Part 1 of *Dynamic Thinking*-- Dynamic thinking in the brain, linked to the body, Oxford University Press, (in press, 2014)

Chapter 1: The dynamics of neural activation variables

Gregor Schöner, Hendrik Reimann, and Jonas Lins

Abstract

Behavior is generated by the central nervous system (CNS). From this simple observation we argue that the inner state of the CNS must be described by continuous variables that evolve continuously in time. We introduce the concept of activation to characterize the inner state of the CNS and postulate that activation evolves in time as described by a dynamical system. We discuss the form this neural dynamics takes based on the need for states of the CNS to be stable—for behaviorally significant states to resist perturbations. Stability means that the system coheres around special states called attractors where the rate of change of activation is balanced. This might occur under the influence of sensory input, or when neurons are coupled together, passing excitatory or inhibitory activation back-and-forth. We also discuss the movement of the CNS into and out of attractor states, which are formally called instabilities. Such instabilities arise in the CNS due to the nonlinear way in which neurons interact. In sum, this chapter thus introduces the core concepts of neural dynamics on which Dynamic Field Theory (DFT) is based.

As you are reading these lines, your nervous system is engaged in the three aspects of behavior, perception, action, and cognition. Whenever your gaze falls onto a particular part of the visual array, your brain processes sensory information. Your brain controls motor actions that actively shift your eyes from fixation to fixation. And your brain makes sense of the visual patterns, recognizing letters, linking the recognition across multiple fixations and bringing about complex thoughts. Understanding how the brain, together with the sensory and motor periphery, brings about perception, action, and cognition requires a theoretical language that reaches across theses different domains. A central theme of this book is that the neural processes from which behavior emerges evolve continuously in time and are continuously linked to each other and to online sensory information. These processes generate graded signals that steer motor behavior. Continuity in state and in time invites the language of dynamical systems. This Chapter will introduce the core elements of that language.

Within the language of dynamical systems, stability is a critical concept. Stability is the capacity to resist change in the face of variable inputs, such as variation in sensory inputs or variation in the signals received from other neural processes. For instance, if you are looking at a picture in this book, you may be able to focus on only that picture and ignore distractions, the music you have running in the background, the cars passing by the window next to you, the other pictures in the book. The rich environments in which we are immersed always provide alternatives to what we are currently processing. Our rich behavioral repertoire always provides alternatives to the motor action we are currently engaged in. And inside our nervous system, neural processes are richly interconnected and inherently noisy. So for any particular neural process to be effective and to impact on behavior it needs to be stabilized against the influence of all the other competing processes and against noisy inputs. We will discuss in this chapter how the concept of stability can be formalized in the language of dynamical systems, and how the theoretical models must be formed so stability is assured.

Stability means resistance to change. Cognition, however, requires change. Detecting a stimulus, initiating an action, or selecting one of multiple possible actions, all of these are decisions that imply change: The neural state before the decision differs from the neural state after the decision has been taken. To understand how stable neural processes allow for change, we need to understand how neural states are released from stability, what we will call a dynamical instability. This chapter will discuss stability and the basic types of dynamic instabilities that are central to Dynamic Field Theory and will recur throughout the book.

We begin with the concept of neural activation to capture the inner state of the Central Nervous System (CNS). We will talk about how activation can be linked to states of the world outside the nervous system, that is, to sensory stimuli or motor actions. Next, we will introduce the core notions of neural dynamics. Insisting that neural states have stability properties narrows down the range of dynamical models. We will look at the linear dynamical model of a single activation variable to introduce the basic notions of dynamical systems: Fixed points and their stability. Even a single activation variable may interact---with itself. We will introduce the notion of a sigmoid nonlinearity and will find that self-excitation of an activation variable may give rise to a first instability, the detection instability which occurs in response to input. We then consider two activation variables that interact inhibitorily, leading to competition. Such a simple system may already make selection decisions. When one of two inputs becomes dominant, a second instability, the selection instability, occurs. Excitatory and inhibitory interaction and the two instabilities they produce are constitutive of Dynamic Fields, as we shall see in Chapter 2, and will follow us throughout this book.

<2>1.1 Activation

How do neural processes supported within the CNS generate behavior? To begin addressing that question we clearly need some way to characterize different inner states of the CNS that lead to different kinds of behavior. In choosing such a characterization we are selecting a level of description of the CNS. In Dynamic Field Theory, we hypothesize that it is the activity of populations of neurons within circumscribed brain areas that is tightly related to behavioral patterns. Chapter 3 will operationalize this hypothesis by constructing activation fields from the firing rates of a population of neurons. In Chapter 2 we will show how neural activation fields and their dynamics may form neural representations of behavior. In this chapter, we will use the concept of neural activation variables and look at the simplest cases in which behavior is linked to only one or two such activation variables. In Chapter 2 we shall find out that these activation variables are best viewed as measures of the neural activity within circumscribed subpopulations of neurons. Localized hills or peaks of activation in neural activation fields represent these subpopulations.

A neural activation variable, the way we will use it, is a real number that may both be positive or negative. One may think of an activation variable as akin to the membrane potential of a neuron, so that the probability of eliciting an action potential is larger the higher the activation level. The biophysics of neurons are briefly reviewed in Box 1.1, but DFT is not intended to be biophysically detailed and the analogy to neural firing is not critical to understanding DFT. In fact, we do not use actual units of electrical potential to describe activation, nor will we take into account the mechanisms of spike generation and of synaptic transmission. We will, however, capture the basic idea of synaptic transmission by assuming that there is a threshold, which we set to be zero, so that only activation values above that threshold, that is, only positive levels of activation, are transmitted to other activation variables. This assumption is formalized through the sigmoidal function, illustrated in Figure 1.1, which increases monotonically from zero for very negative levels of activation to one for large positive activation levels.

[INSERT BOX 1.1 AROUND HERE]

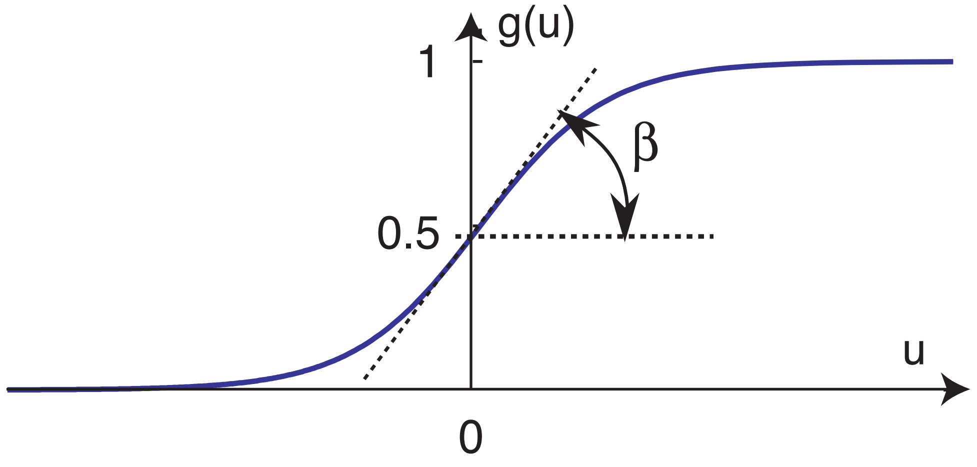

Figure 1.1: A sigmoidal threshold function, $g(u)$, is plotted as a function of activation level, u. The sigmoid maps low levels of activation onto zero, large levels of activation onto one, and links these two regimes smoothly as a monotonically increasing function. By convention, we position the half-point of the sigmoid at the activation level of zero. That convention effectively defines the activation scale. In DFT models we typically use the mathematical formalization of $g(u) = 1/(1 + \exp(-\beta u))$, where β is the slope of the sigmoid at zero activation. Larger values of β create steeper (more non-linear) sigmoids.

Connectionism uses a similar concept of activation to describe the inner state of each unit of parallel processing, the abstract connectionist “neuron”. Most connectionist models use graded activation variables. Connectionist neurons may then be “on” or “off” (Thomas, McClelland, 2008), characterized again by a sigmoidal threshold function applied to the activation level. Some connectionist models use binary activation variables to begin with, so they do not require a separate sigmoidal threshold function. In Chapter 3 we will see that the activation variables of DFT are measures of activity in small subpopulations of neurons. These variables thus do not directly reflect of the state of individual neurons. In typical connectionist models, the model neurons are similarly meant to encompass activity of more than one real neuron. Thus, overall, the concept of activation is used more variably in connectionism, but is not qualitatively different from the dynamic concept of activation used in this book.

A concept of activation is also invoked in some versions of classical cognitive architectures, models of cognition that are based on the computer metaphor and on concepts of information processing. In ACT-R, activation is an attribute of items of memory that determines how accessible the memorized information is (Anderson, 1983). Multiple factors like the salience of an item, the strength of its association with other items, or the strength of a memory trace of the item may contribute to the level of activation. The probability of retrieval of an item is an increasing (sigmoidal) function of activation and the latency of retrieval is an exponentially decaying function of its activation level. These two relationships link activation to observable response rates and response times. In a broad sense, there is some analogy between this notion of activation and our dynamic concept of activation, in that high levels of activation have more impact on behavior (responses) than low levels. The theoretical setting of ACT-R is so different, however, from neural dynamics, that this analogy is not easy to carry along and for the remainder of this book we will ignore that alternate notion of activation.

So if activation characterizes the inner state of a part of the CNS, how might that inner state be related to what is outside the CNS? Ultimately, the CNS is connected to the outside world through the sensory surfaces, the retina, the cochlea, the skin, the distributed proprioceptive sensors, and other sensory systems. Moreover, neural activity drives motor systems, activating muscles and bringing about mechanical change in the world. The connections sensor cells make to a portion of the CNS can be characterized as an influence on the activation level of relevant activation variables. This will be quite easy to conceptualize within DFT as we shall see shortly. Conversely, activation variables may impact on motor systems, driving muscle activation and changing the physical state of an effector. That is actually trickier to conceptualize than you may first think. In terms of the metaphor of the Braitenberg vehicles that we used in the introduction to this part of the book, motor action always brings along the potential of closed sensory-motor loops as any motor action has sensory consequences. We will address this problem in depth in Chapter 4.

Much of functional neurophysiology is dedicated to looking for systematic relationships between stimulus or motor parameters and the activity of neurons. This is often based on information-theoretic notions, in particular, coding and prediction. In this book, we try to stay away from such notions. Coding principles and their relationship to feed-forward neural networks are briefly reviewed in Box 1.2, where we also discuss how the language of neural dynamics is necessary to make sense of recurrent neural networks.

For now, let us say then that in DFT the inner state of the CNS is related to the world outside through two directions of influence: the state of the world influences the levels of activation and those levels of activation influence the state of the world through motor actions. In fact, it is ultimately only through those links to the sensory and motor systems that the inner states of the CNS have meaning. This may, in the end, be the concrete manifestation of the embodiment stance to cognition (Riegler, 2002), and we shall come back to this point multiple times in this book.

[INSERT BOX 1.2 AROUND HERE]

<2>1.2 Neural dynamics

The inner state of the CNS typically varies continuously in time. Unlike digital computers, organisms do not have a clock that updates the state of the CNS in a computational cycle. Nor is there any behavioral evidence that processing occurs from time step to time step. To the contrary, there is behavioral evidence for online updating of CNS states that occurs in continuous time. For instance, if the target, to which a pointing movement is directed, is shifted at any time during the processes of movement preparation or initiation, the movement begins to reflect that shift after a delay of about 100 ms. That shift is invariant as the timing of the target shift is varied (Prablanc, Martin, 1992). We should think, therefore, of activation variables as functions of continuous time, denoted mathematically by $u(t)$, where, u, stands for activation, and, t, for continuous time.

Does this time dependence itself have to be continuous? In other words, does $u(t)$ change smoothly in time or may $u(t)$ abruptly jump from one value to another? At the level of the biophysics of neurons, the formation of an action potential would seem to be an abrupt event, although it is actually continuous on a finer time scale (see Box 1.1). There is really no evidence that behavior is driven by such microscopic events. To the contrary, there is behavioral evidence for inertia, for gradual change of activation states. A classical example is visual inertia in motion perception (Anstis, Ramachandran, 1987), in which a percept of visual motion is set up by a first stimulus of apparent motion, followed by an ambiguous stimulus that offers two possible paths of motion, one path in the same direction as the first motion, the other at an angle to the initial path. Observes prefer the motion path in the same direction. (The exact mapping of such perceptual continuity to our activation variables requires some work, which we will do in a formal way only in Chapter 2).

The postulate that activation variables, $u(t)$, are continuous functions of continuous time has important consequences. It rules out, for instance, the idea that the values of activation variables originate from simple input-output computations (see Box 1.2), because in such input-output systems any abrupt change of input induces a matching abrupt change in output. Neural dynamics formalizes this postulate of continuous evolution of activation in continuous time. Neural dynamics means that the time course of an activation variable, $u(t)$, is the solution of a differential equation

$$\tau \dot{u} = f(u) \qquad (1.1)$$

where $\dot{u}(t)$ is the rate of change of u, and τ is a positive constant that serves to define the units of time (e.g, seconds or milliseconds). Here, $f(u)$ is a smooth function of activation, u, and we need to figure out, which function, f, produces the right time courses of activation.

Before we do that, let's unpack Equation (1.1). The rate of change of an activation variable is formally its derivative with respect to time, $\dot{u}$. If we were to plot the time course of activation, $u(t)$, against time, t, the rate of change would be the slope of that function. To make that intuitive, think of activation as the position of a particle. The rate of change of the position of a particle is its velocity, simple as that! The differential equation above, Equation (1.1), forms a dynamical system for the activation variable, $u(t)$ (see Box 1.3 for a tutorial on dynamical systems). The solutions of the differential equation are time continuous (in fact, differentiable) trajectories of activation, $u(t)$, for which Equation (1.1) is true, that is, whose rate of change, $\dot{u}$, is the prescribed function, $f(u)$, of its current activation, u.

But what function, $f(u)$, would be appropriate? We need another postulate to narrow in the class of admissible dynamical systems defined by $f(u)$. That additional postulate is *stability*. Intuitively, stability means something like resilience, the capacity to recover from perturbations. In the CNS, neural noise is a common form of perturbation. Neural processes vary stochastically (see Box 1.4 for an excursion on noise and fluctuations). Neural variability acts as stochastic perturbations on any activation variable that receives neural input. Stability enables the activation level to resist such perturbations. Other forms of perturbation are distractors, that is, activation states that are not compatible with the current activation pattern in the CNS. For instance, when gaze is fixed on a visual target, neural activation from a visual stimulus outside the fovea would tend to attract attention and to redirect gaze to that new location. Stability is the capacity to resist such distractor activation (even if resistance is limited in time and strength, see Kopecz, Schöner, 1995, for an early neural dynamic account of such resistance).

Because the CNS is highly interconnected, an activation variable is exposed to influences from many other activation variables or directly from sensory stimulation. Most of the time, many of these influences are not consistent with the current state of the activation variable, that is, they would tend to drive activation away from the current state. Without stability, the CNS would not be able to shield a particular state from all other possible influences that would disrupt neural function quite generally. We will examine the postulate of stability in more detail below and again in Chapters 2 and 4. For now, we use the stability postulate to constrain the class of neural dynamics, $f(u)$, that generates behaviorally meaningful time courses of activation.

