1

Introduction

Fuzzy control uses sentences, in the form of rules, to control a process. The controller can take many inputs, and the advantage of fuzzy control is the ability to include expert knowledge. The interface to the controller is more or less natural language, and that is what distinguishes fuzzy control from other control methods. It is generally a nonlinear controller. There are, however, very few design procedures in the nonlinear domain compared to the linear domain. This book proposes to stay as long as possible in the linear domain, on the solid foundations of linear control theory, before moving into the nonlinear domain with the design. The design method consists accordingly of four steps: design a PID controller, replace it with a linear fuzzy controller, make it nonlinear, and fine-tune the resulting controller. A nonlinear process may have several equilibrium points, and the local behaviour can be different from the behaviour far from an equilibrium, which makes it difficult to control. In order to demonstrate various aspects of nonlinear control, the book uses a simulator of a train car on a hilly track.

Fuzzy controllers appear in consumer products such as washing machines, video cameras, and cars. Industrial applications include cement kilns, underground trains, and robots. A *fuzzy controller* is an *automatic controller*, that is, a self-acting or self-regulating mechanism that controls an object in accordance with a desired behaviour. The object can be, for instance, a robot set to follow a certain path. A fuzzy controller acts or regulates by means of rules in a more or less natural language, based on the distinguishing feature: fuzzy logic. The rules are invented by plant operators or design engineers, and fuzzy control is thus a branch of artificial intelligence.

1.1 What Is Fuzzy Control?

Conventionally, computer programs make rigid *yes* or *no* decisions by means of decision rules based on two-valued logic: true/false, yes/no, or one/zero. An example is an air conditioner with a thermostatic controller that recognizes just two states: above the desired temperature or below the desired temperature. *Fuzzy logic*, on the other hand, allows intermediate truth-values between true and false.

Foundations of Fuzzy Control: A Practical Approach, Second Edition. Jan Jantzen.

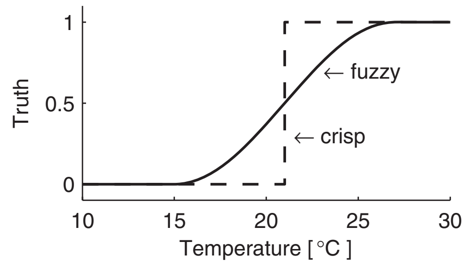

Figure 1.1 A warm room. The crisp air conditioner considers any temperature above 21°C warm. The fuzzy air conditioner considers gradually warmer temperatures. (figwarm.m)

A fuzzy air conditioner may thus recognize ‘warm’ and ‘cold’ room temperatures. The rules behind are less precise, for instance:

- *Rule*. If the room temperature is warm and slightly increasing, then increase the cooling.

Many classes or *sets* have *fuzzy* rather than sharp boundaries, and this is the mathematical basis of fuzzy logic: the set of ‘warm’ temperature measurements is one example of a fuzzy set.

The core of a fuzzy controller is a collection of *linguistic* (*verbal*) rules of the *if–then* form. Several variables may appear in each rule, both on the *if* side and on the *then* side. The rules can bring the reasoning used by computers closer to that of human beings.

In the example of the fuzzy air conditioner, the controller works on the basis of a temperature measurement. The room temperature is just a number, and more information is necessary to decide whether the room is warm. Therefore, the designer must incorporate a human being’s perception of warm room temperatures. A straightforward approach is to evaluate beforehand all possible temperature measurements. For example, on a scale from 0 to 1, truly *warm* corresponds to 1 and definitely *not warm* corresponds to 0,

Grades of *warm*	0.0	0.0	0.4	0.9	1.0
Temperature (°C)	10	15	20	25	30

This example uses *discrete* temperature measurements, whereas Figure 1.1 shows the same idea graphically, in the form of a *continuous* mapping of temperature measurements to truth-values. The mapping is arbitrary, that is, based on preference, not mathematical reason.

1.2 Why Fuzzy Control?

If PID control (proportional-integral-derivative control) is inadequate – for example, in the case of higher-order processes, systems with a long deadtime, or systems with oscillatory modes (Åström and Hägglund 2006) – fuzzy control is an option. But first, let us consider why one would not use a fuzzy controller:

- The PID controller is well understood, easy to implement – both in its digital and analogue forms – and it is widely used. By contrast, the fuzzy controller requires some knowledge of fuzzy logic. It also involves building arbitrary membership functions.

- The fuzzy controller is generally nonlinear. It does not have a simple equation like the PID, and it is more difficult to analyse mathematically; approximations are required, and it follows that stability is more difficult to guarantee.
- The fuzzy controller has more tuning parameters than the PID controller. Furthermore, it is difficult to trace the data flow during execution, which makes error correction more difficult.

