



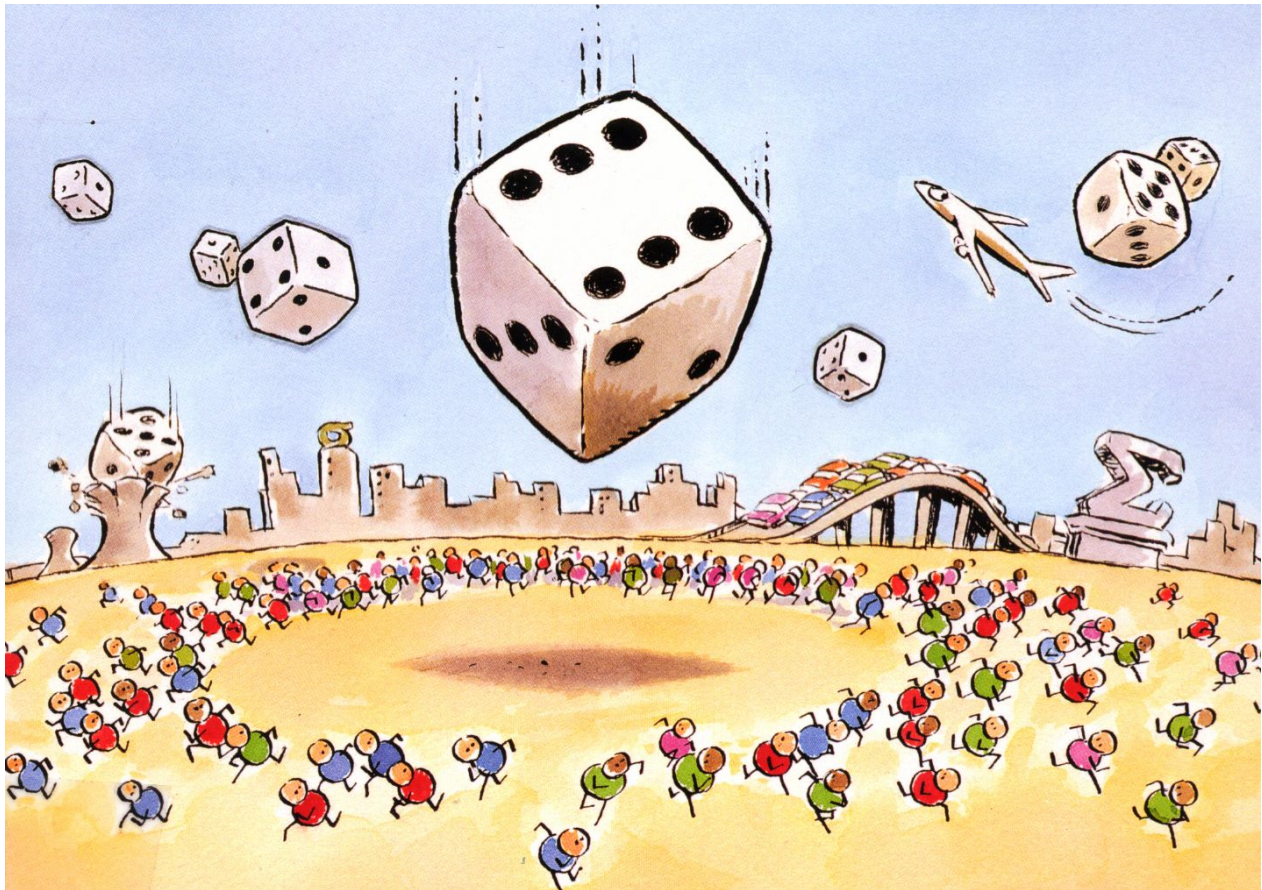
# Computational Neurosciences

## (MSc Neurosciences, WS+SS; MSc Physics, SS)

### Version V2.13 (June 10, 2022)

The Mini-SCRIPT! With loot from:  
...Dayan/Abbott, Hertz/Krogh/Palmer  
...lots of Internet sources and own publications!

Dr. Udo Ernst



Handle with care, it's got lots of maths and probs inside!!!

This page intentionally left blank!

# Contents

<b>1</b>	<b>SPIKES, RATES, AND THE DELTA-FUNCTION</b>	<b>9</b>
1.1	Spikes and spike train representations . . . . .	9
1.2	The 'delta' function . . . . .	9
1.3	Spike trains and rates . . . . .	10
1.4	Rate estimation by convolution . . . . .	13
1.5	Back to Discretized Spike Trains . . . . .	13
<b>2</b>	<b>STATISTICS OF SPIKE TRAINS</b>	<b>15</b>
2.1	Variability and Distributions . . . . .	15
2.1.1	Bernoulli-Process . . . . .	15
2.1.2	Poisson-Process . . . . .	16
2.1.3	Gaussian-/Normal Distribution . . . . .	16
2.1.4	Comparison of Bernoulli, Poisson and Gauss . . . . .	17
2.1.5	Interspike-Intervals and Exponential Distribution . . . . .	17
2.2	Descriptive Statistics . . . . .	18
<b>3</b>	<b>DECODING</b>	<b>20</b>
3.1	Overview . . . . .	20
3.2	Receiver-Operator Characteristic (ROC) . . . . .	21
3.2.1	Errors and correct decisions . . . . .	21
3.2.2	Maximizing Performance . . . . .	21
3.2.3	Two samples . . . . .	22
3.2.4	Special case: Gaussians . . . . .	23
3.3	Bayesian Estimation . . . . .	23
3.3.1	The Loss Function . . . . .	23

3.3.2	Bayesian inference . . . . .	26
3.3.3	Loss functions: Continuous variables . . . . .	26
3.3.4	Loss functions: Discrete variables . . . . .	27
3.3.5	Caveats of Bayesian estimation . . . . .	28
3.3.6	Extensions . . . . .	28
<b>4</b>	<b>FREQUENCY ANALYSIS</b>	<b>30</b>
4.1	Fourier Analysis . . . . .	30
4.1.1	Continuous sine/cosine Fourier transform . . . . .	30
4.1.2	Continuous Fourier transform, complex representation . . . . .	31
4.1.3	Discrete Signals . . . . .	33
4.2	The Convolution Theorem . . . . .	34
4.2.1	Non-periodic Functions . . . . .	35
4.2.2	Convolutions in Computational Neuroscience . . . . .	36
4.3	Wavelet Analysis . . . . .	39
<b>5</b>	<b>GENERALIZED LINEAR MODELS (GLMs)/LINEAR-NONLINEAR-POISSON MODELS (LNPs)</b>	<b>41</b>
5.1	Linear Superposition of Elementary (Pulse-)Responses . . . . .	41
5.2	Non-linear Transfer Function and Stochastic Output Spikes . . . . .	42
5.3	Spike-triggered average and reverse correlation . . . . .	43
5.4	Multivariate Inputs/Multidimensional Receptive Fields . . . . .	45
5.5	Abstracting Receptive Fields: Tuning Curves . . . . .	46
<b>6</b>	<b>FROM MEMBRANE POTENTIALS TO THE INTEGRATE-AND-FIRE NEURON</b>	<b>47</b>
6.1	The integrate-and-fire neuron . . . . .	47
6.1.1	Analytical Solution . . . . .	47

6.1.2	Examples – How the IAF neuron works . . . . .	48
6.2	Cell membranes, ions, currents and potentials . . . . .	49
6.2.1	Leakage currents . . . . .	49
6.2.2	Kirchhoff’s Law and Membrane Potential Dynamics . . . . .	52
6.2.3	Computing the resting potential . . . . .	54
<b>7</b>	<b>CONDUCTANCE-BASED NEURON MODELS: FROM HODGKIN-HUXLEY TO FITZHUGH-NAGUMO</b>	<b>55</b>
7.1	Hodgkin-Huxley Model . . . . .	55
7.1.1	Voltage-gated ion channels . . . . .	55
7.1.2	Hodgkin-Huxley Parameters/Equations . . . . .	56
7.1.3	Simplifying Hodgkin-Huxley . . . . .	56
7.2	The Fitzhugh-Nagumo neuron . . . . .	57
<b>8</b>	<b>AXONS, DENDRITES AND SYNAPSES</b>	<b>59</b>
8.1	Axonal and Dendritic Transmission . . . . .	59
8.1.1	Passive cable equations . . . . .	59
8.1.2	Solving the passive cable equations . . . . .	61
8.1.3	Nonlinear extensions . . . . .	62
8.1.4	Unmyelinated axons . . . . .	62
8.1.5	Myelinated axons . . . . .	62
8.1.6	Multicompartmental models, numerical methods, and boundary conditions	63
8.2	Synapses . . . . .	64
<b>9</b>	<b>COLLECTIVE PHENOMENA IN SPIKING NETWORKS</b>	<b>67</b>
9.1	Synchronization of pulse-coupled oscillators . . . . .	67
9.1.1	Motivation . . . . .	67

