause the certainty equivalence property not to hold. In the discrete-time case with uncertainty about the parameter values in the transition matrix and/or the control response matrix of the state equation, but still with a linear state equation and quadratic objective function, a matrix Riccati equation can still be obtained for iterating to each period's solution. The discrete-time case of a non-quadratic loss function but only additive disturbances can also be handled, albeit with more complications.

Chapter- 9

Fuzzy Control System

A **fuzzy control system** is a control system based on fuzzy logic—a mathematical system that analyzes analog input values in terms of logical variables that take on continuous values between 0 and 1, in contrast to classical or digital logic, which operates on discrete values of either 0 or 1 (true or false).

Overview

Fuzzy logic is widely used in machine control. The term itself inspires a certain skepticism, sounding equivalent to "half-baked logic" or "bogus logic", but the "fuzzy" part does not refer to a lack of rigour in the method, rather to the fact that the logic involved can deal with fuzzy concepts—concepts that cannot be expressed as "true" or "false" but rather as "partially true". Although genetic algorithms and neural networks can perform just as well as fuzzy logic in many cases, fuzzy logic has the advantage that the solution to the problem can be cast in terms that human operators can understand, so that their experience can be used in the design of the controller. This makes it easier to mechanize tasks that are already successfully performed by humans.

History and applications

Fuzzy logic was first proposed by Lotfi A. Zadeh of the University of California at Berkeley in a 1965 paper. He elaborated on his ideas in a 1973 paper that introduced the concept of "linguistic variables", which in this article equates to a variable defined as a fuzzy set. Other research followed, with the first industrial application, a cement kiln built in Denmark, coming on line in 1975.

Fuzzy systems were largely ignored in the U.S. because they were associated with artificial intelligence, a field that periodically oversells itself, especially in the mid-1980s, resulting in a lack of credibility within the commercial domain.

The Japanese did not have this prejudice. Interest in fuzzy systems was sparked by Seiji Yasunobu and Soji Miyamoto of Hitachi, who in 1985 provided simulations that demonstrated the superiority of fuzzy control systems for the Sendai railway. Their ideas were adopted, and fuzzy systems were used to control accelerating, braking, and stopping when the line opened in 1987.

Another event in 1987 helped promote interest in fuzzy systems. During an international meeting of fuzzy researchers in Tokyo that year, Takeshi Yamakawa demonstrated the use of fuzzy control, through a set of simple dedicated fuzzy logic chips, in an "inverted pendulum" experiment. This is a classic control problem, in which a vehicle tries to keep a pole mounted on its top by a hinge upright by moving back and forth.

Observers were impressed with this demonstration, as well as later experiments by Yamakawa in which he mounted a wine glass containing water or even a live mouse to the top of the pendulum. The system maintained stability in both cases. Yamakawa eventually went on to organize his own fuzzy-systems research lab to help exploit his patents in the field.

Following such demonstrations, Japanese engineers developed a wide range of fuzzy systems for both industrial and consumer applications. In 1988 Japan established the Laboratory for International Fuzzy Engineering (LIFE), a cooperative arrangement between 48 companies to pursue fuzzy research.

Japanese consumer goods often incorporate fuzzy systems. Matsushita vacuum cleaners use microcontrollers running fuzzy algorithms to interrogate dust sensors and adjust suction power accordingly. Hitachi washing machines use fuzzy controllers to load-weight, fabric-mix, and dirt sensors and automatically set the wash cycle for the best use of power, water, and detergent.

As a more specific example, Canon developed an autofocusing camera that uses a charge-coupled device (CCD) to measure the clarity of the image in six regions of its field of view and use the information provided to determine if the image is in focus. It also tracks the rate of change of lens movement during focusing, and controls its speed to prevent overshoot.

The camera's fuzzy control system uses 12 inputs: 6 to obtain the current clarity data provided by the CCD and 6 to measure the rate of change of lens movement. The output is the position of the lens. The fuzzy control system uses 13 rules and requires 1.1 kilobytes of memory.

As another example of a practical system, an industrial air conditioner designed by Mitsubishi uses 25 heating rules and 25 cooling rules. A temperature sensor provides input, with control outputs fed to an inverter, a compressor valve, and a fan motor. Compared to the previous design, the fuzzy controller heats and cools five times faster, reduces power consumption by 24%, increases temperature stability by a factor of two, and uses fewer sensors.

The enthusiasm of the Japanese for fuzzy logic is reflected in the wide range of other applications they have investigated or implemented: character and handwriting recognition; optical fuzzy systems; robots, including one for making Japanese flower arrangements; voice-controlled robot helicopters, this being no mean feat, as hovering is a "balancing act" rather similar to the inverted pendulum problem; control of flow of powders in film manufacture; elevator systems; and so on.