[INSERT BOX 1.3 AROUND HERE]

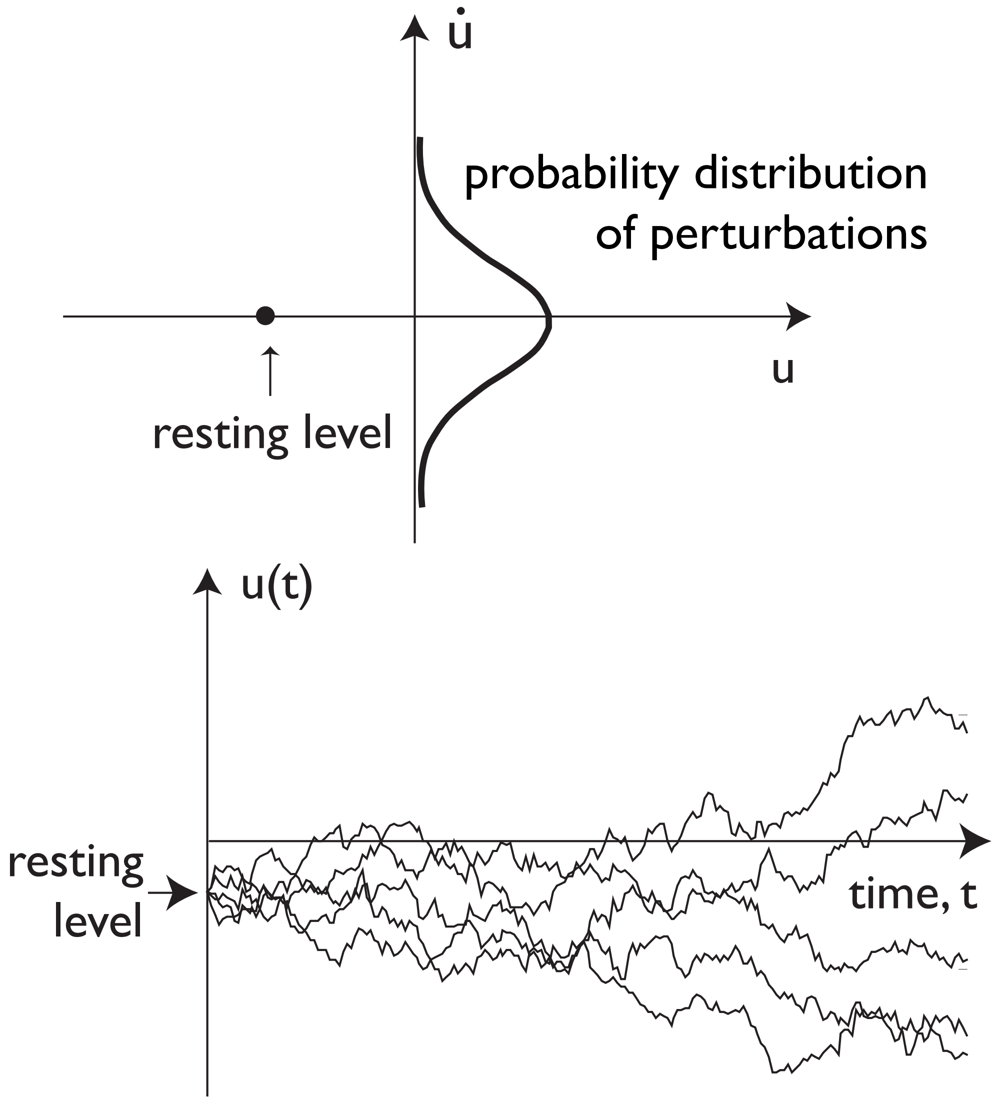

Figure 1.2: On top, a neural dynamics is illustrated by plotting the rate of change of activation, $\dot{u}$, against activation, u. In this case, the mean rate of change is zero across all levels of activation, but random rates of change are drawn independently at each moment in time from a Gaussian distribution (which is illustrated for the level of zero activation: this distribution is meant to stick out from the page, the same distribution would exist for every level of activation). On bottom, different time courses of activation, $u(t)$, that are generated by this stochastic neural dynamics are shown as functions of time, t. All trajectories start at the same level of activation, labeled "resting level", but evolve differently due to different samples being drawn from the probability distributions.

How stability constrains the function, $f(u)$, can be understood by first looking at the trivial case in which $f(u) = 0$, illustrated in Figure 1.2. In this case, the rate of change of activation is constant at zero, independently of the current level of activation. Any initial level of activation will thus remain unchanged over time. But what happens when random perturbations impact the activation variable? A random perturbation can be modeled as a random kick that generates a non-zero rate of change for a short (infinitesimal) moment in time (see Box 1.4 for a brief tutorial stochastics). The random perturbations may be distributed as a Gaussian as hinted in the figure, so large kicks are less frequent than small kicks, the average kick size being zero. Kicks at different times are assumed to be independent of each other. Such random influences are called Gaussian white noise, $\xi(t)$, and form a good model of sources of stochasticity based on fundamental laws of probability (Arnold, 1974). Formally, the neural dynamics with noise can be written down as

$$\tau\dot{u} = \xi(t). \qquad (1.2)$$

Any time a positive kick is applied, activation increases. Every time a negative kick is applied, activation decreases. Over time, activation performs a random walk as illustrated in the figure, in which multiple time courses obtained by different samples of the noise process are shown. As is apparent from those simulations, the variance of the random walk increases without bound! This is essentially the law of Brownian motion, first modeled mathematically by Einstein (1905). Intuitively, this increase of variance comes from the fact that there is no systematic restoring force that pushes activation back to the starting value. If perturbations have driven activation to a certain level, say a positive level, future kicks are just as likely to further drive activation away from the starting level as they are to drive levels of activation back to the starting level.

Clearly, this model is missing something to become functionally meaningful: It is missing a restoring force that keeps activation within bounds. Such a restoring force would have to assure that when large positive activation levels have been reached, the probability of negative rates of change becomes much larger than the probability of positive rates of change, so that kicks back toward lower activation levels become prevalent. Analogously, when very negative activation levels have been reached, the probability of positive rates of change must become larger than the probability of negative rates of change. Figure 1.3 illustrates such probability distributions. They are centered on a line with negative slope so that, in fact, the mean rate of change is negative far out on the positive activation axis and positive far out on the negative activation axis.

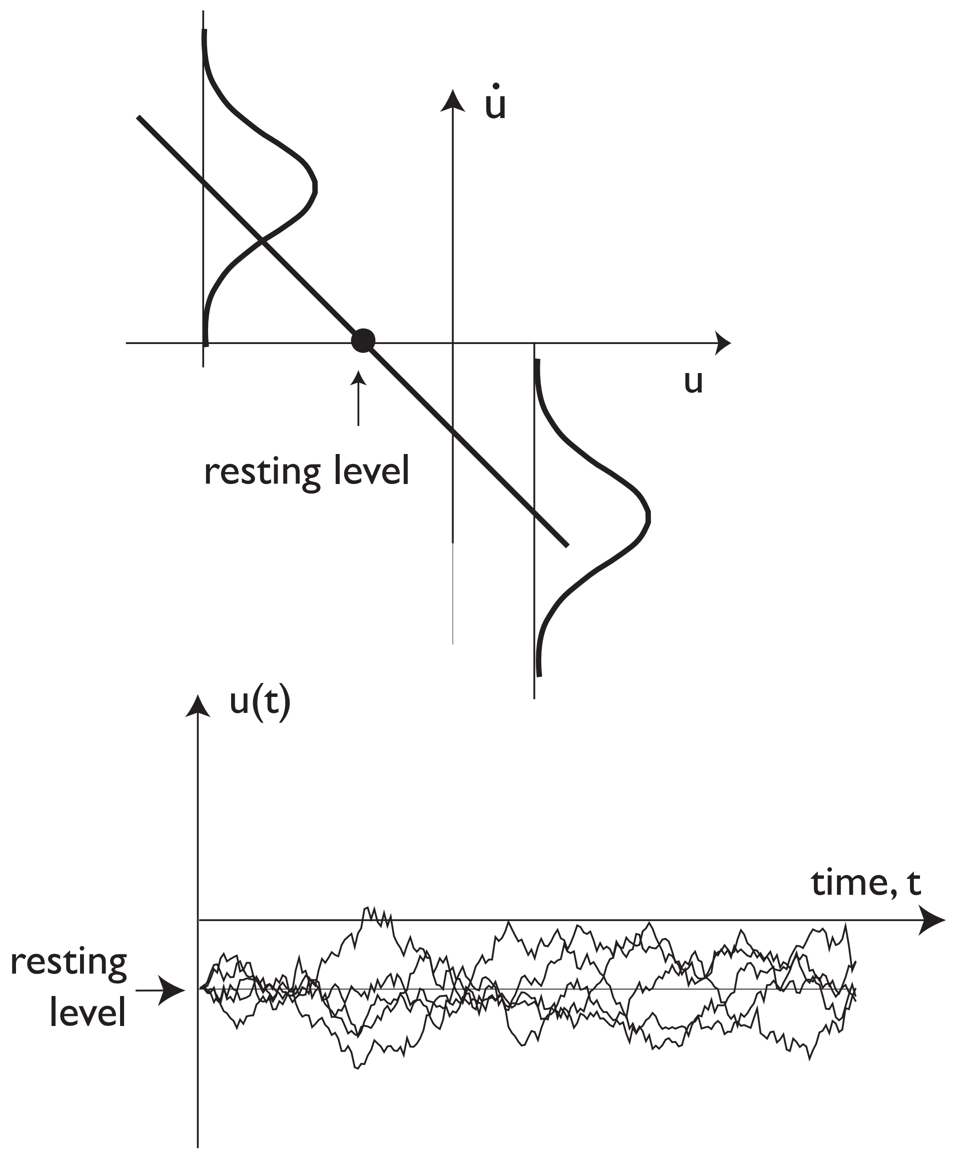

Figure 1.3. Analogous to Figure 1.2, but now the mean rate of change is a function of the activation level illustrated on top by the straight line with negative slope. Two examples of probability distributions are illustrated. The one on the right is centered on a negative rate of change; the one on the left is centered on at positive rate of change. Their means lie on the straight line. The different samples of the activation trajectories shown on bottom now remain bounded and are centered on the resting level.

Mathematically, this model can be written as

$$\tau\dot{u} = -u + h + \xi(t). \tag{1.3}$$

and its deterministic portion is illustrated in Figure 1.4. Here, $-u$ makes the straight line with negative slope. By adding a negative constant, $h < 0$, we have shifted the straight line downwards, so that it intersects the activation axis at $u_0 = h$. That intersection point we call the resting level of activation. It is formally the solution of

$$\tau\dot{u} = 0. \qquad (1.4)$$

This solution is a *fixed point*, a constant solution, $u(t) = h$, of the dynamics (see Box 1.3). This fixed point is also an *attractor*, defined by the fact that activation converges to the fixed point over time from any initial activation level in the vicinity of the fixed point. Our earlier reasoning about how activation levels remain bounded explains this convergence as well: If activation starts at levels higher than that of the fixed point, then the neural dynamics has negative rates of change which imply that activation will decrease, and thus approach the fixed point from above. If activation starts at levels lower than that of the fixed point, positive rates of change imply that activation will grow, approaching the fixed point from below. It is thus the negative slope of the rate of change around the fixed point that imposes the stability constraint. The level of activation at the fixed point is the stable activation state. The negative slope of the rate of change at the fixed point thus brings about stability.

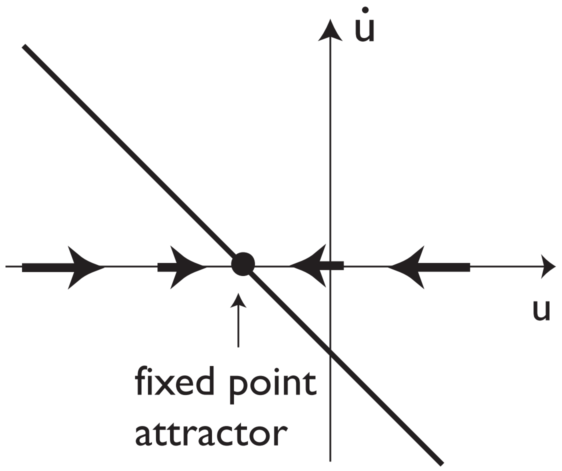

Figure 1.4 Dynamics of a single neural activation variable of the form, $\tau\,\dot{u} = -u + h$, illustrated by plotting the rate of change of activation, $\dot{u}$, against activation, u, itself. The intersection with the activation axis at the resting level is an attractor, a stable fixed point. Along the activation axis, arrows show the direction of change. The length of the arrows indicates the rate of change, which approaches zero near the attractor.

A more formal way of seeing the convergence to the fixed point is to solve the differential equation. Box 1.3 shows how to do this analytically. More commonly, in DFT we solve differential equations numerically on a digital computer (see Box 1.4 for a review of numerics). Such numerical simulations formally instantiate the iterative account we have been using intuitively. Time is sampled at discrete times separated by a small size, Δt. The time course of activation, $u(t)$, is approximated by a discrete time sequence, $u(t_i)$, where $t_i = i \cdot \Delta t$ and i counts discrete time, $i = 0, 1, 2, \ldots$. In the simplest numerical procedure (called the Euler formula), the time sequence may be obtained from the approximation of the rate of change

$$\dot{u} \approx \frac{u(t_i) - u(t_{i-1})}{\Delta t}. \qquad (1.5)$$

Inserting this into the dynamics (still neglecting noise) we obtain after some rearranging of terms:

$$u(t_i) = u(t_{i-1}) + \frac{\Delta t}{\tau}[-u(t_{i-1}) + h]. \qquad (1.6)$$

On the right hand side, only present values of u at the earlier time, t_{i-1}, are needed. They determine the value of activation at the next time step, t_i, on the left hand side. In other words, this is an iterative equation: Starting with some initial value of activation, future values can be obtained by iterating the equation in discrete time into the future. (Numerical solutions of the stochastic version Equation 1.3 of the dynamics are discussed in Box 1.4).

Figure 1.5 illustrates time courses obtained this way. Different solutions were obtained by setting different initial conditions, $u(0)$, so that activation starts out different levels. Clearly, independent of the different initial levels, activation converges in all cases to the fixed point at the resting level. This convergence, often called “relaxation”, takes the form of an exponential decay of the difference from the fixed point, a characteristic of the solutions of linear equations. The time constant of the exponential decay is the parameter τ. That is why we said earlier, that τ fixes the units of time. This time constant is also called the “characteristic time” or “relaxation time” of the neural dynamics.