On the other hand, fuzzy controllers are used in industry with success. There are several possible reasons:

- Since the control strategy consists of *if–then* rules, it is easy for a process operator to read. The rules can be built from a vocabulary containing everyday words such as ‘high’, ‘low’, and ‘increasing’. Process operators can embed their experience directly.
- The fuzzy controller can accommodate many inputs and many outputs. Variables can be combined in an *if–then* rule with the connectives *and* and *or*. Rules are executed in parallel, implying a recommended action from each. The recommendations may be in conflict, but the controller resolves conflicts.

Fuzzy logic enables non-specialists to design control systems, and this may be the main reason for its success.

1.3 Controller Design

Established design methods such as pole placement, optimal control, and frequency response shaping only apply to linear systems, whereas fuzzy control is generally nonlinear. Since our knowledge of the behaviour of nonlinear systems is limited, compared with the situation in the linear domain, this book is based on a design procedure founded on linear control:

1. Design a PID controller.
2. Replace it with a linear fuzzy controller.
3. Make it nonlinear.
4. Fine-tune it.

The idea is to exploit the design methods within PID control and carry them forward to fuzzy control. The design procedure is feasible because it is possible to build a linear fuzzy controller that functions exactly as any PID controller does. The following example introduces the design procedure.

1.4 Introductory Example: Stopping a Car

Assume that we are to design a controller that automatically stops a car in front of a red stop light, as a part of future safety equipment. Figure 1.2 illustrates the situation, and it defines the symbols for the brake force (F), the mass of the car (m), and its position (y). Assume also that we can only apply the brakes, not the accelerator pedal, in order to keep the example simple. Even though the example is simple, it is representative; think of parking a robot in a charging dock, parking a ferry at the quay, or stopping a driver-less metro train at a station.

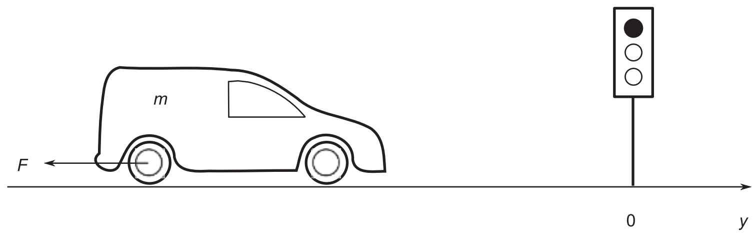

Figure 1.2 Stopping a car. The position y is positive towards the right, with zero at the stop light. The brakes act with a negative force F on the mass m.

Figure 1.3 shows a simulation model in Simulink (trademark of The MathWorks, Inc.). The block diagram includes a limiter block on the brake force, and the model is therefore nonlinear.

Step 1: Design a PID controller

Our first attempt is to try a proportional (P) controller,

$$F = K_p e \tag{1.1}$$

where K_p is the *proportional gain*, which can be adjusted to achieve the best response. The error $e \geq 0$ is the position error measured from the reference point *Ref* to the current position $y \leq 0$, that is,

$$e = Ref - y \tag{1.2}$$

Since y is negative and $Ref = 0$, then e is positive. But K_p is also positive, and the P controller in Equation (1.1) would demand a positive force F – in other words, acceleration by means of the accelerator pedal. The problem definition above ruled out the accelerator pedal, however, and we can conclude that a proportional controller is inadequate.

Our second attempt is to apply a proportional-derivative (PD) controller, since it includes a prediction. The controller is

$$F = K_p (e + T_d \dot{e}) \tag{1.3}$$

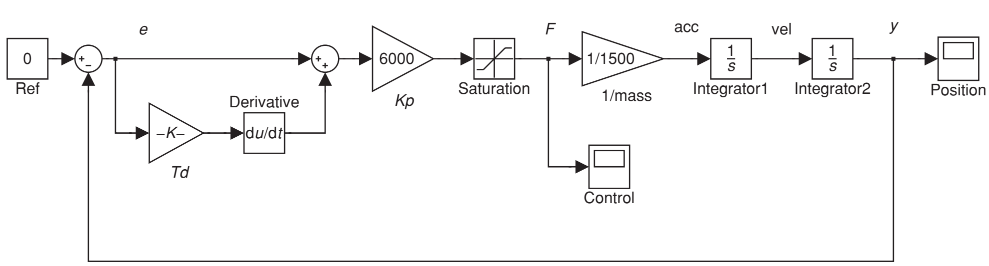

Figure 1.3 Simulink block diagram. A PD controller brakes the car from initial conditions $y(0) = -15$ and $\dot{y}(0) = 10$. (figcarpd.mdl)