Work on fuzzy systems is also proceeding in the US and Europe, though not with the same enthusiasm shown in Japan. The US Environmental Protection Agency has investigated fuzzy control for energy-efficient motors, and NASA has studied fuzzy control for automated space docking: simulations show that a fuzzy control system can greatly reduce fuel consumption. Firms such as Boeing, General Motors, Allen-Bradley, Chrysler, Eaton, and Whirlpool have worked on fuzzy logic for use in low-power refrigerators, improved automotive transmissions, and energy-efficient electric motors.

In 1995 Maytag introduced an "intelligent" dishwasher based on a fuzzy controller and a "one-stop sensing module" that combines a thermistor, for temperature measurement; a conductivity sensor, to measure detergent level from the ions present in the wash; a turbidity sensor that measures scattered and transmitted light to measure the soiling of the wash; and a magnetostrictive sensor to read spin rate. The system determines the optimum wash cycle for any load to obtain the best results with the least amount of energy, detergent, and water. It even adjusts for dried-on foods by tracking the last time the door was opened, and estimates the number of dishes by the number of times the door was opened.

Research and development is also continuing on fuzzy applications in software, as opposed to firmware, design, including fuzzy expert systems and integration of fuzzy logic with neural-network and so-called adaptive "genetic" software systems, with the ultimate goal of building "self-learning" fuzzy control systems.

Fuzzy sets

The input variables in a fuzzy control system are in general mapped into by sets of membership functions similar to this, known as "fuzzy sets". The process of converting a crisp input value to a fuzzy value is called "fuzzification".

A control system may also have various types of switch, or "ON-OFF", inputs along with its analog inputs, and such switch inputs of course will always have a truth value equal to either 1 or 0, but the scheme can deal with them as simplified fuzzy functions that happen to be either one value or another.

Given "mappings" of input variables into membership functions and truth values, the microcontroller then makes decisions for what action to take based on a set of "rules", each of the form:

```
  IF brake temperature IS warm AND speed IS not very fast
  THEN brake pressure IS slightly decreased.
```

In this example, the two input variables are "brake temperature" and "speed" that have values defined as fuzzy sets. The output variable, "brake pressure", is also defined by a fuzzy set that can have values like "static", "slightly increased", "slightly decreased", and so on.

This rule by itself is very puzzling since it looks like it could be used without bothering with fuzzy logic, but remember that the decision is based on a set of rules:

- All the rules that apply are invoked, using the membership functions and truth values obtained from the inputs, to determine the result of the rule.
- This result in turn will be mapped into a membership function and truth value controlling the output variable.
- These results are combined to give a specific ("crisp") answer, the actual brake pressure, a procedure known as "defuzzification".

This combination of fuzzy operations and rule-based "inference" describes a "fuzzy expert system".

Traditional control systems are based on mathematical models in which the control system is described using one or more differential equations that define the system response to its inputs. Such systems are often implemented as "PID controllers" (proportional-integral-derivative controllers). They are the products of decades of development and theoretical analysis, and are highly effective.

If PID and other traditional control systems are so well-developed, why bother with fuzzy control? It has some advantages. In many cases, the mathematical model of the control process may not exist, or may be too "expensive" in terms of computer processing power and memory, and a system based on empirical rules may be more effective.

Furthermore, fuzzy logic is well suited to low-cost implementations based on cheap sensors, low-resolution analog-to-digital converters, and 4-bit or 8-bit one-chip microcontroller chips. Such systems can be easily upgraded by adding new rules to improve performance or add new features. In many cases, fuzzy control can be used to improve existing traditional controller systems by adding an extra layer of intelligence to the current control method.

Fuzzy control in detail

Fuzzy controllers are very simple conceptually. They consist of an input stage, a processing stage, and an output stage. The input stage maps sensor or other inputs, such as switches, thumbwheels, and so on, to the appropriate membership functions and truth values. The processing stage invokes each appropriate rule and generates a result for each, then combines the results of the rules. Finally, the output stage converts the combined result back into a specific control output value.

The most common shape of membership functions is triangular, although trapezoidal and bell curves are also used, but the shape is generally less important than the number of curves and their placement. From three to seven curves are generally appropriate to cover the required range of an input value, or the "universe of discourse" in fuzzy jargon.

As discussed earlier, the processing stage is based on a collection of logic rules in the form of IF-THEN statements, where the IF part is called the "antecedent" and the THEN part is called the "consequent". Typical fuzzy control systems have dozens of rules.