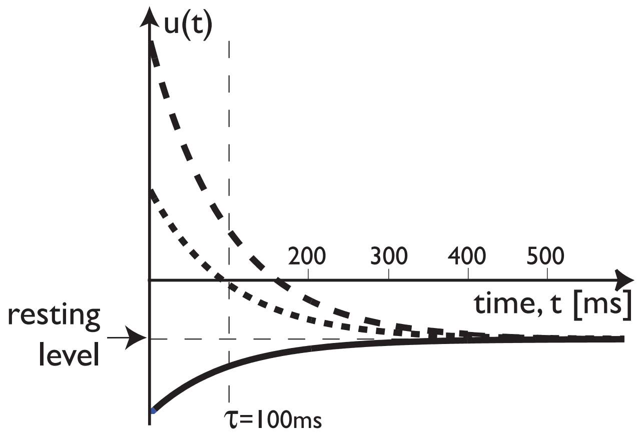

Figure 1.5 Three activation trajectories are shown as functions of time. These were obtained by numerically solving $\tau\dot{u} = -u + h$. Activation converges ("relaxes") to the resting level, h, from different initial values. The time, τ, that it takes to reduce the initial distance from the attractor by 36.8% (the reciprocal of the Euler number e) is marked by the dashed vertical line. This time is independent of the absolute level of initial activation and defines the time scale of the dynamics.

The last step to make sense of neural dynamics is to consider inputs to the dynamics, which may originate from the sensory surfaces or from other activation variables. In neural dynamics, inputs are contributions to the rate of change. Positive contributions are excitatory inputs; negative contributions are inhibitory inputs. To be specific, consider an input from a sensory system, $s(t)$, that varies in time. Figure 1.6 illustrates how the neural dynamics changes as the input increases from zero to a positive value, s_0, in an abrupt step. Because the input does not depend on the activation level itself its increase shifts the entire dynamics, that is, the negatively sloped function of activation upward. As a result, the zero-crossing moves to the right, from the resting level, h, to a positive value, the new fixed point at $h + s_0$. The system was initially at resting level but because that is no longer a fixed point activation begins to change. Activation relaxes exponentially to the new fixed point with the same time constant with which it relaxes to the resting level in the absence of input. Note that what impacts on other neurons is not activation itself, but the output of the activation variable that is obtained by applying the sigmoidal threshold function to the activation variables. Figure 1.6 shows the time course of this thresholded output level. While activation responds to a step change of input with an exponential time course, the output level has a more abrupt time course.

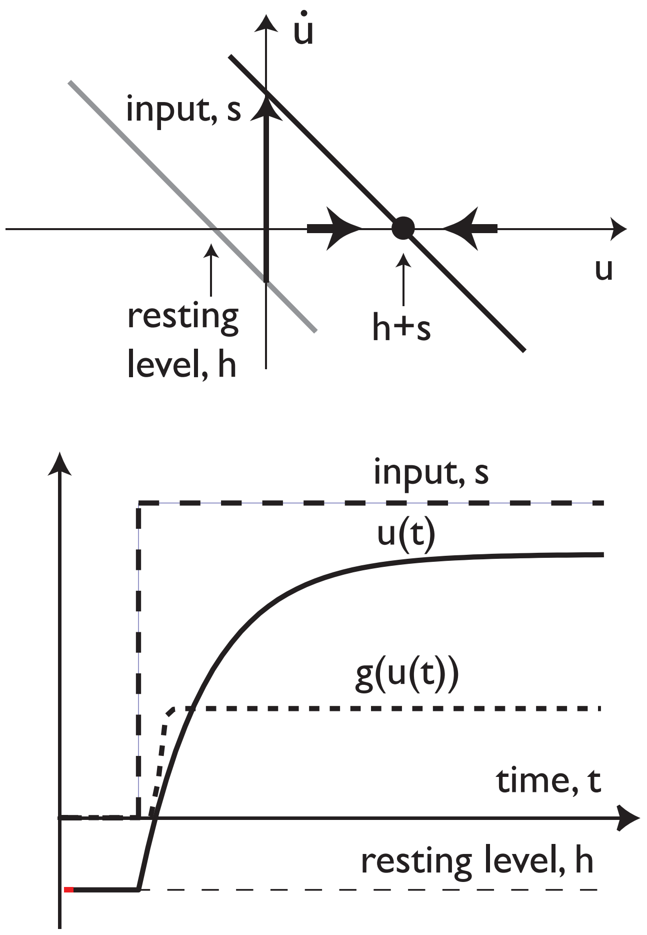

Figure 1.6 On top, the neural dynamics $\tau\dot{u} = -u + h + s(t)$ is illustrated. The gray line reminds us of the dynamics without input, $s(t)$, that has a fixed point at $u = h$, the resting level. Input shifts the rate of change upwards, leading to a new fixed point at $u = h + s$. On bottom, the resulting activation trajectory, $u(t)$ (solid line), is shown together with a sketch of the associated input, $s(t)$ (dashed line). The dotted line shows the output of the activation variable obtained by applying the sigmoid threshold function to the activation trajectory.

You see that the attractor structures the time courses of activation. The attractor itself may move, here even jump. Activation changes smoothly, tracking and at all times moving toward the attractor. In this simple case of a single activation variable driven by a single input, the neural dynamics acts as a low-pass filter, smoothing the time course of input on the characteristic time scale, τ. Exercise 1.1 gives you the opportunity to explore with an interactive simulator how the neural dynamics generates continuous time courses out of potentially discontinuous inputs. Next we will look at how more complex neural dynamics may do the opposite--transform continuous inputs into discontinuous activation time courses that represent the simplest form of decision making, the decision that an input has been detected.

.

<2>1.3 Self-excitation and the detection instability

All of this has been about a single activation variable receiving external input. Now we will look at neural interaction. Neural interaction refers to the dependence of the rate of change of an activation variable on input from other activation variables. Neural interaction includes, therefore, the forward neural connectivity that characterizes many connectionist networks. More typically, however, neural interaction refers to patterns of coupling that include recurrent loops of connectivity. A limit case that we will use as a starting point here is the neural dynamics of a single activation variable that receives excitatory input from itself. That is the simplest form of recurrent neural connectivity, a network consisting of only one neuron that connects back onto itself as illustrated in Figure 1.7. Such circuits exist in the CNS, but we will see in Chapter 2, that this limit case really stands for the neural dynamics of small populations of neurons that are mutually coupled through excitatory connections. Mathematically, self-excited neural dynamics can be formulated by adding a single term to the rate of change we considered so far:

$$\tau \dot{u} = -u + h + s(t) + c \cdot g(u), \qquad (1.7)$$

where the parameter $c > 0$ represents the strength of the self-excitatory contribution. The sigmoid threshold function, $g(u)$, was illustrated earlier (Figure 1.1) and can be formalized mathematically as

$$g(u) = \frac{1}{1 + \exp(-\beta u)}. \qquad (1.8)$$

Consistent with the concept of activation, only sufficiently positive levels of activation impact on other activation variables, which is assured by passing activation through the sigmoidal function, $g(u)$. This mathematical formulation highlights how input now is dependent on the activation level, u, which is the signature of neural interaction. Note that the dependence of the rate of change of activation on the activation variable itself through the "$-u$" term is not part of neural interaction, as this term does not represent input, but establishes the intrinsic neural dynamics that generates stability.

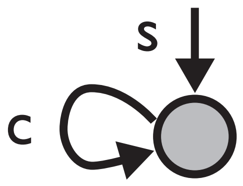

Figure 1.7 The dynamics of a single activation variable, illustrated by a circle filled in grey, is represented in the manner of neural networks. Excitatory external input, s, is indicated by an incoming arrow. Self-excitatory neural interaction is illustrated by a single recurrent loop ending in an arrow. The strength of that input is modulated by the parameter, c.

Figure 1.8 illustrates this neural dynamics with self-excitation. For very negative activation levels, the sigmoid returns zero and we have the linear dynamics from before. For very positive activation levels, the sigmoid returns a constant (here 1), so that the linear dynamics are shifted upward by c. The sigmoid connects these two regimes, leading overall to a nonlinear dynamical system. A dynamical system is nonlinear, whenever the dependence of the rate of change on the current level of the activation variables is not a straight line. Figure 1.8 shows that without external input, $s(t)$, (and for sufficiently negative h and sufficiently small c), the dynamics does not change qualitatively over the linear dynamics. There is still a single attractor at the resting level and the rate of change is negative everywhere to the right of that attractor. The system is "mono-stable" around the resting state, meaning, there is only a single attractor along the entire activation axis. That is the attractor, in which activation would settle.

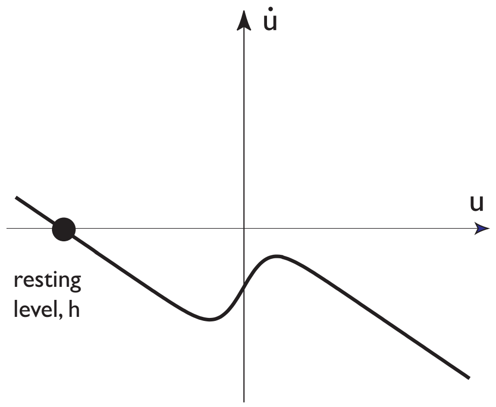

Figure 1.8. The neural dynamics of a single activation variable with self-excitatory neural interaction is shown in the absence of external input, s.

If excitatory input of increasing strength is applied, the dynamics is shifted upwards as shown in Figure 1.9. At some point, the nonlinear dynamics touches the activation axis at positive activation levels and, with just a little more input, two new fixed points arise. The one at a higher, positive level of activation is an attractor, as can be recognized by the negative slope of the dynamics at that fixed point. The sigmoid threshold function applied to this attractor level of activation returns non-zero values, to that this is attractor represents an "on" state of the activation variable. The fixed point at a somewhat lower, but still positive level of activation is a repellor, which can be inferred from the positive slope of the dynamics at that fixed point. Small deviations from the repellor are amplified by the dynamics: Deviations to the right are linked to positive rates of change, so activation grows further away from the repellor. Deviations to the left are linked to negative rates of change, so activation decreases away from the repellor. The repellor therefore divides the activation axis into two regimes that are called basins of attraction. One leads to the new "on" attractor, the other to the old "off" attractor at negative levels of activation. This is illustrated in Figure 1.10, in which the dynamics at this point is solved numerically starting with different initial conditions. Starting at larger activation levels than the repellor leads to convergence to the new on-attractor; starting at lower activation levels than the repellor leads to convergence to the old attractor at negative activation levels.

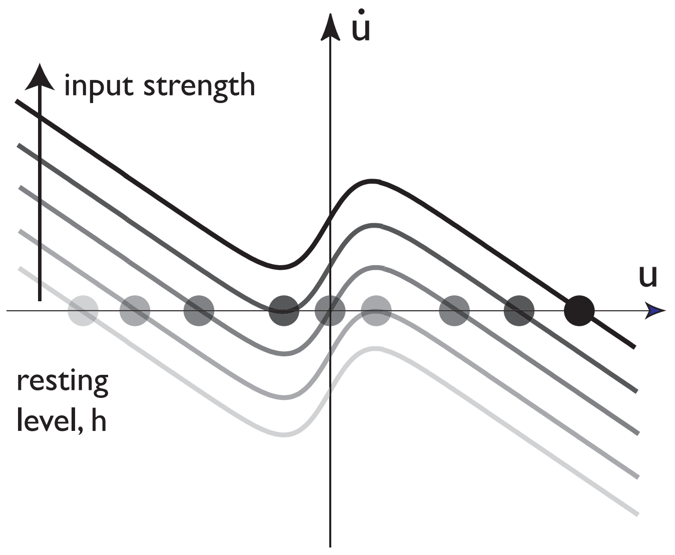

Figure 1.9 The neural dynamics of a single activation variable with self-excitatory neural interaction in the presence of increasing amounts of external input, s, is illustrated by graphs going from light grey to solid black. Circles that are filled by matching grey levels mark the fixed points. Note that the three inner fixed points are unstable.

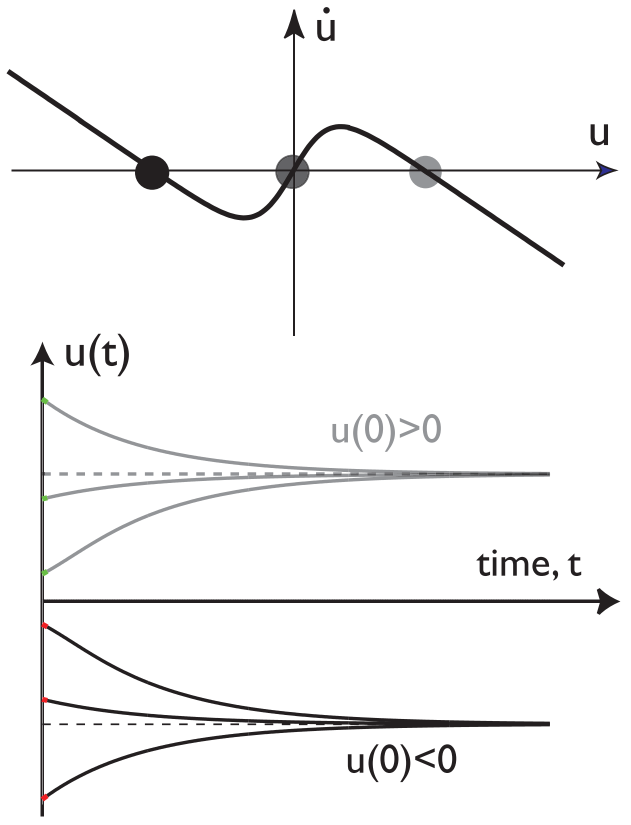

Figure 1.10 The dynamics of a single activation variable with self-excitation, $\tau\dot{u} = -u + h + s(t) + c \cdot g(t)$, is shown on top in the presence of external input, $s(t)$ = constant of intermediate strength. The dynamics has an attractor at a negative activation level (circle filled in solid black), and another one at a positive activation level (circle filled in light grey), separated by a repellor at zero activation level (circled filled in dark grey). Simulated activation trajectories shown

on bottom converge to the positive attractor (dashed line in light grey) when started at positive initial activation levels (light grey curves), and to the negative attractor (dashed line in solid black) when started at negative initial activation levels (solid black curves). This shows that the dynamics is bistable.

Although the new fixed points have appeared as input was applied, activation is not be affected by them so far. Before input arrived, the system was sitting in the "off" attractor. When input arrived, that attractor (left-most attractor in Figure 1.9) shifted somewhat, but the activation variable tracks that shift. A qualitative change is brought about only when the "off" attractor becomes unstable, as input is further increased. The repellor moves toward the "off" attractor at negative activation and ultimately collides with it, annihilating the attractor. The dynamics lifts off the activation axis and no attractor remains at negative activation level. At this point, activation can no longer stay around the off state. Activation will vigorously grow, converging to the "on" attractor on the right.

This phenomenon, the disappearance of an attractor, is associated with a loss of stability (see Box 1.3). The slope of the dynamics at the attractor becomes flat just before the attractor disappears. This means that the restoring force, that drives activation back to the attractor after a perturbation, becomes weaker. This is why such a change of the dynamics around an attractor is called an instability. Mathematicians prefer the term "bifurcation" as such instabilities always involve multiple fixed points colliding or splitting.

The instability is a significant event, even though it happens only at one particular level of input and even though the system quickly moves away from the now unstable and then vanished attractor. This is because instabilities separate different dynamic regimes. Before this instability, the system is bi-stable, it has two attractors at its disposal, the "on" state at positive levels of activation and the "off" state at negative levels of activation. After the instability, only the "on" state is left, which is now mono-stable.