9.1.2	Oscillator model . . . . .	68
9.1.3	Fixed points . . . . .	70
9.1.4	Considering delays $\tau$ . . . . .	73
9.2	Stochastic drive and irregular firing . . . . .	77
<b>10</b>	<b>POPULATION DYNAMICS</b>	<b>82</b>
10.1	Simplified Wilson-Cowan Dynamics . . . . .	82
10.1.1	Heuristic derivation . . . . .	82
10.1.2	Gain functions . . . . .	83
10.1.3	Example: Cortical Microcircuits . . . . .	84
10.2	Wilson-Cowan equations . . . . .	85
10.2.1	Full dynamics . . . . .	85
10.2.2	Time coarse-grained dynamics . . . . .	87
10.3	Analyzing simple population models . . . . .	88
10.3.1	Population dynamics: one subpopulation . . . . .	88
10.3.2	Population dynamics: two coupled subpopulations . . . . .	89
10.3.3	Limitations of the simplified dynamics . . . . .	91
10.4	Spatially extended systems . . . . .	91
10.4.1	Cortical dynamics . . . . .	94
10.4.2	Constant input . . . . .	95
10.4.3	Spatially inhomogeneous input . . . . .	96
10.4.4	Two-dimensional activation dynamics . . . . .	97
10.4.5	Example: Orientation tuning . . . . .	98
<b>11</b>	<b>COMPUTATION AND CLASSIFICATION</b>	<b>100</b>
11.1	Single-Layer Perceptrons . . . . .	100

11.1.1	Feedforward Networks . . . . .	100
11.1.2	Definition of the Perceptron . . . . .	101
11.1.3	Computation . . . . .	101
11.1.4	Learning . . . . .	103
11.1.5	Continuous output functions and gradient descent . . . . .	104
11.2	Multilayer Perceptrons . . . . .	105
11.2.1	The Backpropagation Algorithm . . . . .	106
11.2.2	Some (General) Remarks on Learning . . . . .	108
11.3	(K)-Nearest-Neighbor Classification . . . . .	109
11.4	Support Vector Machines . . . . .	109
11.4.1	Maximizing the Margin . . . . .	110
11.4.2	Dealing with Non-Separable Cases . . . . .	112
11.4.3	The Kernel Trick . . . . .	113
11.4.4	Examples for Kernels and SVMs . . . . .	114
<b>12</b>	<b>MEMORY AND LEARNING</b>	<b>115</b>
12.1	The Hopfield Model . . . . .	115
12.1.1	The Paradigm . . . . .	115
12.1.2	Choosing the weights . . . . .	116
12.1.3	Storage capacity . . . . .	117
12.1.4	Energy function of the Hopfield network . . . . .	117
12.1.5	Further aspects . . . . .	119
<b>13</b>	<b>APPENDIX</b>	<b>120</b>
13.1	Stability analysis . . . . .	120
13.1.1	Fixed points . . . . .	120

13.1.2	Stability of $N$ -dimensional systems . . . . .	121
13.1.3	Stability of one-dimensional systems . . . . .	122
13.1.4	Stability in more than one dimension . . . . .	123
13.1.5	Stability of two-dimensional systems . . . . .	126
13.2	Differential equations . . . . .	130

# 1 SPIKES, RATES, AND THE DELTA-FUNCTION

## Motivation:

Action potentials, or spikes, are the generic currency with which networks in the brain process and exchange information. Spikes are pulse-like events which occur on short timescales, and specific methods for working with such events are required. Aim of this chapter will be to develop a mathematical language to handle spikes, and provide fundamental tools for data analysis and network modeling.

## 1.1 Spikes and spike train representations

Information transmission/processing in the brain relies (not entirely, but heavily) on spikes. The form of the spike can for most purposes be considered to be not important.

How do we describe temporal sequences of spikes ('spike trains')? There are two possibilities:

(a) **Binned representation (discretized in time, e.g. measured and sampled):**

Consider time to be discretized in steps of  $\Delta t$ : A spike train can be described by a vector  $\vec{s}$  with components  $s_j, j = 0 \dots \infty$ ,  $s_j = 1$  if spike was in interval  $t \in t_0 + [j, j + 1]\Delta t$

**Problem:** this representation depends on the chosen discretization (example: imagine a smaller  $\Delta t$ ).

(b) **Idealized representation (in continuous time):**

Spikes are represented by infinitely short pulses, only spike times matter:  $t_1, t_2, \dots, t_k, \dots$

**Advantage:** spike times are defined precisely, and if we need to count spikes we can perform integration instead of having to work with sums (i.e., technically less difficult).

**Problem:** Which mathematical function could we use for representing spikes in continuous time?

$\Rightarrow$  Here we need to introduce a novel, formal method to work with pulse-like events: the "Delta"-function!

## 1.2 The 'delta' function

For representing spikes in continuous time, we need some function that can be very "thin", but has a finite integral of 1. Such a definition will allow us to count the spikes in a spike train by performing a simple integration over time.

Two examples how such a function might be constructed:

(a) Consider a rectangular function :

$$g(t) = 0 \quad \text{for } t > |\Delta t|/2 \quad (1)$$

$$g(t) = \frac{1}{\Delta t} \quad \text{for } t \leq |\Delta t|/2 \quad (2)$$

If we let  $\Delta t$  go towards 0, the function gets more and more 'spikey', but its integral is always well defined, and its value is always 1.

(b) As a second example, consider this triangular function defined between  $-b$  and  $b$ :

$$f(t) = \frac{t+b}{b^2} \quad \text{for } t < 0 \quad (3)$$

$$f(t) = \frac{-t+b}{b^2} \quad \text{for } t \geq 0 \quad (4)$$

Now, let  $b$  go to 0...

$\Rightarrow f(t)$  becomes also weird! - This function itself is not really well-defined, but again its integral exists and its value is 1.

**Idea:** We define the  $\delta$ -function by its integral.

**The Delta ( $\delta$ -)function:**

$$\int_G \delta(x) dx = \begin{cases} 1 & \text{if } 0 \in G \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

**Rules for computing with the  $\delta$ -function:**

$$\text{Rule \#1:} \quad \int_G h(x') \delta(x - x') dx' = h(x) \text{ if } x \in G \quad (6)$$

$$\text{Rule \#2:} \quad \delta(ax) = \frac{\delta(x)}{|a|} \quad (7)$$

This is everything we need for computing with spikes.

### 1.3 Spike trains and rates

We will now compute firing rates from spike trains, learn different definitions of firing rates, and train how to use the newly introduced  $\delta$ -function.

Express spike trains as a sequence of delta pulses:

$$\rho(t) := \sum_{i=1}^n \delta(t - t_i) \quad (8)$$

$\implies$  Well-defined, since the integral over  $\rho$  is  $n$  (# of events in spike train).

There are three (slightly) different ways to define rates: the **spike-count rate**  $R$ , the **instantaneous rate**  $r(t)$ , and the **average or mean firing rate**  $\langle r \rangle$ :

(a) **Spike-count rate:** Consider the spike count  $K$  per time interval  $T$ :

$$K := \int_0^T \rho(t) dt \quad (9)$$

$$R := \frac{K}{T} \quad \text{units: [Hz]} \quad (10)$$

The spike-count rate can be used, for example, to describe how many events have been observed in one “trial” of an experiment. **Problem:** This measure can not represent how the spike rate might vary over time within  $T$ , so, what would be the rate at one specific point in time?