Consider a rule for a thermostat:

```
  IF (temperature is "cold") THEN (heater is "high")
```

This rule uses the truth value of the "temperature" input, which is some truth value of "cold", to generate a result in the fuzzy set for the "heater" output, which is some value of "high". This result is used with the results of other rules to finally generate the crisp composite output. Obviously, the greater the truth value of "cold", the higher the truth value of "high", though this does not necessarily mean that the output itself will be set to "high", since this is only one rule among many. In some cases, the membership functions can be modified by "hedges" that are equivalent to adjectives. Common hedges include "about", "near", "close to", "approximately", "very", "slightly", "too", "extremely", and "somewhat". These operations may have precise definitions, though the definitions can vary considerably between different implementations. "Very", for one example, squares membership functions; since the membership values are always less than 1, this narrows the membership function. "Extremely" cubes the values to give greater narrowing, while "somewhat" broadens the function by taking the square root.

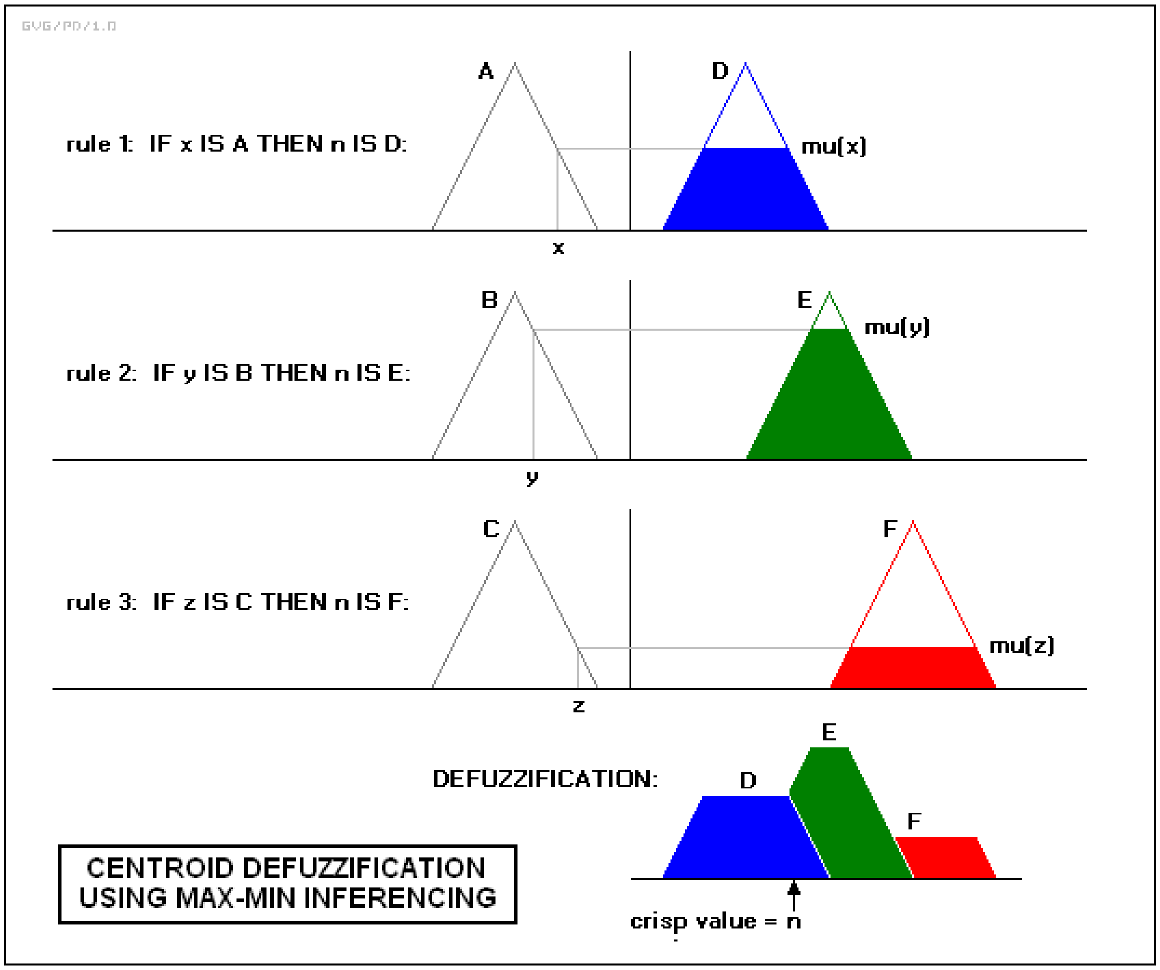
In practice, the fuzzy rule sets usually have several antecedents that are combined using fuzzy operators, such as AND, OR, and NOT, though again the definitions tend to vary: AND, in one popular definition, simply uses the minimum weight of all the antecedents, while OR uses the maximum value. There is also a NOT operator that subtracts a membership function from 1 to give the "complementary" function.