We call this instability the "detection instability" because when it is run through in the order narrated here, with increasing input from bi-stable to mono-stable, it generates the "decision" that significant input has been detected. That decision is reflected by the fact that the activation level goes through zero, so that the sigmoidal threshold function goes from zero to a positive value. Note that this decision mechanism differs from the classical notion of signal detection theory, in which a detection decision is made when a criterion level is exceeded. In the detection instability, the detection decision is stabilized. That is, once the decision has been made, it is maintained even if in a next moment in time the input strength falls back below the critical level due to sensory noise, for instance. The system remains in the "on" attractor, which is possible, because the system is bistable. In classical threshold thinking, in contrast, a decision is a momentary event that is not stabilized per se. If we were to use such threshold thinking in the context of dynamical systems thinking, we would run into problems. This is because in dynamic thinking, the decision variable activation is updated continuously in time based on time varying sensory inputs. A threshold mechanism would poorly then: A "yes" decision would often be followed by a switch to a "no" and perhaps a switch back to "yes" as sensory inputs fluctuate. Thus, continuous time decision making in noisy sensory environments really requires that decisions are stabilized. We will return to this point in Chapters 2 and 4.

When input strength drops to sufficiently low values, however, the detection decision is undone as the bi-stable regime merges into the mono-stable "off" regime. This is the first instability we discussed above that happens at lower levels of input, on the right side of the graph in Figure 1.9. We refer to that instability as the "reverse detection instability", which marks the point at which the loss of a detection is signaled. Again, this decision is stabilized. If input strength rises beyond this lower critical level, then the non-detection decision is maintained.

Together, these two instabilities form the basis of a phenomenon called decision hysteresis: The critical level at which a detection is signaled depends on the direction of change of input strength. This is illustrated in Figure 1.11 which traces the attractor state that is realized when stimulus strength is increased or decreased. Empirically, hysteresis is ubiquitous in perceptual psychophysics. In fact, hysteresis has been known from the earliest days of psychophysics as the dependence of perceptual judgments on the direction of stimulus change (Stevens, 1957). It has often been attributed to time delays in perceptual processing, so that judgments are based on past rather than on present stimuli, as well as on inertia of response, so that past judgments influence current judgment. Hysteresis was then typically eliminated from data by averaging judgments made for the two directions of stimulus change. There is a growing body of work, however, that shows that there is a purely perceptual component to hysteresis. This comes from very carefully designed studies, in which factors such as stimulus uncertainty, response delay and response mode have been excluded or minimized (Hock, Schöner, 2010). An example is the detection of apparent motion. Apparent motion is visual motion seen between locations on the visual array at which there is luminance change. This is how we see motion from sequences of still images as in the movies or on TV. In the laboratory, the probability of seeing apparent motion can be manipulated by varying a stimulus parameter, the background-relative-luminance-contrast or BRLC. Higher values induce motion, low values induce a flickering percept without a motion direction. When BRLC is gradually increased, motion is first detected at higher values of BRLC than the BRLC values at which observers lose the motion percept when BRLC is gradually decreased (Hock, Kogan, Spinoza, 1997). This instance of hysteresis thus matches quite closely the neural dynamic phenomenon illustrated in Figure 1.11.

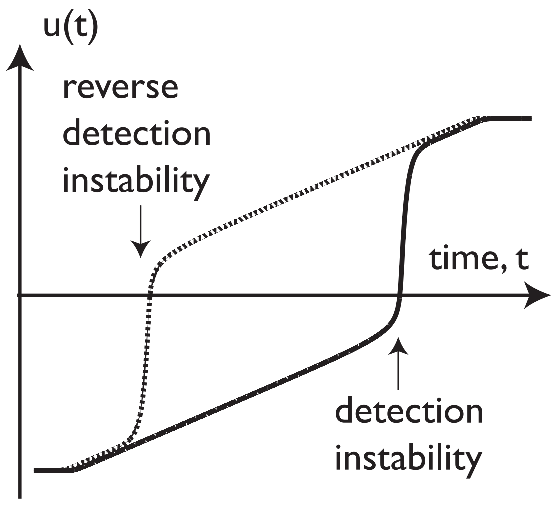

Figure 1.11 The dynamics of a single activation variable with self-excitation, $\tau\dot{u} = -u + h + s(t) + c \cdot g(t)$, is simulated when external input, $s(t)$, is first increased, then decreased, in both cases linearly in time. The resulting activation level is shown as a solid line for increasing, and as a dashed line for decreasing input (plotted against a reversed time axis). The dependence of the realized activation level on the direction of change of input is a signature of hysteresis.

A related conceptual point is illustrated very nicely by the detection instability. Although the neural dynamics evolve in continuous time and the activation variables change values continuously, events emerge at discrete moments in time from the neural dynamics. Consider, for instance, a stimulus, $s(t)$, that gradually increases in strength. The detection instability translates this gradual change into a discrete jump of activation from the off- to the on-state when stimulus strength reaches a critical level. That jump occurs at discrete time, the time at which the system goes through the instability and makes the transition. Discrete time is not inherent in the dynamical system, unlike a digital computer whose computation progresses at discrete times driven by its clock cycle. Time for the dynamical system is running continuously. From that continuous evolution of the neural dynamics the detection event emerges at a discrete moment in time through the detection instability.

The detection instability also illustrates a final conceptual point. In DFT, the attractors are not "sitting" somewhere in the neural dynamics, "waiting" for the system to "fall into" them. Instead, the attractors arise on the fly. Without sufficient sensory input, the "on" attractor does not exist. It arises afresh when input reaches a critical level, which happens to be the level at which the reverse detection instability occurs. Only from then on is the attractor "around," available for the system to move to it. That decision, to go to the "on" attractor is brought about by a separate instability, the detection instability, which makes the "off" attractor unstable and destroys it. There are neural network models, such as the Hopfield model (Hopfield, 1981) or the brain-state-in-a-box model (Anderson, Silverstein, Ritz, Jones, 1977), in which many attractors co-exist at the same time, built into the neural network. In such models, a perceptual decision consists of relaxing from an initial state, set by the stimulus, to the nearest attractor. In a sense, such neural networks implicitly invoke information processing as a concept: The network is given an input at a discrete time. It then runs to an attractor, which may be considered its output. That "response" must somehow be "read" by some other system. There is nothing inside the network that would signal that the network is "done", that it has finished its "computation". This task lies outside the network. In contrast, we think of behavior as unfolding in continuous time in DFT. The neural dynamics evolves continuously, receiving inputs that may vary perhaps even just because the behavior moves the sensors around. The time varying inputs may move the attractors around. But because the neural dynamics is almost always in an attractor, it follows those changes, tracks the attractor. All this goes on continuously until an instability occurs, that triggers a detection event. The neural dynamics shifts to a new attractor and behavior changes. In the new neural attractor, the system is again ready to track any further changes in input. All this is autonomous, no other system is needed to "read" the output or "know" when the dynamics is done computing. (In Chapter 4 we will look at the closed sensory-motor loop that was invoked here in detail.)

<2>1.4 Sustained activation and working memory

The hysteresis of detection is a first indication that in neural dynamics the stimulus does not uniquely determine the inner state of the CNS. Through hysteresis that inner state depends on the history of stimulation and of activation. Working memory pushes this idea further. In working memory, the inner state of the CNS reflects perceptual events in the past while the associated sensory stimulation is no longer available on the sensory surfaces. The simple neural dynamics of a single activation variable with self-excitation provides a first model of working memory in the form of sustained activation (Fuster, Alexander, 1971; Fuster, 1995). The memory function emerges when the bistable co-existence of the on- and off-state extends all the way to zero input strength. Figure 1.12 illustrates the idea. In the figure, the neural dynamics without input, $s(t)$, is already bistable. This may be because the resting level is high enough for self-excitation alone to be sufficient to lift activation above zero. Or it may arise because the strength c of self-excitation is sufficiently large for self-excitation to push activation above zero once positive activation levels have been reached. This neural dynamics has, therefore, an off- and an on-state even without any external input, $s(t)$.

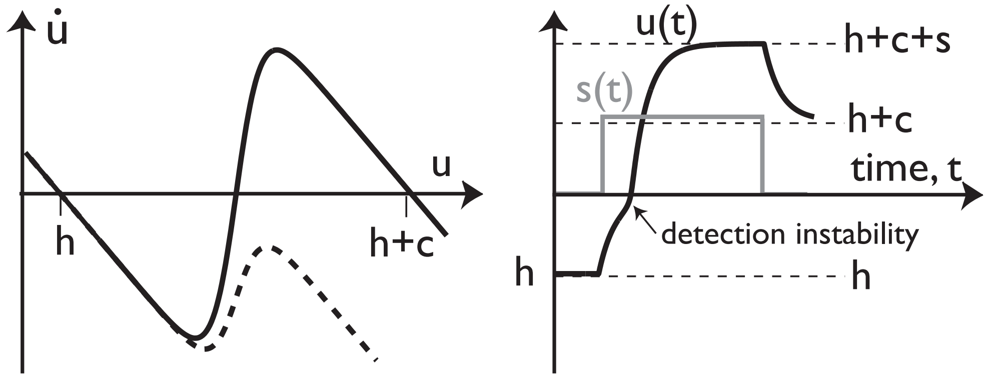

Figure 1.12 The dynamics of a single activation variable with self-excitation, $\tau\dot{u} = -u + h + s(t) + c \cdot g(t)$ is shown on the left when self-excitation is strong (solid line) compared to when it is weak (dashed line). In both cases, no external stimulus, $s(t)$, is present. For the larger self-excitation, the time courses on the right shows how the activation variable (solid black line) goes through the detection instability when a stimulus (grey line) is presented, but then remains in the on-state when the stimulus is removed.

If the activation variable starts out in the off-state, a sufficiently strong input, $s(t)$, will lift the dynamics up, beyond the detection instability, eliminating the off-state and inducing a switch into the on-state. The activation trajectories in Figure 1.12 show how this happens. When the input is again removed, the activation variable tracks the on-attractor, which drops to lower activation levels but persists in the absence of input. The neural dynamics remains on, reflecting a memory of past stimulation. In principle, the system may remain indefinitely in this self-sustained state.

Models that take into account that multiple activation variables may represent multiple different stimuli explain additional characteristic features of working memory such as a limited memory capacity. This will be a topic in Chapter 2 and in much more detail in Chapter 6. We will also address more specifically how memories may be reset through inhibition and how working memory may be controlled. But the basic mechanism is already contained in this simple model.

<2>1.5 Inhibitory interaction and selection

We are finally ready to move beyond a single activation variable. Two activation variables form the simplest true neural network depicted in Figure 1.13. If the two activation variables excite each other, the dynamics is not much different from the case we just discussed, including detection instabilities and the potential for sustained activation. In fact, we will see in Chapter 2, that the single self-excited activation variable really stands for a local population of activation variables that are mutually coupled in an excitatory fashion.

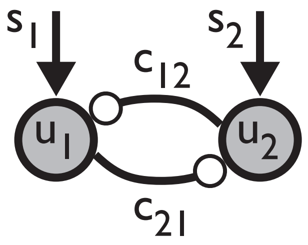

Figure 1.13 The dynamics of two, mutually inhibitory activation variables, each shown as a circle filled in grey, is illustrated in the manner of neural networks. Arrows indicate excitatory external inputs. Lines ending with an open circle indicate inhibitory coupling.

Something new happens, however, if we consider inhibitory neural interaction. Mathematically, the two activation variables, u_1 and u_2, have these neural dynamics:

$$\tau \dot{u}_1 = -u_1 + h + s_1(t) - c_{12} \cdot g(u_2) + \xi_1(t) \qquad (1.9)$$

$$\tau \dot{u}_2 = -u_2 + h + s_2(t) - c_{21} \cdot g(u_1) + \xi_2(t) \qquad (1.10)$$

Here, both variables share the same resting level, h, but have their own external input, $s_1(t)$ and $s_2(t)$, respectively, as well as their own noise sources, ξ_1 and ξ_2. Interaction or coupling means that activation variable u_2 contributes to the rate of change of activation variable u_1 and vice versa. Again, as for self-excitation, only positive levels of activation impact on other activation variables, so the interaction is mediated by the sigmoidal threshold function, $g(\cdot)$. The coupling is inhibitory because the contribution is negative (see equations 1.9 and 1.10, assuming strengths, $c_{12} > 0$ and $c_{21} > 0$). The symmetric form of interaction is called mutual inhibition.

Clearly, if both activation levels are below zero, no interaction happens due to the sigmoidal threshold function. So for something interesting to happen, assume both activation variables receive positive inputs, s_1 and s_2, that would be sufficient to produce positive activation levels if the variables were not coupled. To understand mutual inhibition, we examine different cases. Assume that activation variable u_2 already was at a positive level of activation before u_1 left its resting level. This situation may have arisen because u_2 received input earlier than u_1 or because the input to u_2 was stronger than the input to u_1. The sigmoidal function on u_2 then returns a value of 1, so that the inhibitory influence of u_2 on u_1 manifests itself, pulling the rate of change of u_1 down by c_{12}. Figure 1.14 illustrates what this implies. Were it not for that inhibitory input, u_1 would have a single attractor at the on-state, reflecting its significant input s_1. The downward shift of the rate of change caused by inhibition from u_2 moves the attractor to the left, into the negative activation regime. As a result, the sigmoid of u_1 returns zero, so that u_1 cannot, in turn, inhibit u_2. Activation variable u_2 has won the competition set up by mutual inhibitory coupling.

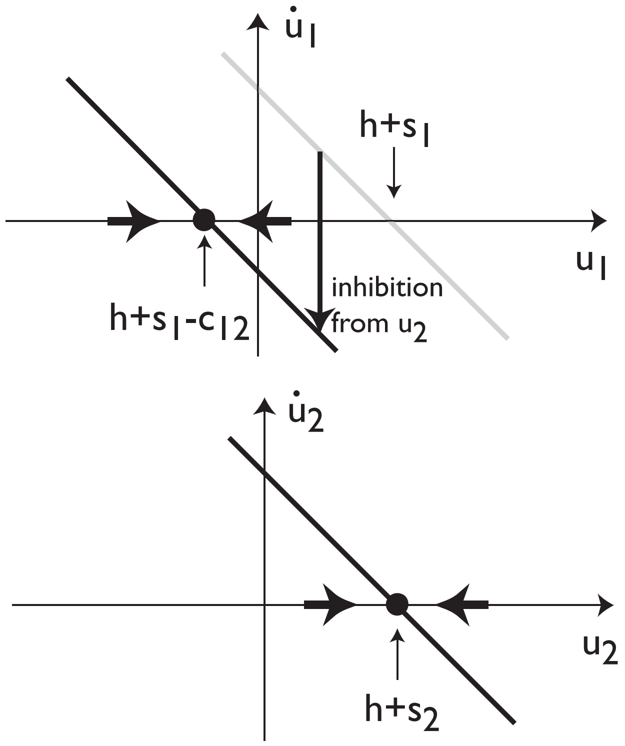

Figure 1.14 The dynamics of two activation variables that interact through mutual inhibition and both receive external input (s_1 and s_2) is illustrated by plotting separately the rates of change of each activation variable against the

corresponding activation level. The dependence of the rate of change of each activation variable on the other activation variable is taken into account by considering different cases. On top, the grey line is the dynamics of u_1 when u_2 is sufficiently below zero so that the sigmoid returns zero. The attractor then lies at $h + s_1$. The line in solid black is the dynamics when u_2 is above zero, so that the sigmoid returns one. This dynamics is shifted down by the strength of inhibitory interaction, c_{12}, and has an attractor at $h + s_1 - c_{12}$, which lies below zero. On bottom, the line in solid black is the dynamics of u_2 when u_1 is below zero. This leads to an attractor for u_2 at $h + s_2$, which is above zero. The dynamics outlined in solid black in both plots are thus consistent with each other.