(b) **Instantaneous firing rate:** consider average spike count per (infinitesimally small) time window  $\Delta t$ . To get a good statistics, requires typically to average over “repetition of trials”  $m = 1 \dots M$ :

$$r_{\Delta t}(t) := \frac{1}{\Delta t} \int_t^{t+\Delta t} \frac{1}{M} \sum_{m=1}^M \rho_m(t') dt' \quad (11)$$

$$r(t) := \lim_{\Delta t \rightarrow 0} r_{\Delta t}(t) \quad (12)$$

**Problem:** meaningful limes for  $\Delta t \rightarrow 0$  needs  $M \rightarrow \infty$ , hence, it requires lots of observations for achieving a high temporal resolution...

(c) **Average/Mean firing rate:** consider average spike count per time interval  $T$ :

$$\langle r \rangle := \frac{1}{T} \int_0^T \frac{1}{M} \sum_{m=1}^M \rho_m(t') dt' \quad (13)$$

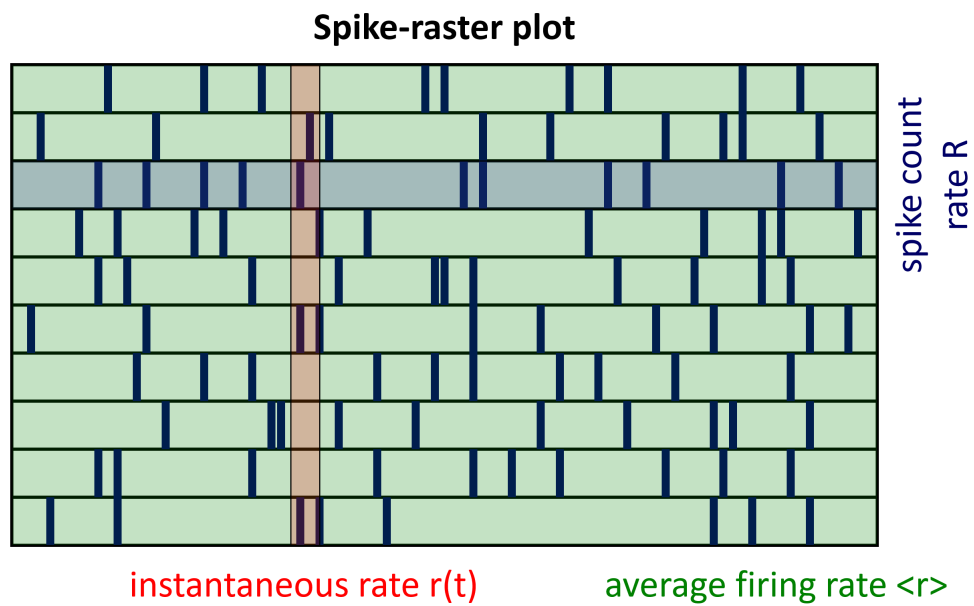
$$= \frac{1}{T} \sum_j r_{\Delta t}(t_j) \Delta t \quad \text{which becomes with } \lim_{\Delta t \rightarrow 0}: \quad (14)$$

$$= \frac{1}{T} \int_0^T r(t) dt \quad (15)$$

Its advantage is a good compromise between temporal resolution and keeping the required number of observations low. There is a “tradeoff” between increasing the temporal

resolution by minimizing  $T$  and minimizing the required number of observations  $M$ .

**Remark:** spike times usually vary from trial to trial, so the instantaneous rate can also be interpreted as a firing probability.



## 1.4 Rate estimation by convolution

**Problem:** In an experiment, we often have only a very small number of spike train observations, and still want to properly define an instantaneous firing rate. What could we do?

**Idea:** 'Smear' or 'smooth' spikes over time by convolving with a localized function – this is equivalent to assuming that the underlying rate from which the spike was sampled does not change too rapidly with time.

**Mathematically:**

$$r_{approx}(t) = \int_{-\infty}^{+\infty} w(\tau) \rho(t - \tau) d\tau \quad (16)$$

$w$ : smoothing function (a 'kernel')

$\rho$ : a sequence of spikes

Conditions on a well-defined kernel for smoothing spike trains:

$$w(\tau \rightarrow \pm\infty) = 0 \quad (17)$$

$$\int_{-\infty}^{+\infty} w(\tau) d\tau = 1 \quad \text{and } w > 0 \quad (18)$$

The width of the kernel depends on our assumption of how fast the underlying rate is changing.

**Side remark:** Kernels will reappear over and over: LNP models, receptive fields, spike-triggered averages, Wavelet transforms – hence, keep the intuitive/graphical explanation in mind:

**How it works:**

(a) Consider only one spike at  $t_0$ :  $\rho(t) = \delta(t - t_0)$ :

$$\Rightarrow r_{approx}(t) = w(t - t_0).$$

(b) Consider  $n$  spikes at  $t_i$ : intuitively, we “add up” (superimpose) shifted kernels  $w$ :

$$\Rightarrow r_{approx}(t) = \sum_{i=1}^n w(t - t_i).$$

## 1.5 Back to Discretized Spike Trains

...from a continuous description, we can always come back to the discrete world:

A spike sequence was described as  $\rho(t) = \sum_{i=1}^n \delta(t - t_i)$ . We can now define a discrete spike

train with sampling interval  $\Delta t$  as

$$s_j := \int_{t_0+j\Delta t}^{t_0+(j+1)\Delta t} \rho(t)dt \quad (19)$$

Discrete spike trains are useful in data analysis and modelling. Experimental equipment samples spikes in discrete time, and simulations of neural system often evolve in time steps of finite size. For example, for creating a spike train in Matlab we can write:

```
n_trials = 42, n_times = 100, dt = 0.001; p = 50*dt;  
s = rand(n_times, n_trials) < p;
```

For computing different “flavors” of firing rates one would compute

- (a)  $R$  (spike-count rate):  $\text{sum}(s(:, 17))/(n\_times*dt)$
- (b)  $r(t)$  (firing rate):  $\text{sum}(s, 2)/n\_trials/dt$
- (c)  $\langle r \rangle$  (average firing rate):  $\text{sum}(\text{sum}(s, 1), 2)/n\_trials/(n\_times*dt)$

**Motivation:** if  $p$  is low, then these spike trains look very similar to cortical spike trains if the neuron fires with a constant rate! This random process is termed a “Bernoulli process”. Soon, we will study these particular statistics of spike trains.

## 2 STATISTICS OF SPIKE TRAINS

### Motivation:

Previously we computed rates to obtain information about the average number of spikes at a certain time or during a prolonged interval. But...

- (a) ...spike observation is stochastic since there are many variables and factors we can not observe or control.
- (b) ...recordings of spikes are subject to different sources of noise.
- (c) ...spike transmission itself is typically unreliable and subject to noise.

**...to analyze data:** you have to cope with noise and to apply appropriate statistical tests.

**...to understand encoding:** you have to know statistics to draw meaningful conclusions.

**...to understand neural processing:** you have to understand variability of neural responses and signal transmission.

**...to decode from brain signals:** you do better if you know statistics.

### 2.1 Variability and Distributions

In the following, we will introduce three important stochastic processes for working with spikes (and also other neural variables!) and discuss their mutual relations:

#### 2.1.1 Bernoulli-Process

Named after: Jakob I. Bernoulli, Switzerland, 1655-1705:

- (a) Sample space:  $O_k$  in  $\{0, 1\}$
- (b) Probability to obtain symbol "1":  $p$
- (c) Repeat process  $n$  times, probability to obtain  $k$  1's

$\implies$  Binomial distribution:

$$B_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (20)$$

In the context of spike trains,  $p$  would be the probability to observe one spike in a time window of size  $\Delta t$ , and  $B_{n,p}(k)$  the probability to observe  $k$  spikes in a time window of  $T = n\Delta t$ .

The (average) spike-count rate of this process would be  $r = p/\Delta t$ .

To obtain a continuous description, we let  $\Delta t$  go to zero and simultaneously reduce  $p$  by the same amount to keep the rate  $r = p/\Delta t$  constant. In the limit  $\Delta \rightarrow 0$  we obtain the Poisson-distribution  $P_\lambda(k)$  with an expected spike count  $\lambda = rT$  over the interval  $T$ :

### 2.1.2 Poisson-Process

Named after: Siméon Denis Poisson, France, 1781-1840:

- (a) Sample space  $O_k$  in  $\{0, 1, 2, \dots\}$
- (b) Probability to obtain  $k$  events in a time interval  $T$ , whose intervals are exponentially distributed with  $\lambda$

$\Rightarrow$  Poisson-distribution:

$$P_\lambda(k) = \frac{\lambda^k}{k!} \exp(-\lambda) \quad (21)$$

The Poisson process has some interesting properties:

- Its variance  $\sigma^2 = \lambda$  is identical to the mean  $\mu = \lambda$ , i.e. the expected spike count in a time window of size  $T$ !
- The sum of multiple Poisson processes is again a Poisson process with a mean which is the sum of the means of the single processes!