There are several different ways to define the result of a rule, but one of the most common and simplest is the "max-min" inference method, in which the output membership function is given the truth value generated by the premise.

Rules can be solved in parallel in hardware, or sequentially in software. The results of all the rules that have fired are "defuzzified" to a crisp value by one of several methods. There are dozens in theory, each with various advantages and drawbacks.

The "centroid" method is very popular, in which the "center of mass" of the result provides the crisp value. Another approach is the "height" method, which takes the value of the biggest contributor. The centroid method favors the rule with the output of greatest area, while the height method obviously favors the rule with the greatest output value.

The diagram below demonstrates max-min inferencing and centroid defuzzification for a system with input variables "x", "y", and "z" and an output variable "n". Note that "mu" is standard fuzzy-logic nomenclature for "truth value":


GVG/PD/1.0
rule 1: IF x IS A THEN n IS D:
A
D
mu(x)
x
rule 2: IF y IS B THEN n IS E:
B
E
mu(y)
y
rule 3: IF z IS C THEN n IS F:
C
F
mu(z)
z
DEFUZZIFICATION:
D
E
F
CENTROID DEFUZZIFICATION USING MAX-MIN INFERENCING
crisp value = n

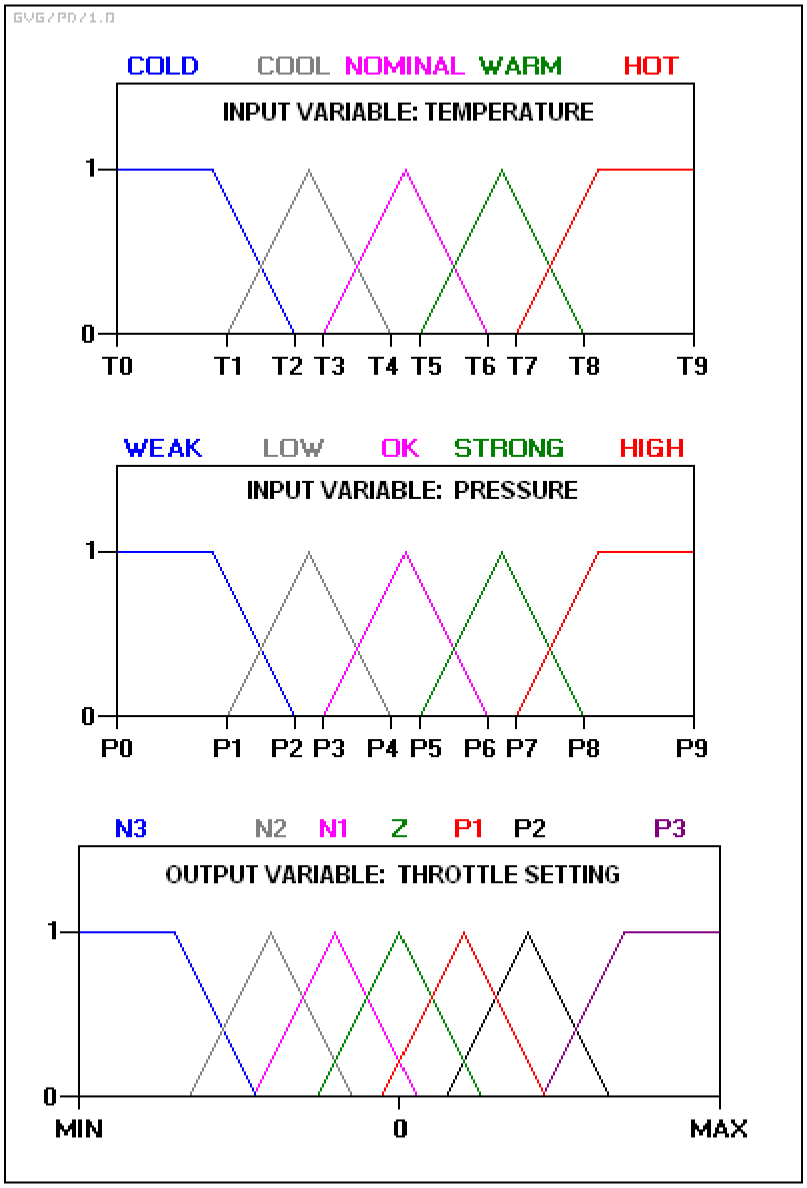
Notice how each rule provides a result as a truth value of a particular membership function for the output variable. In centroid defuzzification the values are OR'd, that is, the maximum value is used and values are not added, and the results are then combined using a centroid calculation.