The reverse outcome is expected if activation at u_1 had risen above zero, before activation at u_2 had a chance to do so. In neural dynamics, selection among multiple activation variables that all receive significant input is generally sensitive to the temporal order of activation. From Chapter 2 on, we will be looking at many activation variables that are inhibitorily coupled. In such systems, the activation variables receiving the strongest input will grow fastest and reach zero first. They can then begin to suppress the other variables and win the competition. What determines how "strong" an input is in a neural network? We have used only this simple picture of one input strength per activation variable. In more complex neural networks, inputs are patterns defined over the input layer. That input layer connects through a forward synaptic pattern onto the neural network. A neuron in the neural network receives a strong input, if the input pattern "matches" the synaptic pattern of forward connectivity. That is, input units that make a strong excitatory synapse on a given neuron are maximally stimulated and those input units that have no excitatory or even an inhibitory synapse on the given neuron are minimally stimulated. Better match of an activation variable to input in this sense thus translates into larger input strength. And that in turn leads to an earlier rise of activation making the activation variable dominate the competition.

The narrative around Figure 1.14 explains how the competition between two activation variables unfolds. This narrative suggests that the selection decision is stabilized by the neural dynamics: Once one activation variable is excited above threshold, it suppresses the other enough for that other activation variable to remain below threshold. That suppression creates a gap that input to the other activation would need to bridge for that other activation variable to have a chance to get above threshold. In fact, you see in Figure 1.14 that the attractor of u_1 is significantly below zero. Even if input to u_1 was subsequently strengthened so that it became larger than the input to u_2, the attractor of u_1 would remain below zero and the attractor of u_2

would not move. In fact, there is a bistable range, in which either selection decision is possible. Once locked into one decision, the neural dynamics resists change.

The stabilization of *selection* decisions plays a similar functional role as the stabilization of *detection* decisions discussed earlier. When a neural dynamics makes choices in response to continuously varying and noisy sensory signals, the selection decision emerges at a particular time and is then maintained. Figure 1.15 shows that this is even true in the limit case, in which both activation variables receive the same input and the selection decision results from chance, as stochastic perturbations push one activation variable above zero, suppressing the other. This type of bi-stability has limits. If input strengths are very disparate, the more strongly stimulated activation variable will ultimately overcome competition from the other activation variable, even if that variable is already above threshold. We will examine this case in the context of activation fields in the next chapter.

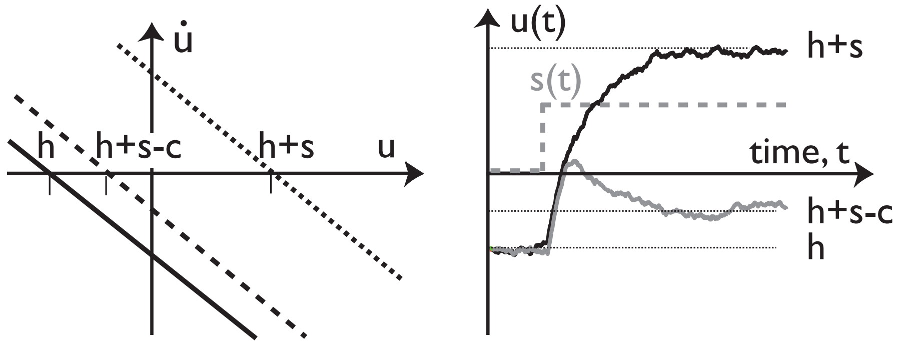

Figure 1.15 The dynamics of one of two competing activation variables is plotted on the left in three cases: Without external input (solid); with external input, but without (dotted) vs. with (dashed) inhibition from the other activation variable. The attractor is at resting level, h, in the first case, is shifted to the positive level of $h + s$, in the second case, and is shifted back to the negative level of $h + s - c$, in the last case. We have omitted the subscripts from s and c, referring to either variable. On the right, activation trajectories for both activation variables are shown (one in solid black, the other in solid grey). These were obtained from a numerical simulation that included noise. The grey dashed line illustrates at which time input, $s(t)$, was applied. The thin dotted lines mark the annotated levels of activation. Note that both activation variables receive the same input and both initially rise in parallel. Near zero, random fluctuations tip the balance in favor of the activation variable shown in black, which reaches threshold slightly earlier and begins to suppress the other activation variable. Towards the end of the simulation, the dynamics of the black variable corresponds to the dotted line in the left panel, while that of the grey variable corresponds to the dashed line.

<2>1.6 Conclusion

We started this chapter with an image of how the CNS generates behavior, linking cognition to perception and action. We intuited that directing movement requires the CNS to have graded inner states. In this chapter we have seen how we may characterize the inner state of the CNS in terms of graded activation.

We postulated that the graded inner state of the CNS must evolve continuously in time to generate behavior that may, in closed loop, couple to continuous sensory information. This led us to propose that the activation variables evolve continuously in time as described by differential equations, the neural dynamics central to this book. In this framework, sensory information contributes to the rate of change of activation. Neural coupling among variables means that they contribute to each other's rate of change, either positively for excitatory coupling, or negatively for inhibitory coupling. Only if the inner states of the CNS resist random or systematic perturbations from competing neural processes may these states generate coherent and persistent behavior even as they are coupled to time varying and noisy inputs. Stability is thus a central demand of the neural dynamics framework. This demand leads to the "$-u$" term--the negative slope of the dynamical system--that creates attractor states within the system. When attractors change with time varying inputs, the activation level tracks those changes thanks to the "$-u$" term.

Finally, we saw how the stability demand leads to the need for instabilities or bifurcations. Decisions require change from a pre- to a post-decision state. And change can only occur if the resistance to change, stability, is overcome. We saw specifically, self-excitation (or similarly, mutual excitation) leads to the detection instability beyond which on-states are stabilized. Mutual inhibitory interaction leads to selection decisions, which are likewise stabilized, but may undergo the selection instability when an alternative receives much stronger input than the selected choice.

Stability, we argued, would enable activation variables to link to time varying sensory inputs. In the examples we looked at, the level of activation tracked the strength of input. What if the sensory input changes along other dimensions than its strength? What if a visual stimulus moves on the retina even as it retains its brightness or contrast, for instance? What we have laid out does not answer that question. More generally, how do particular activation variables come to be linked to particular kinds of sensory inputs? How does a portion of the retina, for instance, link up to a particular set of activation variables? When we think of selection decisions, the lack of an answer to this question becomes painfully obvious: How does one activation variable come to stand for one choice, the other for another choice? If the choices themselves change, how is such a mapping updated, how does an activation variables still "know" which choice it stands for? In the next chapter we will embed the ideas of neural dynamics into a broader context that provides answers to these questions. We will discover that activation is not only graded in activation level, continuous in time, but also continuous in (feature) space.

References

Anderson, J. A., Silverstein, J. W., Ritz, S. A., & Jones, R. S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. Psychological Review, 84, 413–451.

Anderson, J. (1983). A spreading activation theory of memory. *Journal of verbal learning and verbal behavior*, *22*(3), 261–295.

Arnold, L. (1974). *Stochastic differential equations*. Wiley, New York.

Anstis, S. M., & Ramachandran, V. S. (1987). Visual Inertia in Apparent Motion. *Vision Research*, *27*, 755–764.

Braitenberg, V., & Schüz, A. (1991). *Anatomy of the Cortex*. Berlin: Springer Verlag.

Braun, M. (1993). *Differential equations and their applications* (4th ed.). Springer Verlag, New York.

Einstein, A. (1905). On the movement of small particles suspended in a stationary liquid demanded by the molecular-kinetic theory of heat. *Annalen der Physik*, *17*, 549. (Originally in German, English translation published in Einstein, A.: Investigations on the theory of the Brownian movement. Dover Publications, New York, 1956)

Faugeras, O., Touboul, J., & Cessac, B. (2009). A constructive mean-field analysis of multi-population neural networks with random synaptic weights and stochastic inputs. *Frontiers in Computational Neuroscience*, *3*(February), 1–28.

Fuster, J. M. (1995). *Memory in the Cerebral Cortex---An Empirical Approach to Neural Networks in the Human and Nonhuman Primate*. MIT Press.

Fuster, J. M., & Alexander, G. E. (1971). Neuron activity related to short-term memory. *Science*, *173*, 652–654.

Gardiner, C. W. (2009). *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. 4th. Edition. Springer Verlag, Berlin.

Haykin, S. O. (2008). *Neural Networks and Learning Machines* (3rd ed.). Prentice Hall, New Jersey, USA.

Hock, H. S., Kogan, K., & Espinoza, J. K. (1997). Dynamic, state-dependent thresholds for the perception of single-element apparent motion: Bistability from local cooperativity. *Perception & Pschophysics*, *59*, 1077–1088.

Hock, H. S., & Schöner, G. (2010). Measuring perceptual hysteresis with the modified method of limits: dynamics at the threshold. *Seeing and perceiving*, *23*(2), 173–195.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences U.S.A.*, *79*, 2554–2558.

Kandel, E. R., Schwartz, J., & Jessel, T. (2013). *Principles of Neuroscience* (5th ed.). McGrawHill Professional, New York.

Kelso, J. A. S. (1995). *Dynamic Patterns: The Self-Organization of Brain and Behavior*. The MIT Press.

Kloeden, P. E., & Platen, E. (1999). *The Numerical Solution of Stochastic Differential Equations* (2nd ed.). Springer-Verlag.

Koch, C. (1999). *Biophysics of Computation*. New York: Oxford University Press.

Kopecz, K., & Schöner, G. (1995). Saccadic motor planning by integrating visual information and pre-information on neural, dynamic fields. *Biological Cybernetics*, *73*, 49–60.

Perko, L. (2001). *Differential Equations and Dynamical Systems* (3rd ed.). Berlin: Springer Verlag.

Prablanc, C., & Martin, O. (1992). Autonomous control during hand reaching at undetected two-dimensional target displacements. *Journal of Neurophysiology*, *67*, 455–469.

Riegler, A. (2002). When is a cognitive system embodied? *Cognitive Systems Research*, *3*, 339–348.

Thomas, M. S. C., & McClelland, J. J. (2008). Connectionist Models of Cognition. In R. Sun (Ed.), *The Cambridge Handbook of Computational Psychology* (pp. 23–58).

Scheinerman, E. R. (1996). *Invitation to Dynamical Systems*. Prentice Hall.

Schöner, G., Haken, H., & Kelso, J. A. S. (1986). A stochastic theory of phase transitions in human hand movement. *Biological Cybernetics*, *53*, 247–257.

Schöner, G., & Kelso, J. A. S. (1988). Dynamic pattern generation in behavioral and neural systems. *Science*, *239*, 1513–1520.

Schutte, A. R., & Spencer, J. P. (2009). Tests of the dynamic field theory and the spatial precision hypothesis: capturing a qualitative developmental transition in spatial working memory. *Journal of Experimental Psychology. Human Perception and Performance*, *35*(6), 1698–725.

Stevens, S. S. (1957). On the psychophysical law. *The Psychological Review*, *64*(3), 153–181.

Trappenberg, T. P. (2010). *Fundamentals of Computational Neuroscience* (2nd ed.). Oxford, UK: Oxford University Press.

<2>General Information on Exercises

The exercises in this book use a collection of simulators programmed in the computing environment Matlab. They are available as Matlab code or as independent executable files for different operating systems. You can obtain all necessary files here:

http://www.dynamicfieldtheory.org

Additional instructions for running the simulations can also be found at that site.

Each simulator implements a specific dynamical system and provides a graphical user interface (GUI), which visualizes the state of the system and allows the user to set inputs or change system parameters. The modeled dynamical system is running immediately and continuously as soon as the simulator is started (that is, its state is regularly updated according to a set of differential equations). In some cases this may not be obvious from the visualizations, because often the dynamical system is initially in an attractor state and does not change unless external inputs are applied.

Each GUI provides a different set of plots and other visualizations, and a set of control elements to change system parameters. The most common control elements used in the exercises are sliders, which allow you to adjust a certain parameter smoothly within a specified range. Note that you can click on the arrows on either side of the slider to change the parameter value in small fixed steps, or click on the slider bar on either side of the slider to change the value in larger fixed steps. The latter method is often very useful, for instance to quickly apply several stimuli of exactly equal amplitude to the system. Parameter changes made through the control elements are applied immediately, and their effects can be observed in the visualizations.

Each GUI additionally contains a shared set of global control buttons that affect the overall behavior of the simulation. The `Pause` button suspends the continuous update of the dynamical system's state. You may still change parameters while the simulation is paused, but these changes will only take effect once the simulation is continued. The `Reset` button re-initializes all elements of the dynamical system, and in particular sets the activation of all dynamic variables and fields back to their resting levels. It does not change any parameter values (to return to the initial parameter values, you may quit and restart the simulation). The

`Parameters` button allows you to access the parameters of all elements in the dynamical system. It opens a parameter panel as a separate window. In this panel, you can first select an element via the dropdown menu at the top, then view and change its parameters, and apply these changes by clicking the `Apply` button (changes that are not applied are lost when the element selection is changed or the panel is closed).

The `Save` and `Load` buttons allow you to write the current parameter values to a text file in JSON-Format, and to retrieve parameter values from such a file. Only the parameters are stored or retrieved, not the state of the dynamical systems. In some simulators, these buttons are replaced by a preset selection element, consisting of a dropdown menu with a list of available presets and a `Select` button. When this button is clicked, the parameter file associated with the currently selected preset is loaded. Note that upon loading parameters by either of these methods, the simulation is always re-initialized, and the previous state of the dynamical system is lost.

Finally, the `Quit` button terminates the simulation. If you are running the simulation in Matlab, the state of the dynamical system is still available in the workspace afterwards in the form of a simulator object (named `sim` by default). A GUI object is also retained in the Matlab workspace, and the simulation may be restarted by calling `gui.run` in the Matlab command window. This continues the simulation in the same state that it had upon termination.

<2>Exercises for Chapter 1

All exercises use the interactive Matlab simulator `launcherTwoNeuronSimulator`. This program simulates two activation variables, informally called neurons, with external input, self-excitation, interaction and noise, as defined by the equations

$$\tau_1 \dot{u}_1(t) = -u_1(t) + h_1 + s_1(t) + c_{11} g\big(u_1(t)\big) + c_{12} g\big(u_2(t)\big) + q_1 \xi_1 \qquad \text{(A1.1)}$$

$$\tau_2 \dot{u}_2(t) = -u_2(t) + h_2 + s_2(t) + c_{22} g\big(u_2(t)\big) + c_{21} g\big(u_1(t)\big) + q_2 \xi_2 \qquad \text{(A1.2)}$$

In the initial parameter setting most of these terms are set to 0 so the actual behavior of each neuron is that of a single dynamic activation variable without self-excitation and input. You will use this simulator to explore the different dynamical systems analyzed in this chapter and get a more practical understanding of the role of each parameter for the system as a whole.