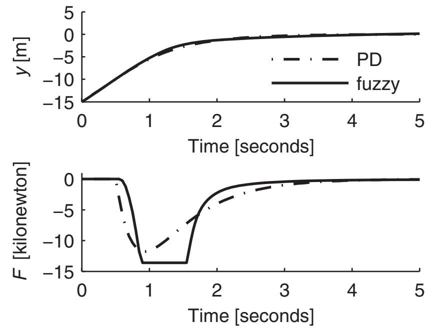

Figure 1.4 Stopping a car. Comparison between a PD controller and a fuzzy controller. (figstopcar.m)

where T_d is the *derivative gain*, which can be adjusted. Now the control signal is proportional to the term $e + T_d\dot{e}$ which is the predicted error T_d seconds ahead of the current error e. Compared to the P controller, the PD controller calls for extra brake force when the velocity is high. Figure 1.4 shows the response and the brake force with

$$K_p = 6000$$
$$T_d = 1$$

During the first 0.5 s, the control signal is zero. Thereafter the derivative action takes over and starts to brake the car. In other words, the controller waits 0.5 s until it kicks in, it quickly increases the braking force, and after about 1 s it relaxes the brake gently. It takes about 5 s to stop the car.

We *tuned* the gains K_p and T_d in order to achieve a good closed loop performance. Hand tuning is possible, but it generally requires patience and a good sense of how the system responds. It is easier to use rules, for example the *Ziegler–Nichols tuning rules*. Although the rules often result in less than optimal settings, they are a good starting point for a manual fine tuning.

Step 2: Replace it with a linear fuzzy controller

A fuzzy controller consists of *if–then* rules describing the action to be taken in various situations. We will consider the situations where the distance to the stop light is long or short, and situations where the car is approaching fast or slowly. The linguistic terms must be specified precisely for a computer to execute the rules.

The following chapters will show how to design a *linear* fuzzy controller, with a performance that is exactly the same as the PD controller in the previous step. It is a design aid, because the PD controller, with its tuning, settles many design choices for the fuzzy controller. One requirement is that the membership functions should be linear.

At the end of this step, we have a fuzzy controller, with a response (not shown) exactly as the PD response in Figure 1.4.

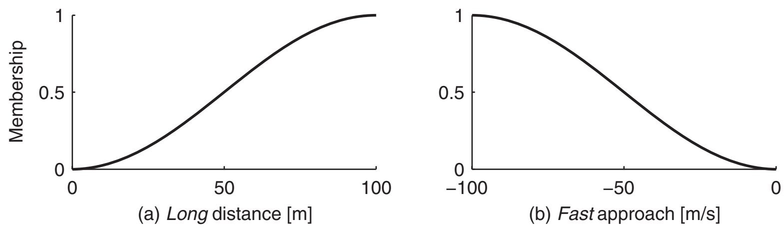

Figure 1.5 Fuzzy membership functions. Curve (a) is related to the distance from the stop light, and curve (b) specifies what is meant by a *fast* approach. The two curves are components of the rule: If distance is long and approach is fast, then brake zero. (figmfcar.m)

Step 3: Make it nonlinear

A complete rule base of all possible input combinations contains four rules:

If distance is long and approach is fast, then brake zero (1.4)

If distance is long and approach is slow, then brake zero (1.5)

If distance is short and approach is fast, then brake hard (1.6)

If distance is short and approach is slow, then brake zero (1.7)

The linguistic terms must be specified precisely for a computer to execute the rules. Figure 1.5 shows how to implement ‘long’, as in ‘distance is long’. It is a fuzzy *membership function*, shaped like the letter *s*. The horizontal axis is the *universe*, which is the interval [0, 100]% of the full range of 15 m. The vertical axis is the *membership grade*, that is, how compatible a distance measurement is with our perception of the term ‘long’. For instance, a distance of 15 m (100%) has membership 1, because the distance is definitely long, while half that distance is long to a degree of just 0.5. Note that the horizontal axis corresponds to the previously defined error e, scaled onto a standard range relative to the maximum distance.

The term ‘fast’, as in ‘approach is fast’, is another membership function. The horizontal axis is again percentages of full range (10 m/s), but the numbers are negative to emphasize that the distance is decreasing rather than increasing. The horizontal axis corresponds to the previously defined time derivative $\dot{e}$ scaled onto the universe. The -100% corresponds to the maximum speed of 10 m/s. Similarly, the membership function for ‘short’ is just a mirror image of the membership function ‘long’, and the membership function ‘slow’ is just a mirror image of ‘fast’.

Turning to the *then*-side of the rules, the term ‘zero’ means to apply the brake force $F = 0$. The term ‘hard’ is the full brake force of -100%.

The nonlinear domain is poorly understood in general, and it usually calls for a trial and error design procedure. Nevertheless, the following chapters provide methods such that at least some *analysis* is possible.

Step 4: Fine-tune it

Figure 1.4 shows the response with the nonlinear controller, together with the initial PD response, after adjusting one tuning factor (input gain on the error, GE). The response is close to the initial PD response, but a little faster. The lower plot with the control signals shows the difference: the fuzzy controller waits longer before it kicks in, then it uses all the available brake force, and thereafter it releases the brake quicker than the PD controller.