Fuzzy control system design is based on empirical methods, basically a methodical approach to trial-and-error. The general process is as follows:

- Document the system's operational specifications and inputs and outputs.
- Document the fuzzy sets for the inputs.
- Document the rule set.
- Determine the defuzzification method.
- Run through test suite to validate system, adjust details as required.
- Complete document and release to production.

As a general example, consider the design of a fuzzy controller for a steam turbine. The block diagram of this control system appears as follows:

The input and output variables map into the following fuzzy set:



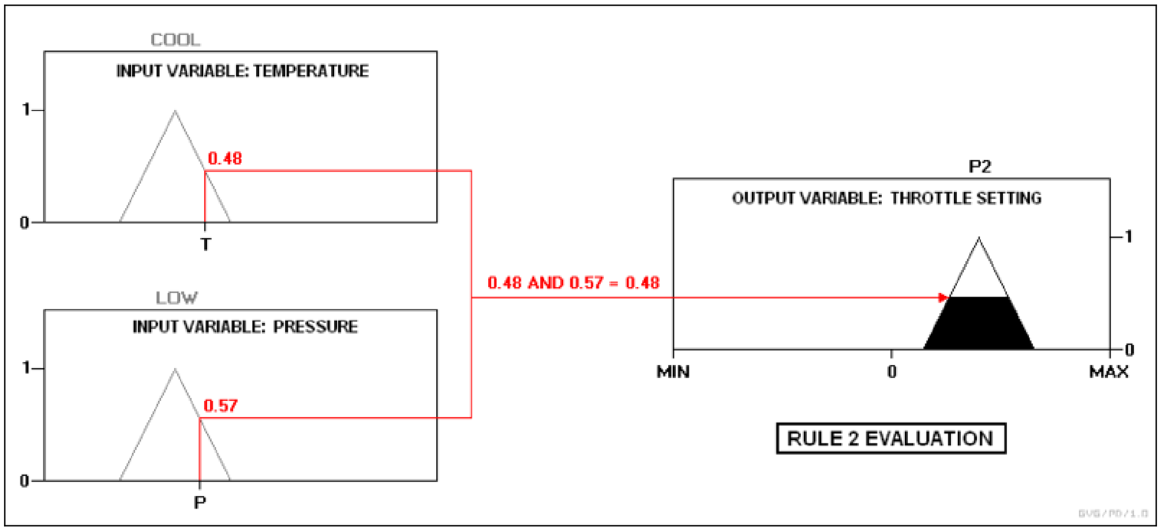
-- where:

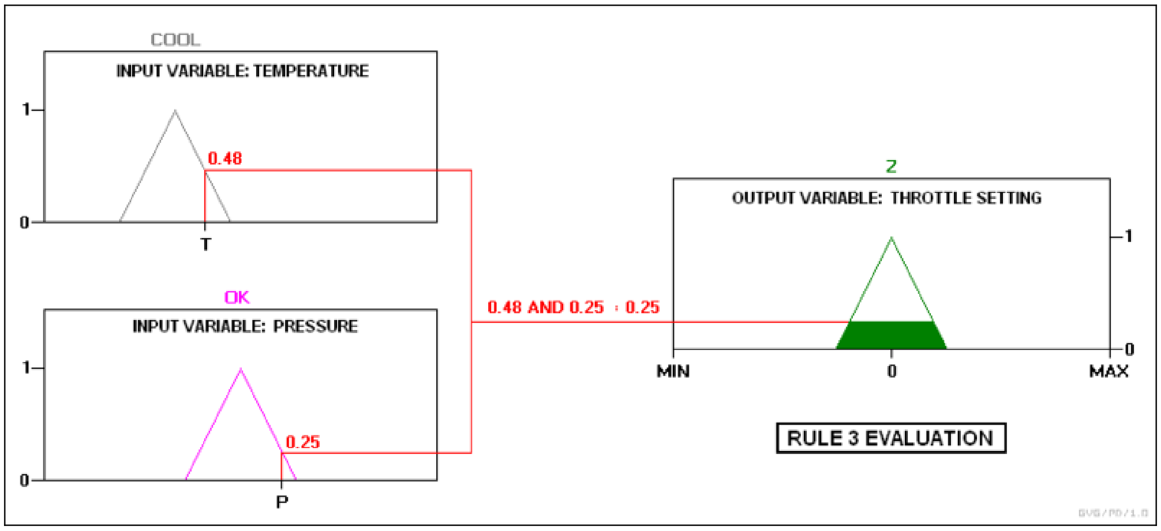
```
N3:    Large negative.
N2:    Medium negative.
N1:    Small negative.
Z:     Zero.
P1:    Small positive.
P2:    Medium positive.
P3:    Large positive.
```