The simulator GUI shows five sets of axes, ten sliders to control parameters of the simulation, and five global control buttons. The sliders are used to modify the parameters of the dynamical system, one slider each for the resting level, h_1, the self-excitation strength, c_{11}, the strength of the interaction term, c_{12}, the variance of the noise, q_1, and the stimulus, $s_1(t)$, and the same for the second neuron. Note the naming convention for interactions between different activation variables or fields that is used throughout this book. The parameters for such interactions have a two-character index (e.g., c_{12}), with the first character specifying the target of the interaction (here activation variable u_1) and the second character specifying its source (here activation variable u_2).

The two rightmost sets of axes show phase plots for the two activation variables. The red line shows the rate of change for different activation values, as specified by Equations A1.1 and A1.2. The red dot indicates the current activation value and current rate of change of the activation variable, and attractor and repellor states in the dynamics are marked as squares and diamonds, respectively. The two sets of axes in the middle contain trajectory plots, showing the recent history of activation states for the two variables, with the present state indicated by the blue dot. The single set of axes on the left shows the trajectories of the two activation variables combined by plotting the activation of one variable against the activation of the other one, both for the current state (blue dot) and the recent history (blue line).

<3>Exercise 1: Single dynamic activation variable with input

Use the simulator to explore the dynamics of a single activation variable with variable input, as specified by $\tau_1 \dot{u}_1(t) = -u_1(t) + h_1 + s_1(t) + q_1 \xi_1$.

a) ***Tracking:*** Explore how the activation variable tracks a shifting input. Use the s_1-slider to set the input parameter to different values and observe how the zero-crossing of the phase plot of u_1 is shifted around. Observe how the state variable tracks the input by relaxing to the new attractor, both in the trajectory plot and the phase plot.

b) ***Relaxation time****:* Note how the state changes faster initially when the distance to the new attractor is larger, but the overall shape of the relaxation curve is always the same. Compare relaxation times for different values of τ: use the `Parameters` button to set τ_2 to a value that is significantly different from the value of τ_1 (to do this, select the corresponding node in the dropdown menu in the parameter panel, enter the desired value of τ, and click `Apply`). Use the same resting level and non-zero stimulus for u_1 and u_2, then `Reset` both activation variables to observe the differences in relaxation time. Do this for several different parameter settings.

c) ***Stability:*** Set the relaxation time parameters to *very* different values, e.g. 10 and 1000. Add a small amount of noise to both systems and observe how the activation variable with higher relaxation time deviates significantly further from the resting level and takes longer to return to it eventually (use $h = 0$, no input, $q_1 = q_2$). How is this effect reflected in the two-dimensional combined trajectory plot?

<3>Exercise 2: Dynamics of a single activation variable with self-excitation

Explore the dynamics of a single neuron with self-excitation, as specified by

$$\tau_1 \dot{u}_1(t) = -u_1(t) + h_1 + s_1(t) + c_{11} g\big(u_1(t)\big) + q_1 \xi_1$$

For this exercise, set the relaxation time parameters of both activation variables back to their initial values, $\tau_1 = \tau_2 = 20$, and set the resting levels back to $h_1 = h_2 = -5$. Start with a stimulus amplitude of zero.

a) ***Detection:*** Increase the self-excitation strength, c_{11}, of the activation variable to a medium value and note the nonlinearity emerging in the phase plot. Move the system through the detection instability by increasing the stimulus amplitude systematically. Move the system back through the reverse detection instability by decreasing the stimulus.

b) ***Hysteresis:*** Modify the self-excitation and stimulus to put the system u_1 into the bistable regime, then copy the parameter values to u_2 to create two identical systems. Demonstrate the hysteresis effect of this system by temporarily varying the stimulus of one system. After resetting the stimulus to the old value, the activation variables of these two identical systems should relax to different attractors.

c) ***Perturbations:*** Find parameter settings for a bi-stable system with moderate self-excitation, reset the system and let it relax to the off-attractor. Subject the system to a random perturbation by temporarily adding a lot of noise to the system. Does the system stay in the off-state after the perturbation, or switch to the on-state? Repeat this process several times and note the ratio of returns vs. switches. How does this ratio change when you vary the self-excitation strength?

<3>Exercise 3: Dynamics of two activation variables with mutual inhibition

Explore the dynamics of two neurons with mutual inhibition, as specified by Equations

$$\tau_1 \dot{u}_1(t) = -u_1(t) + h_1 + s_1(t) + c_{11} g\big(u_1(t)\big) + c_{12} g\big(u(t)\big) + q_1 \xi_1 \qquad \text{(A1.3)}$$

$$\tau_2 \dot{u}_2(t) = -u_2(t) + h_2 + s_2(t) + c_{22} g\big(u_2(t)\big) + c_{21} g\big(u(t)\big) + q_2 \xi_2 \qquad \text{(A1.4)}$$

a) ***Bistability:*** Set the interaction parameters of the system to mutual inhibition ($c_{12} = c_{21} = -10$). Add a stimulus to u_1, then to u_2. Remove the stimuli and reapply them in the opposite order. Note which attractor the system relaxes to in each case.

b) ***Selection:*** Add a small amount of noise to the system and give the same stimulus to both neurons. Reset the system several times to observe the selection decision the system makes. Change the relative strengths of the stimuli by a small amount and observe how the stronger stimulus is favored in the selection.

c) ***Biases:*** Reduce the inhibition of one neuron while keeping the other one invariant. How does this bias the selection decision and why?

[START BOX 1.1]

Box 1.1: Biophysics of neurons

We provide a brief review of the main biophysical features of neurons to establish links to the level of description used in this book. For textbook treatment of the biophysics of neurons see, for instance, Kandel, Schwartz, Jessell (2000) and, especially, Trappenberg (2012), where the link between the biophysical and the population level is addressed in some depth.

Neurons are electrically active cells that maintain an electrical potential across their membranes through ion pumps. Neurons have four functionally relevant components: (1) the axon is the output structure of the neuron and carries travelling excitations of membrane potential called spikes; (2) the soma is the core of the neural cell at which summation of inputs may lead to spike generation; (3) the dendritic tree that collects inputs in the form of membrane potential changes that happen at synapses and transports these to the soma; and (4) synapses, electro-chemical connections between the axons of pre-synaptic cells and the dendritic tree of the post-synaptic cell.

Across the membrane of neurons, a difference in ion concentration between the intracellular and the extracellular space gives rise to an electrical potential, called the membrane potential. The most relevant ions in this process are sodium and potassium, which are both positively charged. Membrane channels are proteins in the membrane that are specifically permeable to a particular type of ion, for instance, sodium or potassium. Membrane channels can be controlled electro-chemically to change configuration such that they are either open or closed. Ion pumps are another type of membrane protein, that use chemical energy to actively transport ions across the membrane against their electro-chemical gradient.

When there is no input to the membrane, the membrane potential is typically around -70 millivolts (intracellular versus extracellular space), the so-called resting potential. In this state, the sodium concentration is much higher on the outside of the axon than on its inside, while the potassium concentration is much higher on the inside. The excess negative charge on the inside stems from largely immobile negative ions and from a slight constant efflux of potassium ions through a few open potassium channels, openings in the membrane through which potassium ions can pass when a electro-chemical control system configures them appropriately. However, this efflux is largely counterbalanced by active sodium-potassium pumps, such that the resting potential is maintained at -70 millivolts. Importantly, the great majority of both sodium and potassium channels are closed while the membrane is at resting potential.

In most neurons in the higher-nervous system of mammals, neural processing is based on spikes. Spikes, also called action potentials, are brief, active changes of the membrane potential that travel along a neuron’s axon. A spike is triggered when the potential at a patch of axon membrane is increased above resting level (depolarized) to a certain threshold. This spike threshold typically lies about 15 to 20 millivolts above the resting potential. The initial depolarization is caused by a flow of ions from a neighboring area of the axon where an action potential is already in progress. When the threshold is reached, voltage-gated sodium channels open. This initiates an all-or-none cascade of events. First, a sodium influx occurs, depolarizing the membrane further, which in turn leads to the opening of even more sodium channels. The result of this positive feedback loop is a very quick depolarization far into the positive range, typically peaking at around +40 millivolts . However, the sodium channels become inactivated and thus impermeable shortly after this, preventing further depolarization. Concurrently, voltage-gated potassium channels are opened, allowing potassium ions to flow out of the axon. This potassium efflux repolarizes the membrane to slightly below the resting potential. This causes the potassium channels to close again, and the original distribution of ions is then restored by active ion-pumps.

The total duration of a spike often amounts to little more than one millisecond. However, the sodium channels cannot be activated for an additional time span of one or two milliseconds, the so-called refractory period, which limits the maximally possible spike frequency to around 500 Hz (less in many neurons). Importantly, because the absolute height of the initial depolarization does not affect the course of events once the threshold has been reached, spikes are virtually identical to each other in amplitude and duration, especially within the same neuron.

Finally, the propagation of spikes is based on currents along the length of the axon fiber, between an already depolarized patch of membrane and a neighboring membrane patch still at resting potential. These currents serve to depolarize the next axon patch to spike threshold. Most axons are wrapped into so-called myelin-sheaths, however, which consist of multiple layers of cell membrane, thus insulating the axon from extracellular space. The myelin-sheath is interrupted by gaps every millimeter or so, called nodes of Ranvier. Only at the nodes of Ranvier can spikes establish, while the current triggering the spike at the next node is conducted within the axon. This so-called saltatory conduction (from latin saltare, “to leap”) greatly accelerates nerve conduction velocity.

The conditions at the cell body (soma) and at the dendrites of a neuron are similar to those at axonal membranes. That is, the distribution of ions between the intracellular and extracellular space determines membrane potential, with sodium and potassium being most relevant, and a resting potential of around -70 millivolts. There is an important difference, though: Potentials at somatic and dendritic membranes are graded, which means that voltage can vary across a wide range without triggering an all-or-none chain of events like spikes (although some neurons are capable of developing spikes at these membranes as well).

Changes in somatic or dendritic membrane potential are induced by synaptic activity. Synapses are contact points between the axon of one neuron (the pre-synaptic neuron) and the dendritic tree of another neuron (the post-synaptic neuron). When a spike in a pre-synaptic neuron reaches the synaptic knob at the end of an axonal branch, neurotransmitters are released into the synaptic cleft. The transmitter molecules diffuse toward the membrane of the post-synaptic neuron where they bind to receptors embedded in the membrane, triggering the opening of ion channels. The binding works according to the key-lock principle, so that a given type of neurotransmitter specifically activates a particular type of channel. Thus, synaptic action can have different effects on the postsynaptic membrane potential depending on which transmitter is released by a synapse. Excitatory transmitters cause sodium channels to open. The ensuing sodium influx depolarizes the postsynaptic membrane, inducing the excitatory postsynaptic potential (EPSP). Inhibitory transmitters, on the other hand, cause potassium channels to open. The resulting potassium efflux hyperpolarizes the membrane, that is, it makes membrane potential more negative. This is known as the inhibitory postsynaptic potential (IPSP). Some inhibitory transmitters cause the opening of chloride channels, allowing an influx of chloride ions. As chloride ions are negatively charged, this likewise induces an IPSP. The size of the postsynaptic potential depends on the firing rate of the pre-synaptic neuron in the form of a sigmoidal function (although for many cortical neuron, the sigmoid saturates only for quite high pre-synaptic firing rates that are outside the normal physiological range).

Once a postsynaptic potential has been induced, it spreads across the dendritic tree to the cell soma, eventually reaching the axon hillock, the starting point of the axon where spikes are generated. As is the case on the axon itself, a spike is generated if membrane potential at the axon hillock reaches a threshold some 20 millivolts above the resting potential (note, however, that many neurons have spontaneous base firing rates). Hence, EPSPs increase the probability of spiking, whereas IPSPs reduce it.

Temporal summation of synaptic input occurs when multiple spikes arrive at a synapse in quick succession, so that the postsynaptic potentials induced by the individual spikes overlap in time and may thus add up to a larger change in membrane potential or, if they have different signs, to cancel each other out**.** Through temporal summation, a post-synaptic cell may be driven to spiking threshold when an EPSP induced by a individual spike may not be sufficient to do this. Conversely, summation of IPSPs lowers spiking probability more than a single IPSP.

Spatial summation refers to the same principle of summation at the point of spike generation when the EPSPs and IPSPs originate from different synapses across the dendritic tree. The arrangement of synapses on the dendritic tree may bring about non-trivial computation, such as shutting off the connections from a particular branch of the dendritic tree by an IPSPs down-stream from that branch (also called shunting inhibition, see Koch, 1999).

For postsynaptic potentials to be summed up, spikes need to arrive at axon hillock within a certain time window. The width of this time window depends on the time constant of the postsynaptic membrane (which in turn depends on properties of the membrane itself as well as on the state of the ion channels, determining membrane resistance and capacitance). The membrane potential evolves according to a dynamics much like the one postulated by in DFT, with a "$-u$" term determining an exponential decay toward the resting level. This can be observed in the laboratory when an electrode is inserted through the membrane into the cell, and a current is injected. The time scale of this exponential is slower for cortical neurons than for neurons in the periphery of the nervous system, making temporal summation more likely. Although spikes last only a millisecond, the integration time scale of cortical neurons is sufficiently slow to enable summation of incoming spikes that are separate by less then 10 milliseconds.

The neural dynamics at the population level that we model in DFT is characterized by this slower time scale of summation (see Trappenberg, 2012, for a deeper discussion of this link). This neural dynamics of populations of neurons can be derived mathematically from the biophysical dynamics of neurons under certain, restrictive conditions in the so-called mean-field approximation, in which the evolution of a population level "activation" is determined by the summed asynchronous spiking activity of neurons in the population (Faugeras, Touboul, Cessac, 2009). In that derivation, the basic form of the neural dynamics on which DFT is based, including the "$-u$" term, the resting level, and input as an additive contribution to the rate of change, is inherited from the biophysical level of description, but acquires a slower time scale when the averaging across the population happens. Similarly, the sigmoidal threshold function used at the population level is functionally analogous to the sigmoidal transfer function that describes the post-synaptic potential as a function of the pre-synaptic firing rate. Making that analogy concrete is not so easy, however, as these sigmoids link very different kinds of variables (spike rates to membrane potentials for the biophysical sigmoid, population activation to its rate of change for population level neural dynamics).

The mathematical derivation of the mean-field approximation is complex, and as a result it is not easy even to state how the population "activation" variable is computed from the spiking activities of all the neurons that contribute. At this point, there is no derivation of the neural dynamics at the population level that is general enough to cover the conditions under which we use the population description. We show in Chapter 3 that the activity of populations of neurons provides the best correlate of neural measures with measures of behavior. The neural dynamics on which DFT is based is a good phenomenological description of how the activity in populations of cortical neurons evolves in time under physiological conditions in which the brain is involved in perception and generates behavior. Although this phenomenological description has not been rigorously derived as an approximate description from biophysical neural dynamics under these physiological conditions, it is not excluded that this could be achieved in the future.

What properties of biophysical neurons are we leaving out in the population level neural dynamics of DFT? Clearly, we are not including discrete spiking events and spike times. The mean-field picture assumes that within a neural population, spikes are generated frequently enough and asynchronously enough to sample continuous time. It is possible, however, that for some neural mechanisms such as the detection of time differences (e.g., in the auditory system) or for learning (e.g., in spike time dependent plasticity), the timing of spikes plays a special role. Those would be cases in which the approximation on which DFT is based begins to break down. At this point, there is no clear empirical for a functional role of the spiking mechanism that would not be captured by population activation, but it is not excluded that such evidence may be discovered.