The behaviour is not necessarily better than PD control. But since the fuzzy controller in step 2 is guaranteed to perform the same way, it is safe to say that the fuzzy controller is at least as good. Whether it performs better after steps 3 and 4 is an open question, but at least the fuzzy controller provides extra options to shape the control signal. This could be important if passenger comfort has a high priority.

Example 1.1 *Tuning by means of process knowledge*

Is it possible to use a mathematical model to find optimal settings for the PD controller?

► *Solution*

Disregarding engine dynamics, skidding, slip, and friction – other than the frictional forces in the brake pads – the force F causes an acceleration a according to Newton's second law of motion $F = ma$. Acceleration is the derivative of the velocity which in turn is the derivative of the position. Thus $a = \ddot{y}$, where the dots are Newton's dot notation for the differentiation operator $\mathrm{d}/\mathrm{d}t$. We can rewrite the differential equation that governs the motion of the car as

$$F = m\ddot{y} \Leftrightarrow \ddot{y} = \frac{F}{m} \tag{1.8}$$

For a Volkswagen Caddy Van (diesel, 2-L engine) the mass, without load and including the driver, is approximately 1500 kg*. Assume that the stop light changes to red when the car is* 15 m *(*49 ft*) away at a speed of* 10 m/s *(*36 km/h *or* 23 mph*). We have thus identified the following constants:*

$$\begin{aligned} m &= 1500 \\ y(0) &= -15 \\ \dot{y}(0) &= 10 \end{aligned}$$

Here $y(0)$ means the initial position, that is $y(t)$ at time $t = 0$, and $\dot{y}(0)$ is the initial speed. The force F arises not from the engine, but from an opposite friction force in the brakes, and it is directed in the negative direction. Since the brake is our only means of control, the control signal F is constrained to the interval

$$-13\,600 \leq F \leq 0 \tag{1.9}$$

This can be seen as follows. According to its data sheet, the car requires at least 27.3 m *to stop when driving at a speed of* 80 km/h*. As all the kinetic energy is converted to work, we have, on the average,*

$$\frac{1}{2}m\,(\dot{y})^2 = Fy$$

and thus

$$
\begin{aligned}
|F| &= \frac{1}{y}\frac{1}{2}m(\dot{y})^2 \\
&= \frac{1}{27.3}\frac{1}{2}1500\left(\frac{80\,000}{3600}\right)^2 \\
&\approx 13\,600
\end{aligned}
$$

We therefore assume that the anti-lock braking system limits the magnitude of the brake force to 13 600 *N (newton).*

The closed loop characteristic equation *is obtained by inserting Equation (1.3) into Equation (1.8):*

$$\ddot{y} = \frac{K_p\,(e + T_d\dot{e})}{m} = -\frac{K_p T_d}{m}\dot{y} - \frac{K_p}{m}y \tag{1.10}$$

There will be a steady state solution, since insertion of $\ddot{y} = \dot{y} = 0$ *yields the solution* $y = 0$; *this is just a check that a solution in accordance with the problem definition is feasible.*

Disregarding the nonlinearity, the transfer function in the Laplace domain is the forward path gain in the block diagram divided by 1 minus the loop gain (Mason's rule),

$$
\begin{aligned}
\frac{y(s)}{Ref} &= \frac{K_p\,(1 + T_d s)\,\frac{1}{m}\frac{1}{s^2}}{1 + K_p\,(1 + T_d s)\,\frac{1}{m}\frac{1}{s^2}} \\
&= \frac{\frac{K_p}{m}T_d s + \frac{K_p}{m}}{s^2 + \frac{K_p}{m}T_d s + \frac{K_p}{m}}
\end{aligned} \tag{1.11}
$$

The denominator is the closed loop characteristic polynomial*, compare Equation (1.10), and it is a second-order polynomial in* s. *The general transfer function of a second-order system is*

$$T = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{1.12}$$

Here ω_n *is the* natural frequency *– the frequency of oscillation without damping – and* ζ *is the* damping ratio. *It is very useful here, because we are looking for the response without overshoot, which is as fast as possible. This is the case when* $\zeta = 1$, *which yields a* critically damped *response. Comparing with Equation (1.11) our damping ratio is*

$$\zeta = \frac{1}{2}\sqrt{\frac{K_p}{m}}T_d$$

and taking $\zeta = 1$ *gives us an optimal tuning relationship between* T_d *and* K_p*,*

$$T_d = \frac{2}{\sqrt{\frac{K_p}{m}}} \tag{1.13}$$

Keeping this relationship ensures that the response has no overshoot, and consequently the velocity will be zero when the car arrives at the stop light. We used this relationship to choose the previously mentioned settings $T_d = 1$ *and* $K_p = 6000$*.*

So far we have turned a blind eye to the numerator in Equation (1.11): the first term contains a differentiation s *which is not present in the general transfer function. Our Ref is constant, however, with a zero derivative, in which case we can ignore the first term.*

The example illustrates that good knowledge of the process to be controlled is beneficial, and it is in some cases crucial. A mathematical model is even necessary in order to analyse stability. But more importantly, even though the fuzzy controller is based on expert rules, it must still be tuned, and the PD controller gave us a tuning (K_p and T_d) that we could carry forward to the fuzzy controller. The PD controller furthermore provides us with a reference for the performance of the fuzzy controller, which is very useful in practice.