If the average number of expected events  $\lambda$  is large,  $\lambda \gg 1$ , with  $\mu = \lambda$  and  $\sigma^2 = \lambda$  the Poisson distribution approximates a Gaussian distribution  $\rho_{\mu,\sigma}(x)$ :

### 2.1.3 Gaussian-/Normal Distribution

Named after: Johann Carl Friedrich Gauß, 1777-1855:

$$\rho_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (22)$$

Here, we went from Bernoulli to Poisson to Gauß - now, what is the relation between Gauß and Bernoulli (closing this loop):

- (a) We know that there must be a large number of expected events (Poisson to Gauss):  
 $\lambda = np \gg 1 \dots$

(b) We know that  $p$  must be small (negligible probability to obtain two spikes per time window  $\Delta t$ )...

$\Rightarrow$  the exact condition from Bernoulli to Gauss is that  $np(1-p) \geq 9$  with  $\sigma^2 = np(1-p)$  and  $\mu = np$ .

#### 2.1.4 Comparison of Bernoulli, Poisson and Gauss

When to use which distribution depends partly on the statistical properties of the neural system you want to study, but is partly also subject to practical considerations: Numerical simplicity and suitability for an analytical treatment:

The **Binomial** distribution is asymmetric (only symmetric if  $p$  is about 0.5):

$\Rightarrow$  ADVANTAGE: numerically simple, used to create stochastic spike trains in computer

$\Rightarrow$  CAVEAT: not a good choice if  $p$  is about 1 .

It's not possible to have more than one spike per bin. However, with  $\lambda = np/T$ , the **Poisson** distribution is better suited to describe spike counts from high rates:

$\Rightarrow$  ADVANTAGE: captures statistics if  $p$  is large in small time wins

$\Rightarrow$  CAVEAT: might be complicate to handle analytically

Thus, if  $\lambda T \gg 1$ , a **Gaussian** or **Normal** distribution is a reasonably good approximation:

$\Rightarrow$  ADVANTAGE: nicely to compute with (lots of analytical stuff...)

$\Rightarrow$  CAVEAT: only good for large rates...

#### Why are these three distributions so important?

- Spike trains in many cortical areas resemble Poisson
- Bernoulli is an easy way to generate spike trains in simulations
- Lots of mathematical properties of these distributions are known (e.g.: two independent Poisson processes add up to a Poisson process with added rates)
- Gauss is a good simplification of Poisson for medium to high rates
- Many theoretical results about population coding and information transmission in single neurons and neuron populations were obtained using these distributions

#### 2.1.5 Interspike-Intervals and Exponential Distribution

Besides spike counts, another important characteristics of spike trains is the **Interspike-Interval distribution**  $\rho(\Delta T)$ .

For example, if we have a “Poissonian” neuron that just fired a spike, what is the probability

that it will fire again after time  $T$ ?

**Result:** this probability distribution is given by the Exponential distribution

$$\rho(\Delta T) = r \exp(-r\Delta T) \quad (23)$$

(can be derived from Bernoulli-process for infinitely small  $\Delta T$ , not shown):

- sample space:  $[0, \infty]$
- can be used to numerically generate series of events at times  $t_1, t_2, \dots$  that are Poisson-distributed
- the mean value is :  $\mu = \int t\rho(t)dt = 1/r$

## 2.2 Descriptive Statistics

Here we introduce some concepts for characterizing the statistics of spike trains. In contrast to the previous paragraphs, we do not work with full distributions here, but instead characterize spike trains in terms of mean and variance (standard deviation).

In particular, we want to know:

**How regular or how variable is spike activity...:**

...over repetitions of trials?

...over recording time?

Assume we have a set of rates  $R_m$  (e.g.,  $m$  indicates the index of one experimental trial).

**Mean and variance for a set of samples:**

$$E[R] = \mu := \frac{1}{M} \sum_m R_m \quad (24)$$

$$\text{Var}[R] = \sigma^2 := \frac{1}{M-1} \sum_m (R_m - \mu)^2 \quad (25)$$

**Computing the variance can be expressed as:**

$$\text{Var}[R] = E[(R - E[R])^2] = E[R^2] - (E[R])^2 \quad (26)$$

**Coefficient of variation, COV:**

$CV = \sigma/\mu$  (how large are deviations from mean w.r.t. mean?)

**Signal-to-Noise ratio, SNR:**

$SNR = \mu/\sigma$  (inverse of CV, the larger SNR, the smaller is the noise, thus: the signal is better)

**Fano-Factor:**

$FF = \sigma^2/\mu$  (is one for a Poisson process, might be used as an indicator)

**(Auto,Cross-)covariance:**

for example, for time-varying signals  $X_t$  and  $Y_t$ :

$$COV_{XY}(\tau) = E[(X_t - E[X_t]) \cdot (Y_{t-\tau} - E[Y_{t-\tau}])] \quad (27)$$

$$= E[X_t Y_{t-\tau}] - E[X_t]E[Y_{t-\tau}] \quad (28)$$

...if  $X = Y$ ,  $COV_{XX}$  is the Auto-Covariance!

**(Auto,Cross-)correlation:**

$$COR_{XY}(\tau) = \frac{COV_{XY}(\tau)}{\sqrt{Var[X_t]Var[Y_{t-\tau}]}} \quad (29)$$

...this is just a scaled version of the covariance normalized to  $[-1, +1]$ !

# 3 DECODING

## Decoding – Motivation:

- ...learn which kind of info the neuron is processing!
- ...learn in which “feature” of a response the info about a stimulus is “hidden”: Examples:
  - o rate
  - o spike timing, or temporal patterns
  - o latency...
- use signal for a technical application (e.g. BCI)

## 3.1 Overview

Decoding is a vast field, systematic approach needed. Some aspects one has to consider:

(a) Number and type of stimulus variables:

- ...from 1 to  $N$  variables (luminance vs. pixel values)
- ...discrete vs. continuous (animal/non-animal vs. direction)

(b) Extraction of features: ...rates, phase synchrony, spectral power, latency (not topic of this lecture, except rates...)

(c) Number and type of response variables:

- ...from 1 to  $N$  variables (one neuron, fMRI voxels)
- ...discrete vs. continuous (spikes vs. BOLD levels)

(d) Estimators/classifiers/reconstruction algorithms

Decoding can be used in many different contexts, for example we can decode from a neural response which stimulus has been presented, we can decode an intended action from brain activity, we can predict the activation of brain area  $B$  from the activation of brain area  $A$ .

Before coming to more elaborate decoding schemes, we will start with the most simple problem: two discrete stimulus states (condition  $+$  vs.  $-$ , and one response variable (discrete or continuous). Most complex problems in which we have multivariate data and continuous 'states' can be broken down to this much simpler situation, and hence allow for a preliminary investigation for getting a first 'feeling' of how much information is in the data....

## 3.2 Receiver-Operator Characteristic (ROC)

Consider two stimuli or stimulus classes, named  $-$  and  $+$ , and a scalar response  $r$ . Since the brain and measurements are noisy,  $r$  will not be identical for every time stimulus  $-$  or  $+$  is presented:

$$p(r|-) \quad , \quad p(r|+) \quad (30)$$

**Problem:** From an arbitrary measurement  $r$ , how can I guess (i.e. estimate) which stimulus has been shown (this situation is termed a 'two-alternative-forced-choice experiment with one sample').

**Idea:** In many cases, the average values  $\langle r \rangle_-$  and  $\langle r \rangle_+$  different, say  $\langle r \rangle_+ > \langle r \rangle_-$ . Thus the simplest choice for an estimator is to select a threshold  $z$  somewhere in between those values, and compare it to  $r$ :

- if  $r > z \implies$  it was  $+$ !
- if  $r < z \implies$  it was  $-$ !