[END BOX 1.1]

[START BOX 1.2]

Box 1.2: Neural coding, feed-forward networks, and recurrence

The classical conception of feed-forward neural networks is illustrated in Figure 1.16. The connectivity among nodes u_i $(i = 1,2,\ldots,6)$ is ordered so that each neuron receives input only from neurons closer (in connections) to the sensory surface (described by input levels

s_1, s_2, s_3), or directly from the sensory surface itself. In such a forward network, the output neurons are those furthest removed from the sensory surface. Their output can be described as a function of the sensory inputs, subsuming all intermediate (hidden) neurons. In the illustration,

$$g(u_6) = \text{function}(s_1, s_2, s_3) \qquad \text{(B1.1).}$$

The function may be non-linear due to the sigmoidal threshold function for each neuron's output, but maps each input onto a unique output. If the function were invertible the network would implement a code, a one-to-one mapping between inputs and outputs. Close to the sensory periphery, where the networks are not deep, such invertible mappings are sometimes observed or postulated, leading to the notion of rate code: Each level of stimulus intensity is uniquely represented by a particular rate of neural firing. In general, however, the map is invertible, so that a many-to-one mapping may result. This is the case, for instance, when different patterns of input are mapped onto the same "response". Still, information theoretic terms are sometimes used to characterize such networks by saying that the output neurons "encode" particular patterns of input, perhaps with a certain degree of invariance, so that a set of changes of the input pattern do not affect the output. A whole field of connectionism or neural network theory is devoted to finding ways for how to learn these forward mappings from examples. A high point of that theory is the proof that certain classes of learning methods make such networks are universal approximators, that is, capable of instantiating any reasonably behaved mapping from one space to another (Haykin, 2008). In this characterization of a feed-forward neural network, time does not matter. Any time course of the input pattern will be reflected in a corresponding time course in the output pattern. The output depends only on the current input, not on past inputs or on past levels of the output or the hidden neurons.

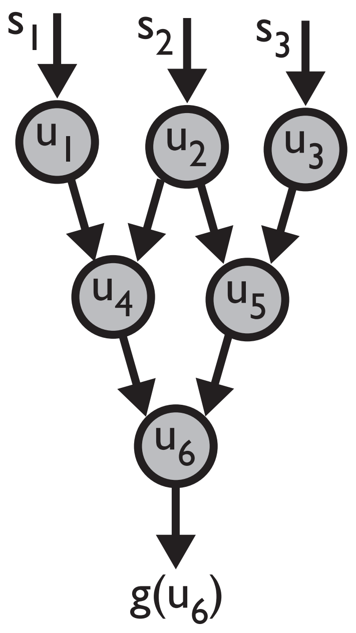

Figure 1.16 In this sketch of a feed-forward neural network, activation variables, u_1 to u_6, are symbolized by the circles. Inputs from the sensory surface, s_1 to s_3, are represented by arrows. Arrows also symbolize connections in which the output of one activation variable is input to another. Connections are ordered such that there are no closed loops in the network.

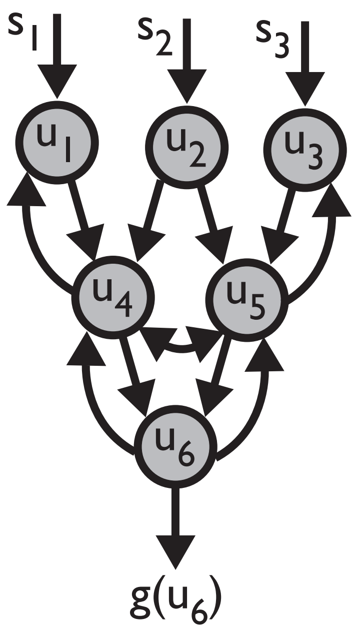

Figure 1.17 Same as Figure 1.16, but now with additional connections that create loops of connectivity, making this a recurrent neural network.

A recurrent network such as the one illustrated in Figure 1.17 cannot be characterized by such an input-output mapping. In a recurrent network, loops of connectivity can be found, so that one particular neuron (e.g., u_4 in the figure), may provide input to other neurons (e.g., u_6), but also conversely receive input from those other neurons either directly (u_6) or through some other intermediate steps (e.g., through u_6 and u_5 or through the chain from u_6 to u_5 to u_2 to u_4). The output cannot be computed from the input value because it depends on itself! Recurrence of this kind is common in the central nervous system as shown empirically through methods of quantitative neuro-anatomy (Braitenberg and Schüz, 1991).

To make sense of recurrent neural networks, the notion of time is needed, at least in some rudimentary form. For instance, neural processing in such a network may be thought of as an iteration process through time. From an initial level of activation, the activation level of all neurons is iteratively updated. At each time step, the output levels that provide input to a neuron are taken from the just previous iteration step. In a sense, such an iteration rule for the activation levels of all neurons represents a dynamical system, although in discrete time (Scheinerman, 1996). On the other hand, the synchronous updating of all neurons by some kind of clock cycle is not neurally realistic. There is no evidence for such updating across an entire network. Instead, as briefly reviewed in Box 1.1, neurons fire asynchronously, effectively sampling continuous time. The mathematical description of how activation evolves in recurrent neural networks in continuous time is exactly the neural dynamics discussed in the main text of this Chapter.

Recurrence and the neural dynamics it implies are no conceptually compatible with the information theoretical notions of encoding. In recurrent networks, there is no one-to-one or even many-to-one mapping from the stimulus space. The output of any neuron depends not only on the inputs to the network, but also on the current state of activation in the network, which reflects the recent history of activation and stimulation. Different histories of stimulation leading up to the same instantaneous stimulus lead to different activation patterns. Information-theoretical measures are still sometimes used to characterize recurrent neural networks as an approximate description (e.g., looking for how deep in time we need to go to extract how much information about a stimulus). In DFT we abandon this language, however, and emphasize, instead, the neural processes captured by neural dynamics.

[END BOX 1.2]

[START BOX 1.3]

Box 1.3: Dynamical Systems

The word “dynamics” has a variety of meanings. In music, for instance, dynamics refers to the varying levels of sound within a piece. A dynamic scenario in computer vision or robotics is simply a time-varying scenario. The word comes from the greek “dynamis” for “power” or “force”. In classical mechanics, dynamics refers to the core idea that movement can be explained and predicted from underlying causes, the forces that act on bodies. In modern mathematics, the theory of dynamical systems is a well-developed field with deep connections to other branches of analysis (see Perko, 2001, for an advanced, but pertinent treatment). This theory is the basis of most mathematically formalized models in the sciences, not only in physics and engineering, but also in chemistry, biology, economics, sociology and many other areas. Braun (1993) provides a highly accessible introduction to dynamical systems with an emphasis on such applications, giving a large number of examples.

The core idea of the theory of dynamical systems is that “the present predicts the future” given a “law of motion”, a dynamical law formalized as a dynamical system. To make that idea concrete, we first need to talk about variables and time courses. Think of a single variable, u, that characterizes the state of a system (we will say something about multiple variables at the very end of this box). In the main text, u is an activation level. In mechanics, u could be the position of a point mass along a line, e.g., along a vertical line when studying free fall. The variable is assumed to capture the evolution in time of a system by its time dependency, $u(t)$. Figure 1.18 illustrates such a time course, here in the form of an exponential function. The derivative of u, denoted by $\dot{u}$ or du/dt, is the rate of change of u, also illustrated in Figure 1.18. If u were the vertical position of a point mass, its rate of change would be the vertical velocity of the point mass. In the figure, as u decreases in time, its rate of change is negative. The decrease slows down over time, and thus the rate of change approaches zero from below. The time courses of the variable, $u(t)$, and of its rate of change, $\dot{u}(t)$, are correlated. Figure 1.18 visualizes this correlation by plotting $\dot{u}(t)$ against u. This reveals the functional relationship between the two quantities, $\dot{u}(t) = -u$.

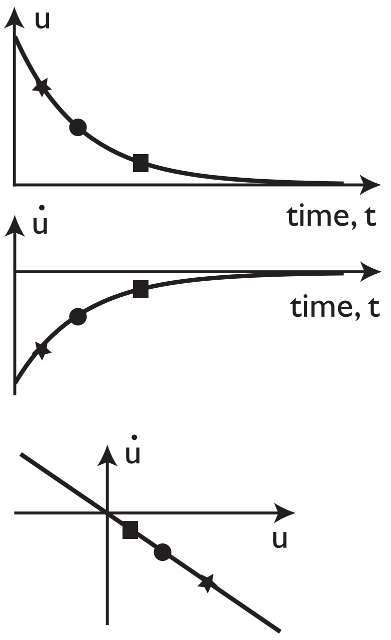

Figure 1.18 Top: The time course of a dynamic variable, u. Middle: The time course of its rate of change, $\dot{u}$. Bottom: The functional relationship between $\dot{u}$ and u obtained by correlating the two. The symbols in the three panels mark corresponding values of $\dot{u}$ and u at three points in time. The time courses on top were obtained from solutions of the linear dynamical system shown on bottom.

More generally, any functional relationship

$$\dot{u}(t) = f(u) \qquad \text{(B1.2)}$$

sets up a dynamical system through a differential equation. Figure 1.19 illustrates a general dynamical system characterized by a nonlinear function, $f(u)$. The core idea of dynamical systems theory is captured by the existence and uniqueness theorem, which says that for any sufficiently smooth function, $f(u)$, and any initial value of u, a unique solution, $u(t)$, of the differential equation exists for a interval of times, t. Thus, given the dynamics captured by the function, $f(u)$, "the present predicts the future". In the Figure, this is made plausible by marking an initial condition for u and highlighting the rate of change for that initial value. In this case, a negative rate of change, predicting an imminent decrease of the activation variable as indicated by the arrow pointing to the left. Thus, in a mental "iteration", we expect the variable to have a somewhat smaller value to the left of the initial value a moment of time later. The dynamics will then supply a new rate of change, which predicts the next value and so on.

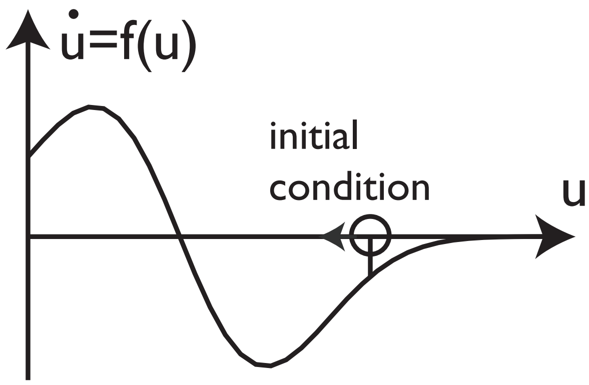

Figure 1.19 A nonlinear dynamics system $\dot{u} = f(u)$ with a particular value of u chosen as initial condition (open circle). The dynamics assigns a rate of change to that initial condition, which predicts the direction of change (arrow).

In the main text we use this form of iterative mental simulation to intuitively understand attractors, the convergence in time to a fixed point of the dynamical system. A fixed point, u_0, is formally defined as a solution of

$$f(u_0) = 0 \qquad \text{(B1.3)}$$

as illustrated in Figure 1.20. Because the function, f, does not depend on time, the fixed point, u_0, is constant in time as well, so that $\dot{u}_0 = 0$, and thus: $\dot{u}_0 = f(u_0)$=0. In other words, the fixed point, u_0, is a constant solution of the differential equation.

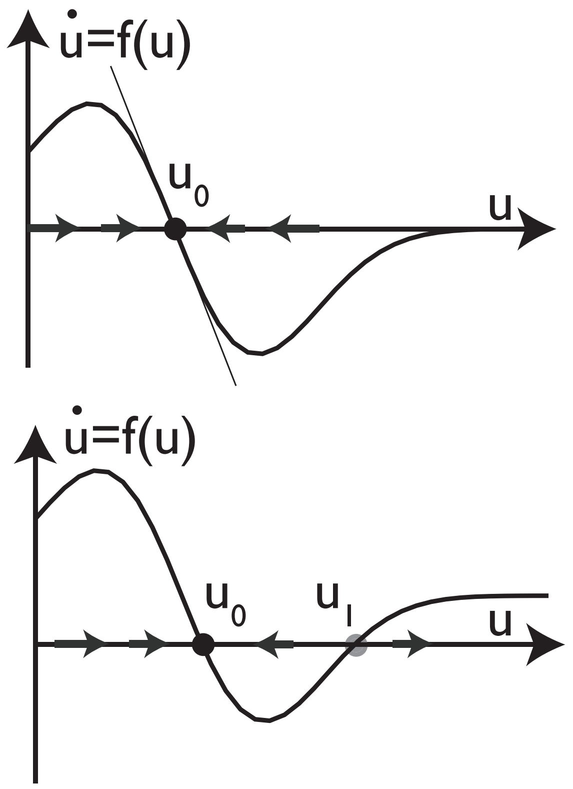

Figure 1.20 Top: The same nonlinear dynamics system of Figure 1.19 with the fixed point, u_0, marked by a filled circle. Arrows indicate the attraction to this asymptotically stable fixed point. The thin line illustrates the (negative) slope of the function, $f(u)$, at the fixed point. Bottom: The dynamics is changed (shifted upwards) and now has two fixed points, an attractor, u_0, on the left and a repellor, u_1, on the right.

A fixed point is "asymptotically stable", if the solutions of the dynamical system that start from initial conditions nearby converge in time to the fixed point. When the dynamics, f, has a negative slope at the fixed point, $\frac{df}{du}(u = u_0) < 0$, then the fixed point is stable. The arrows in Figure 1.20 remind you of the argument that we make in the main text: To the left of the fixed point, the positive rate of change makes for increase toward the fixed point, to the right of the fixed point, the negative rate of change makes for decrease toward the fixed point. An asymptotically stable fixed point is also called a "fixed point attractor" and sometimes just an "attractor" (there are a more complex limit sets that carry that name, but we will not concern ourselves with those in this book).

This mathematical concept of asymptotical stability is sometimes loosely referred to as "stability" by modelers, even though strictly speaking stability is a slightly different concept. Mathematically, a fixed point is "stable" when solutions that start nearby, stay nearby (but do not necessarily converge). Asymptotic stability implies stability, but not vice versa. This is important because instability is the opposite of stability, not of asymptotic stability. A fixed point is "unstable" if there are starting values arbitrarily close to the fixed point from which solutions move away from the fixed point. The lower plot in Figure 1.20 shows an unstable fixed on the right. In fact, this is a "repellor", a fixed point which all solutions starting nearby move away from.

This plot also brings home the important message that stability is a property of a fixed point, not of the entire dynamical system! There are two fixed points here, one stable, the other unstable. Sometimes, researchers talk about "stable" systems. This is a loose way to talk about a system that has a single fixed point, which is stable. Linear systems, in particular, can have only a single fixed point (because a straight line can only go through zero once). Because a lot of systems encountered in modeling are linear or are approximated as linear, it happens quite often that there is a single fixed point, hence this loose talk about the "stability of the system".