The rule set includes such rules as:

```
rule 1:  IF temperature IS cool AND pressure IS weak,
         THEN throttle is P3.
rule 2:  IF temperature IS cool AND pressure IS low,
         THEN throttle is P2.
rule 3:  IF temperature IS cool AND pressure IS ok,
         THEN throttle is Z.
rule 4:  IF temperature IS cool AND pressure IS strong,
         THEN throttle is N2.
```

In practice, the controller accepts the inputs and maps them into their membership functions and truth values. These mappings are then fed into the rules. If the rule specifies an AND relationship between the mappings of the two input variables, as the examples above do, the minimum of the two is used as the combined truth value; if an OR is specified, the maximum is used. The appropriate output state is selected and assigned a membership value at the truth level of the premise. The truth values are then defuzzified. For an example, assume the temperature is in the "cool" state, and the pressure is in the "low" and "ok" states. The pressure values ensure that only rules 2 and 3 fire:





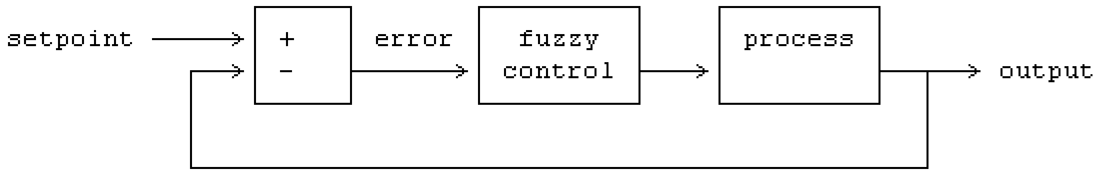
The two outputs are then defuzzified through centroid defuzzification:

```
                                    |          Z      P2
                                 1 -+          *       *
                                    |         * *     * *
                                    |        *   *   *   *
                                    |       *     * *     *
                                    |      *       222222222
                                    |     *       22222222222
                                    |    33333333222222222222
                                    +---3333333322222222222222-->
                                                      ^
                                                    +150
```

The output value will adjust the throttle and then the control cycle will begin again to generate the next value.

Building a fuzzy controller

Consider implementing with a microcontroller chip a simple feedback controller:



A fuzzy set is defined for the input error variable "e", and the derived change in error, "delta", as well as the "output", as follows:

```
LP:  large positive
SP:  small positive
ZE:  zero
SN:  small negative
LN:  large negative
```

If the error ranges from -1 to +1, with the analog-to-digital converter used having a resolution of 0.25, then the input variable's fuzzy set (which, in this case, also applies to the output variable) can be described very simply as a table, with the error / delta / output values in the top row and the truth values for each membership function arranged in rows beneath:

```
              -1    -0.75  -0.5   -0.25    0     0.25   0.5    0.75
1
```

mu(LP)	0	0	0	0	0	0	0.3	0.7	1
mu(SP)	0	0	0	0	0.3	0.7	1	0.7	0.3
mu(ZE)	0	0	0.3	0.7	1	0.7	0.3	0	0
mu(SN)	0.3	0.7	1	0.7	0.3	0	0	0	0
mu(LN)	1	0.7	0.3	0	0	0	0	0	0

-- or, in graphical form (where each "X" has a value of 0.1):

```
         LN            SN            ZE            SP            LP
      +------------------------------------------------------------------
--+
      |
|
-1.0  |  XXXXXXXXXX    XXX           :             :             :
|
-0.75 |  XXXXXXX       XXXXXXX       :             :             :
|
-0.5  |  XXX           XXXXXXXXXX    XXX           :             :
|
-0.25 |  :             XXXXXXX       XXXXXXX       :             :
|
 0.0  |  :             XXX           XXXXXXXXXX    XXX           :
|
 0.25 |  :             :             XXXXXXX       XXXXXXX       :
|
 0.5  |  :             :             XXX           XXXXXXXXXX    XXX
|
 0.75 |  :             :             :             XXXXXXX       XXXXXXX
|
 1.0  |  :             :             :             XXX           XXXXXXXXXX
|
      |
|
      +------------------------------------------------------------------
--+
```