### 3.2.1 Errors and correct decisions

Let us compare the errors and correct decisions we are making using a particular  $z$ . We have to use of "cumulative probability distributions":

$$\beta(z) = \int_z^\infty p(r|+)dr \quad \text{true Positives} \quad (31)$$

$$\alpha(z) = \int_z^\infty p(r|-)dr \quad \text{false Positives} \quad (32)$$

With  $\alpha$  and  $\beta$ , we can define the ROC-function  $g$  as follows (implicit parameter  $z$ ):

$$g \text{ maps } \alpha(z) \text{ to } \beta(z) : \beta = g(\alpha) \quad (33)$$

### 3.2.2 Maximizing Performance

Up to now, a suitable value for the threshold  $z$  was not specified. How can we maximize the number of correct decisions (i.e. minimize the number of errors) by choosing  $z$  'wisely'?

$\implies$  add the true Positives and true Negatives. The true Negatives are  $1-(\text{false Positives})$ . We also assume that the classes  $+$  and  $-$  occur with the same frequency (i.e., they have an identical

'prior'):

$$C_1(z) = \frac{1}{2} \left( \beta(z) + \int_{-\infty}^z p(r|-)dr \right) \quad (34)$$

$$= \frac{1}{2}(\beta(z) + 1 - \alpha(z)) \quad (35)$$

Maximize  $\implies$  take derivative w.r.t.  $z$ , and set to 0:

$$\frac{dC_1}{dz} \stackrel{!}{=} 0 \quad (36)$$

$$\frac{d\beta}{dz} \stackrel{!}{=} \frac{d\alpha}{dz} \quad (37)$$

$$p(z|-) \stackrel{!}{=} p(z|+) \quad (38)$$

Thus  $C_1$  can have multiple minima/maxima!

If we expand  $d\beta/dz$  we obtain:

$$\frac{d\beta}{dz} = \frac{d\beta}{d\alpha} \frac{d\alpha}{dz} \quad (39)$$

Thus, for satisfying equation (37) we obtain:

$$\frac{d\beta}{d\alpha} \stackrel{!}{=} 1 \quad (40)$$

$$\longrightarrow \frac{dg}{d\alpha} \stackrel{!}{=} 1 \quad (41)$$

Hence minima/maxima in performance are at points where the slope of the ROC-curve is 1.

$\implies$  This gives us the maximum performance in a 2-AFC task with one sample!

### 3.2.3 Two samples

What is the performance in a two-alternative, forced-choice paradigm with **two** samples?

For this problem, we need a different estimator. Let's assume we have made two experiments with results  $r_1$  and  $r_2$ . Estimate:

- if  $r_2 > r_1$ :  $r_1$  was from  $-$ , and  $r_2$  from  $+$
- if  $r_1 \geq r_2$ :  $r_1$  was from  $+$ , and  $r_2$  from  $-$

For obtaining the maximum performance  $C_2$ , we have to integrate over all possible pairs of  $r_1$  and  $r_2$  (not shown in this lecture).

The final result reads:

$$C_2 = \int_0^1 \beta(\alpha) d\alpha = \int_0^1 g(\alpha) d\alpha \quad (42)$$

$\implies$  Performance in 2-AFC task with two samples  $r_1, r_2$  is given by integral over ROC-curve!

### 3.2.4 Special case: Gaussians

A generic case are distributions  $p$  which are Gaussians with same variances. With introducing the “Complementary error function”:  $\text{erfc}(z) := 2/\sqrt{\pi} \int_z^\infty \exp(-r^2) dr$  we obtain:

(a) 2-AFC with **one** sample:

$$C_1^{\max} = \frac{1}{2} \text{erfc} \left[ -\frac{\langle r \rangle_+ - \langle r \rangle_-}{2\sqrt{2}\sigma} \right] \quad (43)$$

(b) 2-AFC with **two** samples:

$$C_2 = \frac{1}{2} \text{erfc} \left[ -\frac{\langle r \rangle_+ - \langle r \rangle_-}{2\sigma} \right] \quad (44)$$

One can also rewrite these expressions as functions of  $d' := (\langle r \rangle_+ - \langle r \rangle_-)/\sigma$ .  $d'$  is the “discriminability” or “significance” (expressed in terms of  $\sigma$ ).

## 3.3 Bayesian Estimation

### 3.3.1 The Loss Function

**Motivation:**

(a) What happens if the probability (prior) for a specific stimulus is different from 1/2?

(b) Consider discrimination: Can we do better than just take a threshold?

Answer for a): Consider **Bayes' equation!**

Let's remember how to compute with conditional probabilities...

$$p(r, s) = p(r|s)p(s) \quad (45)$$

$$= p(s|r)p(r) \quad (46)$$

We can think of  $r$  as a neural response observed from an (unknown) stimulus condition  $s$ . So what do we want to know?

- $p(s|r) \implies$  the “posterior”, giving us the probability that stimulus  $s$  underlies the observation  $r$ .

What do we have?

- $p(s) \implies$  the “prior”
- $p(r|s) \implies$  the “likelihood”

Thus we solve w.r.t.  $p(s|r)$ :

$$p(s|r) = \frac{p(r|s)p(s)}{p(r)} \quad (47)$$

$\implies$  This is Bayes' rule (Thomas Bayes, 1701-1761).

**Side question:** What do we do about  $p(r)$ ?

- ...we can compute it:  $p(r) = \sum_{s'} p(r|s')p(s')$
- ...it is not important if  $r$  (the 'observation') is assumed to be fixed, and if we want to maximize with respect to the unknown  $s$ :  $p(s|r) \propto p(r|s)p(s)$ .

So the 'obvious' solution to questions (a) and (b) would be to compute  $p(s|r) \approx p(r|s)p(s)$  for the observed  $r$ , and choose the  $s$  which maximizes this posterior as our estimate. This choice would minimize the number of incorrect predictions.

However, a more general solution needs to consider that different errors might be associated with different penalties: think of a BCI wheelchair user standing on top of a cliff, where an executed movement into the wrong direction could have dramatic consequences. So for a better answer for (b), we introduce a “Loss” or penalty  $L$  associated with incorrect decisions:

For two alternative choices  $+$  and  $-$ , the loss is defined as:

- $L_+$ : we estimated  $+$ , but  $-$  would have been correct
- $L_-$ : we estimated  $-$ , but  $+$  would have been correct

Now let us assume we have observed  $r$  and performed an estimation. The average losses associated with the respective choices  $+$  or  $-$  are:

$$\langle L \rangle_+(r) = L_+ p(-|r) \quad (48)$$

$$\langle L \rangle_-(r) = L_- p(+|r) \quad (49)$$

Our (new) strategy to make an estimation will now be:

- If  $\langle L \rangle_+(r) \leq \langle L \rangle_-(r)$ , we minimize the loss by choosing  $+$ ...
- ...or, put differently, if  $\langle L \rangle_-(r)/\langle L \rangle_+(r) \geq 1$ , we choose  $+$ .

We can rewrite this criterion by using the expressions

$$p(-|r) = p(r|-)p(-)/p(r) \quad (50)$$

$$p(+|r) = p(r|+)p(+)/p(r) \quad (51)$$

from Bayes' equation and inserting them into equation (48) and equation (49):

$$\frac{\langle L \rangle_-(r)}{\langle L \rangle_+(r)} = \frac{L_- p(r|+) p(+)}{L_+ p(r|-) p(-)} \stackrel{!}{\geq} 1 \quad (52)$$

We now introduce the **likelihood ratio**  $l(r) = p(r|+)/p(r|-)$ , and the criterion for making the estimation  $+$  becomes

$$l(r) := \frac{p(r|+)}{p(r|-)} \stackrel{!}{\geq} \frac{L_+ p(-)}{L_- p(+)} = \text{const.} \quad , \quad (53)$$

where the RHS is a constant containing the involved losses and priors.

**Special case:** In case of equal losses  $L_+ = L_-$ , and equal priors  $p(-) = p(+)$  = 1/2:

$$l(r) > 1 \quad \longrightarrow \text{answer } +! \quad (54)$$

$$l(r) \leq 1 \quad \longrightarrow \text{answer } -! \quad (55)$$

The relation to the ROC curve is:  $d\beta/d\alpha = p(z|+)/p(z|-) = l(z)$ .