In non-linear dynamical systems, the fixed points and their stability organize the ensemble of all solutions of the dynamical system. This ensemble is called the "flow" and can be thought of as a mapping from all initial conditions to the states the solutions lead those initial conditions to a given time, t, later. For the dynamical system on the bottom of Figure 1.20, for instance, all initial conditions to the left of the repellor will be mapped onto values increasingly (with increasing time) close to the attractor on the left. All initial conditions to the right of the repellor will be mapped onto increasingly large values of u (which will go to infinity when time goes to infinity). The qualitative theory of dynamical system is aimed to characterize the flow of dynamical systems rather than to solve analytically specific equations. Most textbooks on differential equations focus on solving equations, but the books cited earlier in this box address the qualitative theory of dynamical systems (as does Scheinerman, 1996, a good elementary text that is provided freely online by its author). In the qualitative theory of dynamical systems, flows that are merely slight deformations of each other are all considered to be equivalent (the technical term is "topologically equivalent"). For instance, if we deform the function, f, in the bottom of Figure 1.20 a little bit, but not enough to remove the two fixed points or to change the signs of the slope of f around each fixed point, then the precise time courses of solutions would change, but that change would be minor. Solutions of the original and of the deformed dynamical system could be mapped onto each other such that neighboring solutions in the original system remain neighbors in the deformed system and vice versa (this is topological equivalence). In contrast, the dynamical system on top in Figure 1.20 is not topologically equivalent to the one on bottom. You can see this by looking at solutions for the system on top that start just to the left and just to the right of the location where the repellor is in the bottom system. Those solutions stay close to each other over time for the top system, while they diverge from each other for the bottom system.

When we model neural processes we essentially form hypotheses about categories of solutions, different stable states and how they are connected. This amounts to making assumptions about the flow, the ensemble of all solutions. That is why the qualitative theory of dynamical systems is of interest to us. Qualitatively different flows are often separated by instabilities, which we will look at next. Instabilities thus demarcate regimes with qualitatively different solutions and that is why instabilities are of so much interest to us in DFT.

Instabilities are changes in the number or stability of fixed points. The change comes from some parametric change of the dynamics, that is, of the function, f. We think of such changes as being smooth, that is, the function, f, changes continuously as a continuous parameter is changed. In the main text, input strength is such a parameter, for instance. Even though the function, f, changes smoothly, the solutions may change abruptly, and that happens exactly at instabilities. Figure 1.21 illustrates how this may happen. Here we have taken the portion of the dynamics depicted at the bottom of Figure 1.20 that contains the two fixed points and applied a continuous parameter that shifts the dynamics, f, upwards (f is shown only for three values of that parameter). As this happens, the attractor on the left and the repellor on the right move toward each other, until they collide, forming a single fixed point that is now unstable. At slightly larger values of the parameter, the fixed point is gone! So the stability of a fixed point has changed (attractor to unstable fixed point) and the number of fixed points has changed (from two to zero). This is the "tangent bifurcation" that we also discuss in the main text. The word "bifurcation" is a mathematical term for the looser term "instability" that more commonly used by physicists and modelers. Why instability is a good term is intuitive from Figure 1.21: Just as the bifurcation occurs, the two fixed points collide, the slope of the function, f, at the remaining fixed point becomes zero! So the stability criterion starts to fail at this point.

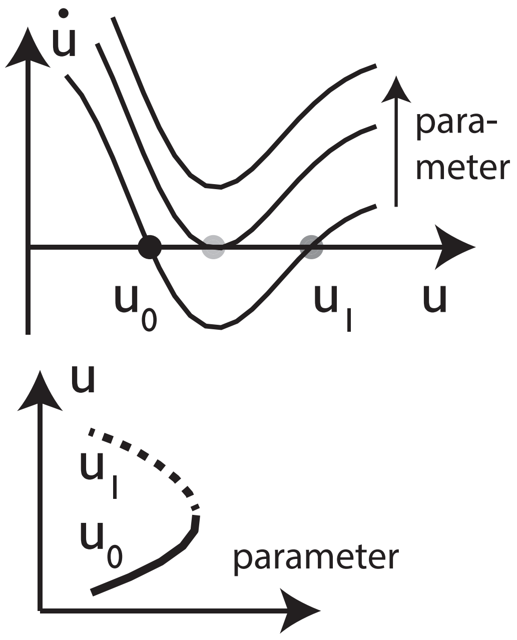

Figure 1.21 Top: A part cut out of the dynamics on bottom of Figure 1.20 is further changed by adding constant parameter to the dynamics. The dynamics at three values of this additive parameter is shown (see text). Bottom: A bifurcation diagram of the dynamics shown on top plots the fixed points of the dynamics as a function of the parameter that changes the dynamics. The two fixed points collide and then disappear as the additive constant parameter increases.

A theorem by Hopf has classified instabilities in dynamical systems using concepts that we will not discuss here. In that classification, the tangent bifurcation is the simplest and most generic instability and most of the instabilities encountered in DFT are tangent bifurcations (the only exceptions arising from special symmetries in a dynamics). What the theory of bifurcations, and more generally, the qualitative theory of dynamical systems helps us modelers with, is solving the problem of inverse dynamics. Forward dynamics means solving a given differential equation, and that is what most textbooks focus on (we do this mostly by numerical methods, see Appendix 1). Inverse dynamics is finding the right differential equation given some assumptions about its solutions. We typically make assumptions about attractors and how their number and stability change as conditions are changed. We can then use bifurcation theory to decide if a particular class of dynamical systems correctly captures that layout of the solutions.

This tutorial box only provides the most basic ideas. In particular, we have been referring to a single variable, u, and its dynamics, $\dot{u} = f(u)$. Most of the time in DFT we have many variables, in fact, conceptually we have infinitely many variables described by entire functions, $u(x)$. The ideas sketched here do carry over into higher dimensions, but the level of mathematics required is more advanced. Fixed points are still fixed points in higher dimensions. The slope of

f is replaced by the real parts of the eigenvalues of the matrix that linearizes f around the fixed point. Attractors are separated not by simple repellors but by lines or surfaces that are invariant solutions, unstable manifolds. But these changes are primarily technical in nature. The only thing that is qualitatively new when we move beyond a single dimension is the occurrence of more complex attractors such as periodic solutions (limit cycle attractors) and more complex bifurcations. In this book we manage to stay away from those, although they do play a role in understanding coordination, for instance (Schöner, Kelso, 1986; Kelso, 1995).

[END BOX 1.3]

[START BOX 1.4]

Box 1.4: Stochastic dynamical systems and their numerical solution

Noise is important in neural dynamics. First of all, one of the salient features of real neural networks is that neural activity is noisy, whatever the cause. Behavioral data are also noisy: Performance varies from trial to trial. Such behavioral variance is an important diagnostic of the underlying dynamics. Models that account for the variability of behavior are stronger than models that predict only the average performance. More specifically, the neural dynamics we use in DFT goes through instabilities. Near instabilities, the neural dynamics is sensitive to noise: A small random perturbation may kick the system out of an attractor that is close to becoming unstable and thus induce a transition to another stable state. Thus, in our modeling we must address noise explicitly.

Mathematically, variability is a topic of probability theory. Combining probability theory with dynamics requires the relatively advanced mathematical techniques of stochastic differential equations (Gardiner, 2009), but fortunately we really only need the simplest case, which can be grasped quite intuitively. The idea is that noise acts as a contribution to the dynamics that is additive, white, and Gaussian. Formally,

$$\dot{u} = f(u) + q\xi(t) \qquad (B1.4)$$

where $f(u)$ is the deterministic portion of the differential equation, q is the noise strength, and $\xi(t)$ is a Gaussian white noise process. First, the noise is *additive*, which really means that its influence is independent of the current level of activation, u. That is a reasonable first approximation. Even if the source of noise was sensitive to the level of activation (e.g., more noise at higher levels of activation as in a Weber law), there would not be any level of activation at which noise is zero. So we are modeling that base level noise that is common across activation levels. Second, the noise is *white*. That means that the noise, $\xi(t)$, at one particular moment in time, t, is statistically independent of the noise, $\xi(t')$, at any other time, t'. This expresses that the contributions of noise to the dynamics are truly random. If there was any dependency across different times, then that would be a deterministic contribution to the dynamics that should have been included in the deterministic portion, f, of the dynamics. Third, noise is *Gaussian*. This means that the distribution of the noise at any moment in time is a Gaussian distribution with zero mean, $< \xi(t) > = 0$. The joint probability distribution of the noise at different moments in time factorizes into Gaussian distributions, that are all generated from the two-point correlation function, $< \xi(t)\xi(t') > = \delta(t - t')$, for two times, t, and t'. The delta-function is zero whenever the two times differ (consistent with the independence at different moments in time: for Gaussian processes, statistical independence is the same as being uncorrelated). The delta-function at the point when both times coincide is infinite, but its integral over time is 1. Obviously, this third property of noise is a bit more technical. It comes, ultimately, from the central limit theorem of probability theory. The idea is the noise comes from many independent sources of randomness, all independent of each other, but having the same distribution. The theorem says, intuitively speaking, that the superposition of such noise sources is Gaussian distributed. In the nervous system, it is easy to imagine that noise comes from many different sources, e.g., variations in membrane potential and synaptic activity across the many neurons, about 10000 on average, that project onto any given cortical neuron.

The upshot is, therefore, that noise adds a random component to the rate of change that give the activation variable a kick that is uncorrelated at every moment in time. The activation variable itself evolves by integrating over time across these random kicks. We illustrated this in Figure 1.2 for the case that $f(u) = 0$, that is, for a purely stochastic dynamics. The simulation shown in that figure is the result of integrating across the Gaussian white noise process. This leads to a time continuous process, called the Wiener process, that is still very random because its increments are independent of each other. That is, at any moment in time, the direction of change is independent of the current level of activation. We used this insight in Figure 1.2 to argue for a deterministic portion, $f(u)$, of the dynamics that limits variance by introducing stability. This was done in Figure 1.3, in which $f(u) = -u + h$.

Conventionally, the source of randomness, the stochastic perturbation on the right hand side of the dynamics, is referred to as "noise". The consequence of randomness is variability of the solutions of the stochastic dynamics. That variability is referred to as "fluctuations". Not all authors strictly adhere to that convention, however. Essentially all the models we use in DFT have a noise component and are thus stochastic differential equations. In many cases we compare the fluctuations of the time courses obtained from the stochastic dynamics to variability across time or trials observed in experiment. In some instances, those comparisons led to quantitative match and predictive power (e.g., Schöner, Haken, Kelso, 1986; Schutte, Spencer, 2009).

The numerical solution of stochastic differential equations differs a little bit from the numerics of deterministic differential equations. Before we review that, we use the opportunity to deepen the discussion of numerics a little, which we didn't do in the main text so as not to break the flow. Numerics is an issue for the modeler, of course, not for the nervous system. The nervous system is essentially an analogue computer that implements neural dynamics directly (although that implementation is not trivial either, using spikes, as we briefly discussed in Box 1.1). But we as modelers solve the dynamical equations numerically on digital computers when we run simulations to account for neural or behavioral data. When we use the neural dynamics to drive robots that behave autonomously based on their own sensory information (as in Chapters 4, 9, 12, and 14) we do the same: The robots have on-board computers, on which we solve the equations in real time, taking inputs from the sensors and sending the computed solutions to the actuators. On computers, time is discrete. The computer goes through computational steps, paced by its clock. The time step available to us at the macroscopic level at which we write our code is much, much larger than the clock cycle on the hardware (e.g., somewhere around 10 to 50 milliseconds for our computational cycles compared to 1 millionth of a millisecond for the hardware clock cycle on a 1 GHz processor).

How to approximate the continuous time dynamics in discrete time is the topic of numerics, a well-established field of applied mathematics. For numerical solutions of deterministic differential equations consult, for instance, Braun (1993); For numerical solutions of stochastic differential equations, consult Kloeden and Platen (1999). Here we outline the main ideas only.

Let us say we want to numerically solve this differential equation, the deterministic version of Equation (B1.4):

$$\dot{u} = f(u). \qquad (B1.5)$$

We assume that we have a computational cycle that allows us to provide estimated values, $u(t_i)$, of the time course of $u(t)$ at the discrete times, $t_i = i\,\Delta t$. Here, Δt, is the time step and we have used an index, $i = 0,1,2,3\ ...$, to count the discrete time events. The classical and simplest approach is called the Euler method and is based on approximating the derivative, $\dot{u}$, around one of the sample times, t_i, by the differential quotient:

$$\dot{u}(t_i) \approx \frac{u(t_i) - u(t_{i-1})}{\Delta t} \qquad (B1.6)$$

If you don't remember this from high-school, look it up, even on Wikipedia. It is easy to figure out. If you insert this into Equation (B1.5), multiply by Δt and add $u(t_{i-1})$, you obtain the Euler formula:

$$u(t_i) = u(t_{i-1}) + \Delta t\, f\big(u(t_{i-1})\big). \qquad (B1.7)$$

(In this derivation, you will first find that the function $f(u(t_i))$ on the right hand side should be taken at the current time step, t_i. That leads to the "implicit Euler" method. When the time step is sufficiently small, we may approximate this value of the function by its value at the previous time step, $f(u(t_{i-1}))$, as in Equation B1.7). This is easy to implement in a numerical program: Initialize the time series by setting $u(t_1)$ to the initial condition. Then loop through the discrete times, computing at each iteration step the next value of $u(t_i)$ based on Equation (B1.7), which makes use only of the just previous value, $u(t_{i-1})$. The time step, Δt, must be chosen small enough that it can sample the time courses of activation. Near an attractor, the time scale of $u(t)$ is given by the relaxation time, τ, illustrated in Figure 1.5. The time step needs to be smaller than the relaxation time: $\Delta t \ll \tau$. In practice, our neural dynamics is most of the time close to an attractor, the stability of which helps keep the numerics stable. We often get away with an Euler step that is only about 10 times smaller than the relaxation time.

When noise comes into the picture, things are a bit different, a fact sometimes overlooked by modelers. The Euler formula for the stochastic differential equation (B1.4) reads:

$$u(t_i) = u(t_{i-1}) + \Delta t\, f\big(u(t_i)\big) + \sqrt{\Delta t}\, q\, \xi(t_{i-1}). \qquad (B1.8)$$

Note that the noise term scales differently than the deterministic term with the Euler step, Δt.

There are much better numerical procedures for solving deterministic differential equations. These get away with a larger Euler step to achieve the same precision. In fact, MATLAB considers the Euler method so outdated, that it doesn't include the Euler algorithm any longer in its library (it is easily programmed by hand, of course). In practice, we still use this simplest and worst (from the point of view of numerics experts) algorithm. First of all, it is good enough. Second, it lends itself to implementation on robots, on which we also take sensor readings at every time step. The more advanced algorithms take into account multiple samples of the dynamical variable at multiple time steps, and many also vary the time step, Δt, depending how strongly the solution varies. Neither is well suited to updating the sensor data. For sensor data, we want to go as fast as we can to track any changes in the input. So we are not so interested in using the largest Euler step that delivers acceptable precision. A final issue is that the more advanced methods for stochastic differential equations are quite complex, requiring a considerable number of estimates and auxiliary variables to be iterated. Although those methods scale better with the time step in principle, the amount of computation needed at each time step can be quite large, more than offsetting the advantage gained by the larger Euler step.

[END BOX 1.4]