1.5 Nonlinear Control Systems

The designer is faced with many choices in the design of a nonlinear fuzzy controller, and some choices are only suitable for some processes. We therefore have to take into account typical kinds of nonlinear processes. Even though nonlinear systems can exhibit a local behaviour that is different from the overall behaviour, there are a few general aspects to consider such as local equilibria and their stability.

Example 1.2 *Chemical tank reactor*

Given a nonlinear chemical tank reactor that requires cooling and heating by means of a surrounding jacket of water, describe the control task.

► *Solution*

A reactor can be in three different temperature states for a given flow of cooling liquid, but only one of them is desirable.

Hot water ramps the temperature of the tank up from the ambient temperature (near 20°C) to a temperature where the reaction begins to accelerate. If the reaction is exothermic *(it develops heat by itself), the heat released through the reaction must be removed by circulating cool water through the jacket.*

The controller must then be able to change from heating to cooling at a specified point that takes account of a certain time delay.

A simple simulation tool, *Autopilot*, will be used for testing nonlinear controllers. Autopilot simulates a driver-less train car on a track as in Figure 1.6. We model the train car using Newton's second law as previously, but this time the forces acting on the train car are nonlinear.

At points where the track is horizontal, such as A, B and C, the normal force counterbalances the gravitational force when the train is standing still, and the sum of the external forces is thus

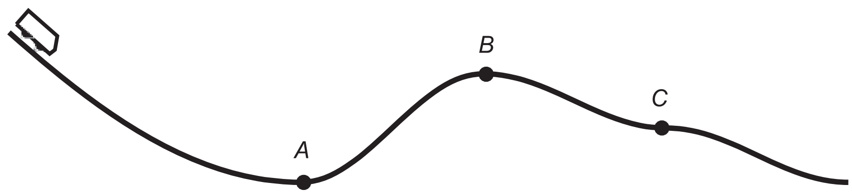

Figure 1.6 Autopilot. A driver-less train car moves on a hilly track. The drawing shows three kinds of equilibrium at stations A, B and C.

zero, that is, they are equilibrium points. Point A is a *stable equilibrium* because, when the train is disturbed slightly from the equilibrium, the forces tend to restore it to the equilibrium point. Point B is an *unstable equilibrium* since a slight disturbance away from the point results in a force that tends to move the train even farther away. Point C is a *saddle point* since a slight disturbance towards the left results in a force that tends to move the train back, while a slight disturbance towards the right results in a force that tends to move the train even farther away.

These are the only three types of equilibrium that a designer can meet. The example is therefore representative of many control problems. Furthermore, we can change the shape of the track as we wish, in order to emphasize other realistic situations in nonlinear control.

A controller affects the motion of the train according to our design. Assume that the task is to drive the train to station A as fast as possible, but without overshoot, then we have to craft a controller that applies a brake force in the correct amount. Assume instead that the task is to balance the train at station B, then we have to devise a rather tight controller that is able to take it there, and keep it in position, even if it is disturbed. The two equilibrium types call for two different control strategies. If the task is to stabilize the train in station C the controller must react in opposite ways to the left and to the right of the equilibrium point. On the left side, the curve helps the train to get back, while on the right side, the controller must work against the curve and use a larger effort.

The control strategy is non-symmetric in this respect, and it is possible to accommodate this in a fuzzy controller. If the task is to stabilize the train at a saddle point or on a slope of the curve, a linear PID controller may well turn out to be inadequate. In that case, a nonlinear fuzzy controller could be considered.

The Autopilot simulator could also model a chemical reactor, at least in principle, because the reactor behaves much like a saddle point. We shall use Autopilot throughout the book as a means to test various design choices and controller configurations.

Example 1.3 *Stop the car on a curve.*

Use Autopilot to simulate a car, but this time on a road shaped like a parabola. The mass of the car is $m = 1500$ kg *as previously, and the initial conditions are the same. The road is curved vertically according to the equation* $(x, y) = (x(t), 0.02x^2(t))$. *How does the previously designed PD controller perform when trying to stop the car at the bottom at* $x = 0$?

► *Solution*

The car is now affected by a downward slope, which affects speed. We therefore expect to use more brake force in order to stop the car at the horizontal position $x = 0$.