- (a) Gaussians with equal variance: deciding according to  $l(r)$  is identical to having one threshold  $z$ .
- (b) Other functions: depends on shape: e.g.:
  - Gaussians with different variances (and same mean!)
  - Bimodal, skewed distributions
- (c) There is a simple graphical interpretation: at one measured  $r$ , choose symbol whose probability distribution (weighted by loss, of course) has a higher value.

### 3.3.2 Bayesian inference

Bayesian inference is a general concept, and possible to apply to:

- multiple input variables
- multiple output variables
- continuous input variables
- ...

Write neural responses as:  $R = \{r_1, r_2, \dots, r_N\}$

Write input/stimulus as:  $s = \{s_1, s_2, \dots, s_M\}$

Bayes' formula is completely analog to the 1D-case:

$$p(s|R) = \frac{p(R|s)p(s)}{p(R)} \quad (56)$$

**Remark:** Why is taking multiple response variables into account useful?

⇒ ...can be used to disambiguate estimations

⇒ ...can be used to reduce noise and therefore improve estimation

#### What does estimation mean?

$R$  is fixed,  $p(s|R)$  is a function on  $s$ . Use  $p(s|R)$  to determine an estimate  $\hat{s}$ . Note that Bayes' equation does not relieve us from defining an appropriate **estimator**. For the case of two classes  $s \in \{+, -\}$  discussed above our choice is obvious: Compare the two losses, and choose the class which minimizes the expected loss. However, when we have more options we have to extend our concept of a loss function:

#### Which is the optimal way to obtain $\hat{s}$ ?

Not clear yet, see ROC example. For instance, if  $s$  is an orientation of a bar – is obtaining 179 as estimate when 180 was the stimulus really an error?

So, what about taking max, mean, or median instead?

In order to systematically address this question, we again need a Loss Function! Then we minimize the expected loss by constructing an appropriate estimator.

### 3.3.3 Loss functions: Continuous variables

Examples for Loss Functions  $L$  on continuous variables:

(a) **MSE** (mean squared error), or  $L_2$ -norm:

$$L(s, \hat{s}) = (s - \hat{s})^2 \quad (57)$$

(b) **MAE** (mean absolute error), or  $L_1$ -norm:

$$L(s, \hat{s}) = |s - \hat{s}| \quad (58)$$

These examples imply well-defined estimators on  $p(s|R)$ :

(a) for the **MSE**: the mean, i.e.

$$\hat{s} := \int s p(s|R) ds \quad (59)$$

the **MMSE** estimator (minimum mean-squared error estimator).

(b) for the absolute error: the median, i.e.

$$\hat{s} := \inf_z \left\{ \int_z^\infty p(s|R) ds \geq \frac{1}{2} \right\} \quad (60)$$

the **MMAE** estimator (minimum mean absolute error estimator).

To arrive at those equations, one has to consider the average loss  $L_{avg}$  induced by choosing an estimator  $\hat{s}$ ,  $L_{avg} = \int p(s|R) L(s, \hat{s}) ds$  and minimize this quantity with respect to  $\hat{s}$ .

### 3.3.4 Loss functions: Discrete variables

For discrete variables (typically with few states, or for variables which can not be ordered like red, green, blue), the **MAP** (maximum-a-posteriori) estimator is used:

$$\hat{s} := \operatorname{argmax}_s p(s|R) \quad (61)$$

If  $s$  is a binary variable with states  $-$  and  $+$ , this estimator is identical to using the likelihood ratio  $l(R)$  with equal losses for the two choices. If the prior  $p(s)$  is uniform over  $s$  (“no prior”), this is identical to **ML** (maximum-likelihood) estimation:

$$\hat{s} := \operatorname{argmax}_s p(s|R) = \operatorname{argmax}_s \frac{p(s|R) \text{const.}}{p(R)} = \operatorname{argmax}_s p(R|s) \quad (62)$$

MAP (or ML) maximize the **Classification Performance**, which implicitly assumes equal loss for all incorrect decisions.

### 3.3.5 Caveats of Bayesian estimation

Experimentally, the likelihood  $p(R|s)$  is often hard or impossible to obtain!

#### Examples:

(a) Curse of dimensions:

- Assume  $r_i$  is number of spikes, and from 0 to 9 spikes could be measured in one trial.
- With 10 neurons,  $p(R|s)$  is a table with  $10^{10}$  entries.
- Would need some  $10^{12}$  trials to populate  $p(R|s)$ .

(b) Limited statistics:

- If we take fewer trials, there will be problems with correct estimation:
- Assume we took 100 measurements for  $+$  and  $-$  each:

$$\langle n \rangle(r = 2|s = +) = 0.01, \quad \text{with underlying } p = 0.02 \quad (63)$$

$$\langle n \rangle(r = 2|s = -) = 0.02, \quad \text{with underlying } p = 0.01 \quad (64)$$

- Now  $r = 2$  measured:  $p(-|r) \gg p(+|r)$ , but in reality it's the other way round!

#### Remedies:

(a) Assume that  $p(R|s)$  takes a certain shape (regularization), is constrained:

- Measure  $p(R|s)$  only for certain stimulus values, or take only few samples  $p(R|s)$  for the full stimulus space, and...
- ...extrapolate/interpolate!

(b) Assume independency of neural signals  $r$ :

- Measure only  $p(r_1|s)$ ,  $p(r_2|s)$ , ... and use the product rule for independent random processes:  $p(R|s) = \prod_i p(r_i|s)$

### 3.3.6 Extensions

Some properties of an estimator  $X=\{\text{ML, MAP, ...}\}$ :

(a) It might have a Bias:

$$b_X(s) = \langle \hat{s}_X \rangle - s \quad (65)$$

(b) Variance of an estimator:

$$\sigma_X^2(s) = \langle (\hat{s}_X - \langle \hat{s}_X \rangle)^2 \rangle \quad (66)$$

(c) Trial-averaged squared estimation error:

$$\langle (\hat{s}_X - s)^2 \rangle = \sigma_X^2(s) + b_X^2(s) \quad (67)$$

**Online estimation:** Possible with an iterative Bayesian scheme:

$$p(s_t, s_{t-1}, s_{t-2}, \dots | r) = \frac{p(r|s_t)p(s_{t-1}, s_{t-2}, \dots | r)}{p(r)} \quad (68)$$

This is especially helpful if an initial estimate is to be refined over time.

Other “famous” estimators:

(a) Estimation on circular manifolds:

- Problem: Gauss at  $0/2\pi$ -boundary
- Solution: vector average, complex average

(b) Population vector

- Take care: It’s bad in most cases deviating from the Cricket example...
- Special case of a linear estimator.

(c) OLE (optimal linear estimator)

- $\hat{s} = AR + B$
- $A$  and  $B$  are specifically adapted to the problem, for yielding a minimal estimation error...

(d) Other estimators

- In general: they require test/training sets(!)
- There’s a tradeoff: training performance/generalization
- Perceptron
- Support vector machine

## 4 FREQUENCY ANALYSIS

**Motivation:** Brain rhythms have been linked to various cognitive functions – but how do we analyze brain data to find out if particular rhythms/oscillations are present/absent?

Use Fourier or Wavelet analysis:

- (a) Fourier more fundamental, assumes stationary composition.
- (b) Wavelet analysis more flexible for time-varying signal content, but not so readily invertible.

### 4.1 Fourier Analysis

#### 4.1.1 Continuous sine/cosine Fourier transform

**Idea:** Signal composed of (co)sine waves of different frequencies, amplitudes and phase shifts.

**Mathematics tells us:** Every signal  $s(t)$  can be described as a superposition of cosine waves!

How to find out about the particular composition of a signal?

**Idea:** Compare signal with cosine wave of circular frequency  $\omega = 2\pi f$  ( $f$ : frequency) and find amplitude  $A$  and phase shift  $\phi$  which “best matches” signal. Then do this for *all* possible frequencies  $f$ !

**Solution #1:**

Try for all possible phase shifts and select phase shift for which overlap (integral over product) is maximized, size of integral then proportional to amplitude!

⇒ This is impractical and unnecessary...

**Solution #2:**

Interpret  $A$  and  $\phi$  as a vector in polar coordinates...