Suppose this fuzzy system has the following rule base:

```
  rule 1:  IF e = ZE AND delta = ZE THEN output = ZE
  rule 2:  IF e = ZE AND delta = SP THEN output = SN
  rule 3:  IF e = SN AND delta = SN THEN output = LP
  rule 4:  IF e = LP OR  delta = LP THEN output = LN
```

These rules are typical for control applications in that the antecedents consist of the logical combination of the error and error-delta signals, while the consequent is a control command output. The rule outputs can be defuzzified using a discrete centroid computation:

```
  SUM( I = 1 TO 4 OF ( mu(I) * output(I) ) ) / SUM( I = 1 TO 4 OF mu(I)
)
```

Now, suppose that at a given time we have:

```
  e     = 0.25
  delta = 0.5
```

Then this gives:

	e	delta
mu(LP)	0	0.3
mu(SP)	0.7	1
mu(ZE)	0.7	0.3
mu(SN)	0	0
mu(LN)	0	0

Plugging this into rule 1 gives:

```
  rule 1:  IF e = ZE AND delta = ZE THEN output = ZE

     mu(1)     = MIN( 0.7, 0.3 ) = 0.3
     output(1) = 0
```

-- where:

- mu(1): Truth value of the result membership function for rule 1. In terms of a centroid calculation, this is the "mass" of this result for this discrete case.
- output(1): Value (for rule 1) where the result membership function (ZE) is maximum over the output variable fuzzy set range. That is, in terms of a centroid calculation, the location of the "center of mass" for this individual result. This value is independent of the value of "mu". It simply identifies the location of ZE along the output range.

The other rules give:

```
rule 2:  IF e = ZE AND delta = SP THEN output = SN

   mu(2)     = MIN( 0.7, 1 ) = 0.7
   output(2) = -0.5
rule 3: IF e = SN AND delta = SN THEN output = LP

   mu(3)     = MIN( 0.0, 0.0 ) = 0
   output(3) = 1
rule 4: IF e = LP OR  delta = LP THEN output = LN

   mu(4)     = MAX( 0.0, 0.3 ) = 0.3
   output(4) = -1
```

The centroid computation yields:

$$\frac{mu(1).output(1) + mu(2).output(2) + mu(3).output(3) + mu(4).output(4)}{mu(1) + mu(2) + mu(3) + mu(4)}$$

$$= \frac{(0.3 * 0) + (0.7 * -0.5) + (0 * 1) + (0.3 * -1)}{0.3 + 0.7 + 0 + 0.3}$$

```
  =  - 0.5
```

-- for the final control output. Simple. Of course the hard part is figuring out what rules actually work correctly in practice.

If you have problems figuring out the centroid equation, remember that a centroid is defined by summing all the moments (location times mass) around the center of gravity and equating the sum to zero. So if X_0 is the center of gravity, X_i is the location of each mass, and M_i is each mass, this gives:

$$0 = (X_1 - X_0) * M_1 + (X_2 - X_0) * M_2 + \ldots + (X_n - X_0) * M_n$$

$$0 = (X_1 * M_1 + X_2 * M_2 + \ldots + X_n * M_n) - X_0 * (M_1 + M_2 + \ldots + M_n)$$

$$X_0 * (M_1 + M_2 + \ldots + M_n) = X_1 * M_1 + X_2 * M_2 + \ldots + X_n * M_n$$

$$X_0 = \frac{X_1 * M_1 + X_2 * M_2 + \ldots + X_n * M_n}{M_1 + M_2 + \ldots + M_n}$$

In our example, the values of mu correspond to the masses, and the values of X to location of the masses (mu, however, only 'corresponds to the masses' if the initial 'mass' of the output functions are all the same/equivalent. If they are not the same, ie some are narrow triangles, while others maybe wide trapizoids or shouldered triangles, then the mass or area of the output function must be known or calculated. It is this mass that is then scaled by mu and multiplied by it's location X_i).

This system can be implemented on a standard microprocessor, but dedicated fuzzy chips are now available. For example, Adaptive Logic INC of San Jose, California, sells a "fuzzy chip", the AL220, that can accept four analog inputs and generate four analog outputs. A block diagram of the chip is shown below:

```
              +---------+                              +-------+
 analog --4-->| analog  |                              | mux / +--4-->
analog
   in         |   mux   |                              |  SH   |
out
              +----+----+                              +-------+
                   |                                        ^
                   V                                        |
            +-------------+                             +--+--+
            | ADC / latch |                             | DAC |
            +------+------+                             +-----+
                   |                                        ^
                   |                                        |
                   8         +------------------------------+
                   |         |                              |
                   |         V                              |
                   |   +-----------+      +-------------+   |
                   +-->| fuzzifier |      | defuzzifier +---+
                       +-----+-----+      +-------------+
                             |                   ^
                             |  +-------------+  |
                             |  |    rule     |  |
                             +->|  processor  +--+
                                | (50 rules)  |
                                +------+------+
                                       |
                                +------+------+
                                |  parameter  |
                                |    memory   |
                                |   256 x 8   |
                                +-------------+

    ADC:  analog-to-digital converter
    DAC:  digital-to-analog converter
    SH:   sample/hold
```