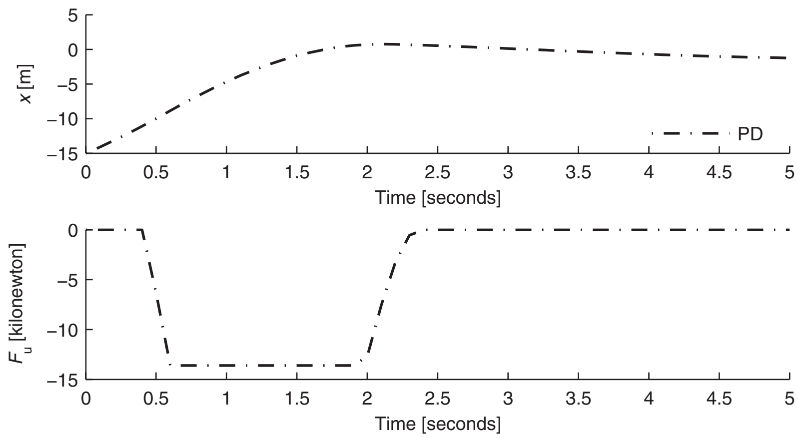

Figure 1.7 Stop the car at the bottom of a parabolic valley. The car starts at $x = -15$ m, and the controller tries to stop it at $x = 0$. (figrunautopilot1.m)

With the previous tuning ($K_p = 6000$ and $T_d = 1$) we get the response in Figure 1.7. Comparing this with Figure 1.4 the controller uses more control force. It kicks in early, it saturates quickly in the limit of the anti-lock braking system, it uses full brake force for more than a second, and then it quickly releases the brake power. But the car did not stop at the stop signal (the reference point); the upper plot shows that there is overshoot, before the car returns to the stop light. It is the upward curvature of the road that makes the car return, because the controller is only allowed to use the brake.

It is apparently difficult for the controller in the previous example to avoid overshoot. This could probably be fixed by tuning the controller a little tighter. The example demonstrates that one setting may not fit all situations. In that case, a solution might be to tune several controllers and interpolate between them, depending on a scheduling variable, such as the position. The method is called *gain scheduling*, and Chapter 8 will discuss how to interpolate between PID controllers by means of fuzzy rules.

1.6 Summary

It is relatively difficult to design a fuzzy controller, because it is in general nonlinear, and nonlinear systems are more or less unpredictable. Instead we propose to stay as long as possible in the linear domain, in accordance with the proposed design procedure. The idea is to start from a PID controller, and then to design a linear fuzzy controller that is equivalent to the initial PID controller. At this point, all the results from linear control theory can be applied, including tuning methods and stability calculations. In the next phase, the fuzzy controller is made nonlinear.

The design procedure has limited scope in the sense that it requires a PID controller. The reward is that the design procedure provides reliability: it guarantees that the fuzzy controller

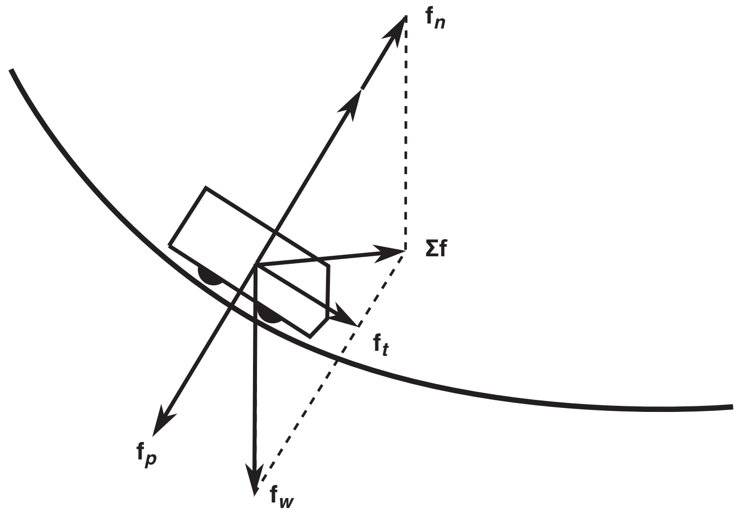

Figure 1.8 External forces affecting the train car.

performs at least as well as its initial PID controller. There is the possibility, but no guarantee, that it will perform better.

Some of the following chapters provide tools for analysing the nonlinear fuzzy controller, in particular, phase plane analysis and describing functions. Still, trial and error is a characteristic of fuzzy control.

1.7 The Autopilot Simulator*

This section is the first *-marked section in the book. The * in the heading signals that the section can be skipped on a first reading, because it contains background material which is unnecessary for the overall understanding. For the advanced student, however, the *-marked sections provide deeper insight and topics for further research. In this case, the section documents the inner workings of the Autopilot simulator written in Matlab (a programming environment for algorithm development, trademark of The MathWorks, Inc.[1]).