⇒ ...has components  $A_{cos}$ ,  $A_{sin}$

⇒ ...if we have a means to compute these components, we can unambiguously obtain  $A$  and  $\phi$

In particular, we can obtain these components by computing (you can check it by computing integrals!):

- $A_{cos}$ : overlap (integral) with **unshifted cosine**:

$$A_{cos}(\omega) \propto \int s(t) \cos(\omega t) dt \quad (69)$$

- $A_{sin}$ : overlap (integral) with **unshifted sine**:

$$A_{sin}(\omega) \propto \int s(t) \sin(\omega t) dt \quad (70)$$

Thus, signal can either be decomposed into sines and cosines of different amplitudes  $A_{cos}$ ,  $A_{sin}$ , or into cosines (or sines) of different amplitudes and phases  $A$ ,  $\phi$ .

#### 4.1.2 Continuous Fourier transform, complex representation

Make everything simpler by introducing a new numerical concept. Instead of always having to handle two numbers (i.e. amplitude and phase, or sin- and cos-components), we use *one* complex number instead:

**Idea:** Use complex numbers  $z$ . They have two components:

- the **real** part  $a$  and...
- the **imaginary** part  $b$ ,

and are written as  $z = a + ib$ .  $i$  is the complex unit and defined as  $i := \sqrt{-1}$ . Complex numbers can be understood as vectors in a 2D-coordinate system with a real axis and an imaginary axis.

- $i = \sqrt{-1}$ , or  $i^2 = -1$ .
- $z = a + ib \implies$  ...it's like going a distance  $a$  horizontally and  $b$  vertically
- $abs(z) = |z| = \sqrt{a^2 + b^2} \implies$  ...it's computing the length of the vector (Pythagoras!)
- $\exp(i\phi) = \cos(\phi) + i \sin(\phi) \implies$  ...Eulers identity
- $z = |z| \exp(i\phi) \implies$  ...Polar/Exponential notation
- $z^* = a - ib \implies$  ...Complex conjugate
- $\text{Re}(z) = a = 0.5(z + z^*) \implies$  ...get the real part
- $\text{Im}(z) = b = 0.5(z - z^*)/i \implies$  ...get the imaginary part

**Practical note:** if you have a complex number  $z$ , you can find out its amplitude by computing  $|z|$  (`abs(z)` in Matlab), and its phase  $\phi$  by computing  $\arg(z)$  (`angle(z)` in Matlab).

Computing with complex numbers is straightforward:

- Addition/Subtraction:  $z_1 \pm z_2 = (a_1 \pm a_2) + i(b_1 \pm b_2)$
- Multiplication:  $z_1 z_2 = (a_1 b_1 - a_2 b_2) + i(a_1 b_2 - a_2 b_1)$

Division  $z_1/z_2$  is a bit more complicated, and more easily written in polar notation:

- Division:  $z_1/z_2 = |z_1|/|z_2| \exp(i(\phi_1 - \phi_2))$
- Multiplication in polar notation:  $z_1 z_2 = |z_1||z_2| \exp(i(\phi_1 + \phi_2))$

### For Aficionados:

Eulers identity can be proven by looking at series expansion of  $\cos(x)$ ,  $\sin(x)$  and  $\exp(x)$ :

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} \quad (71)$$

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \quad (72)$$

$$\exp(ix) = \sum_{n=0}^{\infty} \frac{(ix)^n}{n!} \quad (73)$$

Using complex numbers, the Fourier transform simply writes as

$$S(\omega) = c \int s(t) \exp(-i\omega t) dt \quad (74)$$

comparing the signal  $s(t)$  simultaneously with a sine and cosine template, and delivering an amplitude and phase encoded as the angle and length of a complex number.

Mathematically, Fourier can transform a signal from the time (or space) domain into the frequency domain, and **vice versa**!

The **inverse Fourier transform** (for the vice versa...) is defined as:

$$s(t) = C \int S(\omega) \exp(it\omega) d\omega \quad (75)$$

In short hand notation, the (inverse) Fourier transform (FT) can be written as  $F[\dots]$  and  $F^{-1}[\dots]$ :

$$S = F[s] \quad (76)$$

$$s = F^{-1}[S] \quad (77)$$

$$F[F^{-1}] = 1 \quad (78)$$

The constants  $c$  and  $C$  provide a normalization such that  $F[F^{-1}] = 1$ . There are three possibilities in use:

$$c = 1 \quad , \quad C = \frac{1}{2\pi} \quad (79)$$

$$c = C \quad , \quad C = \frac{1}{\sqrt{2\pi}} \quad (80)$$

$$c = \frac{1}{2\pi} \quad , \quad C = 1 \quad (81)$$

**Example:** Transform of a cosine:

$$s(t) = \cos(\omega_0 t) \quad (82)$$

$$= \frac{1}{2} (\exp(i\omega_0 t) + \exp(-i\omega_0 t)) \quad \dots \text{using the identities from above} \quad (83)$$

$$\rightarrow S(\omega) = \frac{c}{2} \left( \int \exp(-i(\omega - \omega_0)t) dt + \int \exp(-i(\omega + \omega_0)t) dt \right) \quad (84)$$

But how to solve this integral? A trick helps: consider that the inverse FT of the solution  $S(\omega)$  must give us back the initial signal  $s(t)$ :

$$s(t) = C \int S(\omega) \exp(it\omega) d\omega \quad (85)$$

$$= \frac{1}{2} (\exp(i\omega_0 t) + \exp(-i\omega_0 t)) \quad (86)$$

So, in other words, which  $f(\omega)$  would give us:  $\int f(\omega)g(\omega) = g(\omega_0)$ ? Simple – the delta-function again!  $f(\omega) = \delta(\omega - \omega_0)$ ! Thus we obtain:

$$S(\omega) = \frac{1}{2C} (\delta(\omega - \omega_0) + \delta(\omega + \omega_0)) \quad (87)$$

#### 4.1.3 Discrete Signals

The continuous FT is something which is useful for analytical calculations. In contrast, in Neuroscience we have **discrete signals** over a **finite period**  $T$ : this problem is handled by the discrete Fourier transform assuming a periodic signal of finite length  $T$ .

In practice, we will have  $N$  time bins spanning the interval  $T$ . The FT then analyzes the signal for exactly  $N$  frequencies ranging from  $k = 0$  to  $N - 1$  periods/interval with corresponding frequencies  $f = k/T$ .

If a vector  $s_t$  with  $N$  samples is Fourier-transformed, one obtains a vector  $S_k$  with  $N$  amplitudes/phases for the different frequencies.

Consider the two signals:

- 2 Hz sampled at 25 Hz
- 23 Hz sampled at 25 Hz

The first signal looks good when sampled, while the second signal looks like a signal with a **lower** frequency. In fact, it looks like a signal with the “negative” frequency  $23 \text{ Hz} - 25 \text{ Hz} = -2 \text{ Hz}$ .

### Nyquist frequency:

The upper half of a vector from a Fourier transform is equivalent to the signal compared with a cosine having a “negative” frequency. This is the case from the so-called **Nyquist frequency** from  $k = N/2$  on.

⇒ Loosely speaking: For one specific frequency, the Fourier transform represents the corresponding wave with **half** of its amplitude at positive frequencies, and **half** of its amplitude at negative frequencies. Thus, for recovering the amplitude for a specific frequency (wave number  $k$ ), one has to add amplitudes of  $S_k$  and  $S_{N-k}$  (except for  $k = 0$ !).

**Power spectrum:** The power spectrum  $P$  is defined as the square of the absolute of the Fourier transform, i.e.  $P(\omega) = |S(\omega)|^2$ . Please remember that the Fourier transform delivers typically a *two-sided* spectrum, but that the amplitude spectrum  $A(\omega) = |S(\omega)|$  or power spectrum are normally plotted as *one-sided* spectra. If you ‘convert’ a two sided result (from an analytical calculation or by using the fast-fourier transform in e.g. Matlab) to a one-sided display, you have to compute, for  $\omega > 0$ ,

$$A(\omega) = 2|S(\omega)| \quad (88)$$

$$P(\omega) = 2|S(\omega)|^2, \quad (89)$$

while for  $\omega = 0$  you will have  $A(0) = |S(0)|$  and  $P(0) = |S(0)|^2$ . This remark holds in equivalent form for the discrete Fourier transform.