Antilock brakes

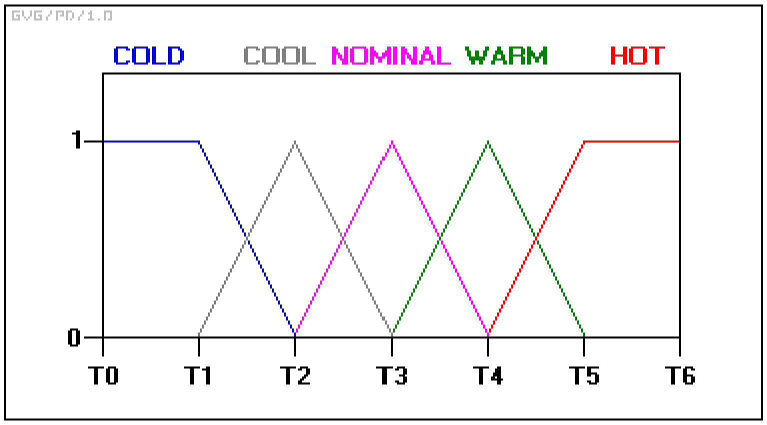
As a first example, consider an anti-lock braking system, directed by a microcontroller chip. The microcontroller has to make decisions based on brake temperature, speed, and other variables in the system.

The variable "temperature" in this system can be subdivided into a range of "states": "cold", "cool", "moderate", "warm", "hot", "very hot". The transition from one state to the next is hard to define.

An arbitrary static threshold might be set to divide "warm" from "hot". For example, at exactly 90 degrees, warm ends and hot begins. But this would result in a discontinuous change when the input value passed over that threshold. The transition wouldn't be smooth, as would be required in braking situations.

The way around this is to make the states *fuzzy*. That is, allow them to change gradually from one state to the next. In order to do this there must be a dynamic relationship established between different factors.

We start by defining the input temperature states using "membership functions":



With this scheme, the input variable's state no longer jumps abruptly from one state to the next. Instead, as the temperature changes, it loses value in one membership function while gaining value in the next. In other words, its ranking in the category of cold decreases as it becomes more highly ranked in the warmer category.

At any sampled timeframe, the "truth value" of the brake temperature will almost always be in some degree part of two membership functions: i.e.: '0.6 nominal and 0.4 warm', or '0.7 nominal and 0.3 cool', and so on.

The above example demonstrates a simple application, using the abstraction of values from multiple values. This only represents one kind of data, however, in this case, temperature.

Adding additional sophistication to this braking system, could be done by additional factors such as traction, speed, inertia, set up in dynamic functions, according to the designed fuzzy system.

Logical interpretation of fuzzy control

In spite of the appearance there are several difficulties to give a rigorous logical interpretation of the *IF-THEN* rules. As an example, interpret a rule as *IF (temperature is "cold") THEN (heater is "high")* by the first order formula *Cold(x)→High(y)* and assume that r is an input such that *Cold(r)* is false. Then the formula *Cold(r)→High(t)* is true for any *t* and therefore any *t* gives a correct control given *r*. Obviously, if we consider systems of rules in which the class antecedent define a partition such a paradoxical phenomenon does not arise. In any case there is sometime of unsatisfactory in considering two variables *x* and *y* in a rule without some kind of functional dependence. A rigorous logical justification of fuzzy control is given in Hájek's book where fuzzy control is represented as a theory of Hájek's basic logic. Also in Gerla 2005 a logical approach to fuzzy control is proposed based on the following idea. Denote by *f* the fuzzy function associated with the fuzzy control system, i.e., given the input r, *s(y) = f(r,y)* is the fuzzy set of possible outputs. Then given a possible output 't', we interpret f(r,t) as the truth degree of the claim *"t is a good answer given r"*. More formally, any system of IF-THEN rules can be translate into a fuzzy program in such a way that the fuzzy function *f* is the interpretation of a vague predicate *Good(x,y)* in the associated least fuzzy Herbrand model. In such a way fuzzy control becomes a chapter of fuzzy logic programming. The learning process becomes a question belonging to inductive logic theory.