Autopilot simulates a driver-less train car on a track, which lies in a vertical plane to keep it simple. The train is affected by a downward gravitation force, a normal force from the track, and a control force from its motor/brakes.

According to Newton's second law, the equation of motion is $\sum \mathbf{f} = m\mathbf{a}$ where $\sum \mathbf{f}$ is the sum of forces acting on the train car, m is its mass, and $\mathbf{a}$ is the acceleration. The equation is valid for vectors, indicated by lower-case letters and set in a bold typeface. At a given horizontal position x, Figure 1.8 defines the external forces: the gravitational force $\mathbf{f}_w$ and the normal force $\mathbf{f}_n$. The former is resolved into its tangential component $\mathbf{f}_t$ and its perpendicular component $\mathbf{f}_p$.

[1] www.mathworks.com

Example 1.4 *Parabolic track*

Given the track curve $(x, y) = (x(t), ax^2(t))$ $(a > 0)$*, characterize the external forces affecting the train car and characterize the equilibrium point.*

► *Solution*

Notice first that the track curve has the shape of an upward bowl (parabola) possessing a minimum.

The normal force is the track's reaction. It balances $\mathbf{f}_p$*, and it also contains an extra velocity-dependent centripetal force because of the track's curvature. Since the track curves upwards, the magnitude of* $\mathbf{f}_n$ *is larger than the magnitude of* $\mathbf{f}_p$*. The resultant force* $\sum \mathbf{f} = \mathbf{f}_w + \mathbf{f}_n$ *consists of a tangential component, which accelerates the car, and a centripetal force, which is responsible for the curvature of the motion.*

The resultant force $\sum \mathbf{f}$ *points to the right, when* x *is negative, and it points to the left, when* x *is positive. In other words, the force always points towards the* y*-axis of the coordinate system. By intuition, this could result in an oscillatory motion around* $x = 0$*, the character of which depends on the initial position.*

At an equilibrium point, the sum of the forces is zero. This happens at $x = 0$ *when the train is standing still. The upward bowl shape of the curve indicates that the equilibrium is a stable equilibrium.*

1.8 Notes and References*

In the mid-1960s, Lotfi A Zadeh (born in 1921) of the University of California at Berkeley, USA, invented the theory of fuzzy sets. He argued that, more often than not, the classes of objects encountered in the real physical world have imprecisely defined criteria for membership (Zadeh 1965). For example, the 'class of numbers that are much greater than 1' or the 'class of tall human beings' have ill-defined boundaries. Yet such imprecisely defined classes play an important role in human reasoning and communication.

Ebrahim (Abe) H Mamdani, a control engineer at Queen Mary College in London, was attempting to develop an adaptive system that could learn to control an industrial process. He used a steam engine as a laboratory model, and with his colleagues set up a program that would teach the computer to control the steam engine by monitoring a human operator. At this point Mamdani's research student, Seto Assilian, tried to apply fuzzy logic. He created a set of simple rules in fuzzy terms, and Mamdani and Assilian then studied ways to use fuzzy rules of thumb directly in automating process controls. A few years later, Mamdani and Procyk managed to develop a linguistic self-organizing controller (Procyk and Mamdani 1979). It was an adaptive controller that was able to learn how to control a wide variety of processes, nonlinear and multi-variable, in a relatively short time. It was called *self-organizing* because at that time the meaning of the words 'adaptive' and 'learning' had not yet been agreed upon. The work of the pioneers led to a growing literature in fuzzy control and wide-ranging applications, as Table 1.1 illustrates.

In Japan, Michio Sugeno developed a self-learning fuzzy controller (Sugeno, Murofushi, Mori, Tatematsu, and Tanaka 1989). Twenty control rules determined the motion of a model car. Each rule recommends a specific change in direction, based on the car's distance from the walls of a corridor. The controller drives the car through angled corridors, after a learning session where a 'driving instructor' pulls it through the route a few times. Self-learning

Table 1.1 Milestones in early fuzzy history.

Year	Event	Reference
1965	First article on fuzzy sets	Zadeh 1965
1972	A rationale for fuzzy control	Zadeh 1972
1973	Linguistic approach	Zadeh 1973
1974	Fuzzy logic controller	Assilian and Mamdani 1974
1976	Warm water process	Kickert and van Nauta Lemke 1976
1977	Table based controller	Mamdani 1977
1977	Heat exchanger	Østergaard 1977
1977	Self-organizing controller	Procyk and Mamdani 1979
1980	Fuzzy conditional inference	Fukami *et al.* 1980
1980	Cement kiln controller	Holmblad and Østergaard 1982
1983	Train operation	Yasunobu *et al.* 1983
1984	Parking control of a model car	Sugeno *et al.* 1989
1985	Fuzzy chip	Togai and Watanabe 1985
1986	Fuzzy controller hardware system	Yamakawa and Miki 1986
1987	Sendai subway in operation	Yasunobu *et al.* 1983
1989	Fuzzy home appliances sold in Japan	
1989	The LIFE project is started in Japan	
1990	Rule learning by neural nets	Kosko 1992
1990	Hierarchical controller	Østergaard 1990, 1996

controllers that derive their own rules automatically are interesting because they could reduce the effort needed for translating human expertise into a rule base.

The first industrial application was in 1978 where a fuzzy controller was operating in closed loop on a rotary cement kiln in Denmark. Fuzzy control then became a commercial product of the Danish cement company FL Smidth & Co (now FLSmidth). The fuzzy control research program in Denmark was initiated in 1974 (Larsen 1981).

The Laboratory for International Fuzzy Engineering (LIFE), Yokohama, was set up by the Japanese Ministry of International Trade and Industry in 1989. It had a six-year budget and a research staff of around 30. LIFE conducted basic research with universities and member Japanese companies and subsidiaries of US and European companies, including Matsushita, Hitachi, Omron, and VW – about 50 companies in all. The research program was trimmed to five major projects: image understanding, fuzzy associative memory, fuzzy computing, intelligent interface, and the intelligent robot. They were all carried out with two themes in view: a navigation system for the blind and home computing.

A European network of excellence called ERUDIT was initiated in 1995 with support from the European Commission. ERUDIT, which lasted six years, was an open network for uncertainty modelling and fuzzy technology, aimed at putting European industry at the leading edge. The network was followed by another network EUNITE[2] with a broader scope: smart adaptive systems. That network was in turn followed by a coordinated action NISIS[3] with an even broader scope: nature-inspired systems.

[2] www.eunite.org
[3] www.nisis.de

For further information

Beginners may start with two articles in Institute of Electrical and Electronics Engineers (IEEE) Spectrum (Zadeh 1984, Self 1990) and then move on to the more advanced textbook by Timothy Ross (2010); it is oriented towards applications in engineering and technology with many calculated examples. The most efficient way to learn about fuzzy logic and fuzzy control is to study the Fuzzy Logic Toolbox for Matlab, especially the Fuzzy Inference System (MathWorks 2012). The toolbox covers other related techniques such as clustering, neuro-fuzzy systems, and genetic algorithms; for an in-depth description, see the related book by toolbox author Roger Jang and his two coauthors (Jang, Sun and Mizutani 1997).

The terms rule base and inference engine are loans from the field of expert systems, and Lee (1990) uses these to give a wide survey of the whole area of fuzzy control. The article lists 150 references.

A major reference on fuzzy control is the book by Driankov, Hellendoorn and Reinfrank (1996). It is explicitly targeted at the control engineering community, engineers in industry, and university students. The more recent book by Michels, Klawonn, Kruse and Nürnberger (2006) is a comprehensive reference book that includes the current state of the art at the time of writing. It views fuzzy control from the viewpoint of classical control theory, just as the present book does. Chapter 3 gives more specific references related to fuzzy control.

Industrial applications are described in a special issue of the journal *Fuzzy Sets and Systems*, for instance, the fuzzy car by Sugeno *et al.* (1989) and an arc welding robot by Murakami *et al.* (1989). There are more early applications in the classical book by Sugeno (1985). Ten years later, Constantin von Altrock (1995) described more than 30 case studies from companies that employed fuzzy and neuro-fuzzy methods. The FLSmidth controller is described in detail by Holmblad and Østergaard (1982).

There are more than ten journals related to fuzzy sets. Two of the major journals are *Fuzzy Sets and Systems* and *International Journal of Approximate Reasoning*, both published by Elsevier, and a third one is *Journal of Intelligent and Fuzzy Systems*, published by IOS Press, Netherlands. The Institute of Electrical and Electronics Engineers, IEEE, started a journal in 1992 called *IEEE Transactions on Fuzzy Systems*. The *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems* is published four times per year by World Scientific Publishing Co. It is a forum for research on imprecise, vague, uncertain and incomplete knowledge. Other journals that occasionally have fuzzy control articles are *Automatica*, the control section of *IEE Proceedings*, *IEEE Transactions on Systems Man and Cybernetics*, *IEEE Transactions on Computers, Control Engineering Practice,* and the *International Journal of Man-Machine Studies*.

There is an active newsgroup called `comp.ai.fuzzy`.[4] It supplies useful news, conference announcements, and discussions.

There are two major professional organizations. The International Fuzzy Systems Association (IFSA) is a worldwide organization dedicated to fuzzy sets. IFSA publishes the *International Journal of Fuzzy Sets and Systems* (24 issues per year), holds international conferences, establishes chapters and sponsors activities. The other organization is the North American Fuzzy Information Processing Society, NAFIPS,[5] with roughly the same purpose.

[4] groups.google.com/group/comp.ai.fuzzy
[5] nafips.ece.ualberta.ca/