## 4.2 The Convolution Theorem

The Fourier transform is not only useful to determine the frequency content of temporal signals, but also to perform so called **convolutions**. A convolution of two functions  $s(t)$  and  $g(t)$  is defined by the following equation:

$$h(t) = \int_{-\infty}^{+\infty} s(t')g(t-t') dt' \quad (90)$$

Here,  $g$  is often referred to as **convolution kernel**. The procedure of a convolution can be visualized as follows: We interpret  $s$  as a arbitrary function, that is compared to a template function  $g$ . To account for possible shifts on the time axis, the template is shifted between  $t = +\infty$  and  $t = -\infty$  and the comparison is made at every position  $t$ . If both functions match perfectly, the result  $h(t)$  is big. If the shifted  $g$  does not match with  $s$ , the result is small.

Applications of this equation are numerous in physics, and reach from simple filter operations to timely resolved frequency analysis (examples will be discussed later). First, we want to

understand the connection between convolutions and the Fourier transformation. As a short notation of the application of the Fourier transform  $F$  to a function  $s$  (resp. its reverse transformation  $F^{-1}$ ) we already introduced:

$$S(\omega) = F[s(t)](k) \quad (91)$$

$$s(t) = F^{-1}[S(\omega)](t) \quad (92)$$

Now we apply the Fourier transform to both the left and the right side of the Definition (90) and gain after short computation,

$$H(\omega) = 2\pi S(\omega)G(\omega), \quad (93)$$

or, in short notation,  $H = 2\pi F[f]F[g]$ . To get the sought result  $h(t)$ , we apply the inverse Fourier transform to the equation, which results in

$$h = F^{-1}[2\pi F[f] \cdot F[g]]. \quad (94)$$

The advantage of equation (94) over (90) lies in the computation speed of the numerical implementation of the Fourier transform in the Fast Fourier Transform algorithm (FFT): despite a three-fold transformation, calculating equation (94) is faster than computing the integral in (90), since the convolution integral corresponds to an *element-wise* multiplication of the Fourier coefficients in the Fourier space  $\omega$  – try to verify!

#### 4.2.1 Non-periodic Functions

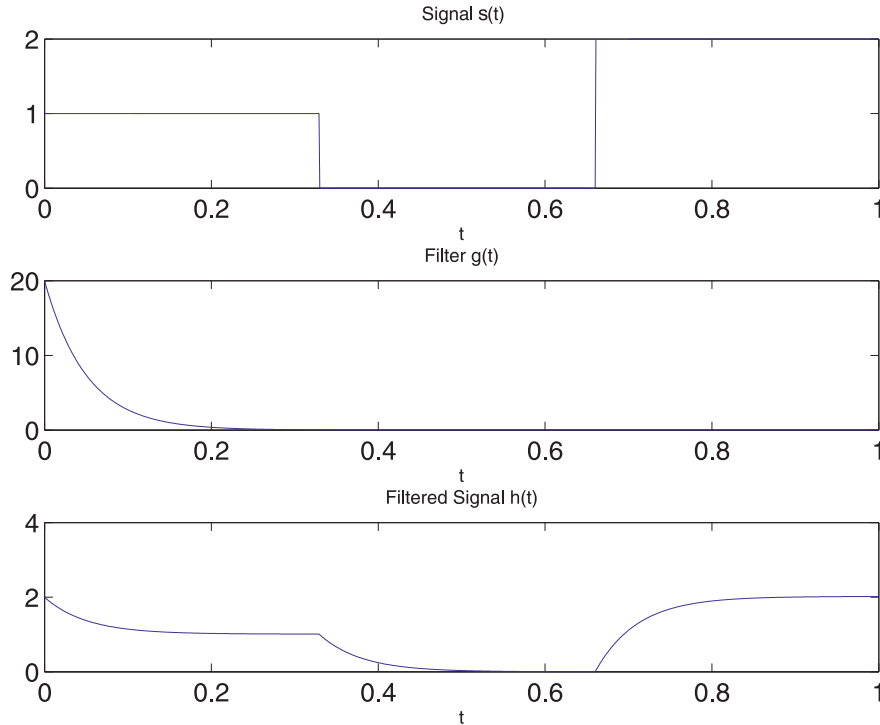
The discrete Fourier transformation can be applied to any arbitrary function  $s$ , periodicity is not required. The convolution theorem, on the other hand, does strictly speaking only apply for the Fourier integral. It thus demands either a definition of the function in  $[-\infty, +\infty]$ , or a periodic function on a finite interval. Only in the latter case, the FFT can be directly used, i.e. without 'post-treatment', to evaluate a convolution integral.

Still, the FFT can be used with non-periodic functions, provided the convolution kernel  $g$  is sufficiently narrow, i.e. its absolute value almost vanishes within  $m \ll n$  sampling points around  $t = 0$ . This is for example given for exponentially decaying functions or Gaussian distributions. Unwanted boundary effects can be suppressed by simply discarding the first and last  $m$  elements of the transform. Another possibility is the continuation of a function above its borders by a suitable extrapolation.

## 4.2.2 Convolutions in Computational Neuroscience

### (a) Filtering of a signal $s(t)$ :

Let us consider the filtering of a temporal signal as an example for the application of the convolution theorem. The signal is chosen to be a rectangle function, that is 1 in the first third of the data acquisition period, then jumps to 0 and increases to 2 in the last third. The filter shall be an exponentially decaying function  $\exp(-t/\tau)$ , which realizes a low pass with time constant  $\tau$ .

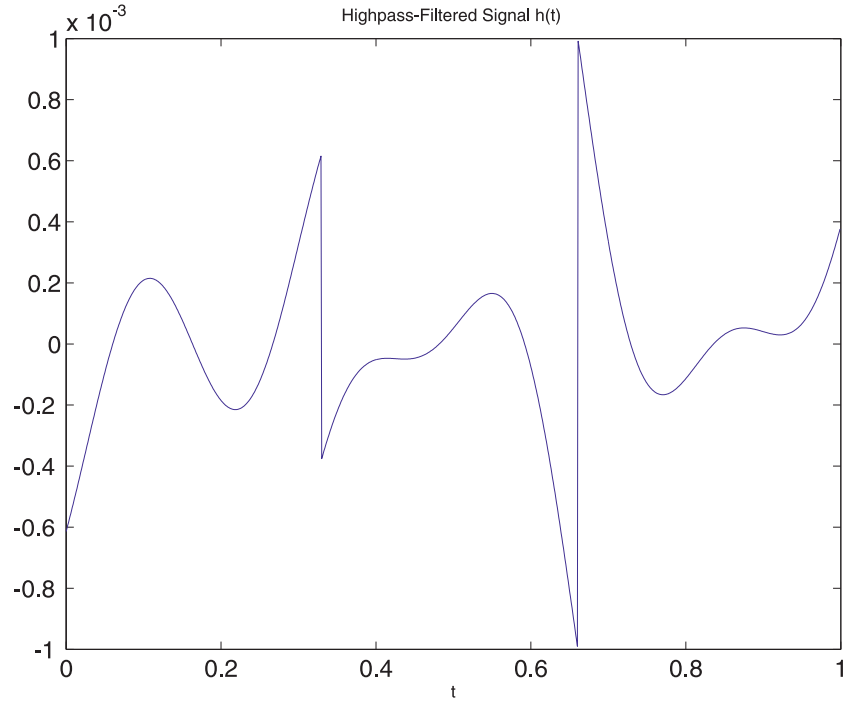


Equation 94 reveals a further, interesting possibility to apply filter operations: The term  $F[g]$ , the transform of the filter kernel, acts through the multiplication with  $F[f]$  similar to a 'switch', that only allows passage of 'permitted' frequencies and suppresses 'unwanted' frequencies. This properties can be used to directly design filters with the sought properties in the frequency space. As an example, we could set all frequency components of a signal to 0 below a threshold frequency. This transformation realizes a high pass on  $f$ :

When defining own filters in frequency space, one has to be cautious: most of these filter are acausal and change the phase of  $f$  also in the not suppressed frequency bands.

### (b) Correlation Functions:

The cross-correlation  $C(\tau)$  between signals  $f(t)$  and  $g(t)$  is defined by the following



equation:

$$C(\tau) = \int f(t)g(t+\tau)dt \quad (95)$$

Here,  $\tau$  denotes the delay between the 'leading' signal  $f$  and the for  $\tau > 0$  'lagging' signal  $g$ . For  $f(t) = g(t)$ ,  $C$  is referred to as the auto-correlation.

equation (95) is not directly a convolution, this means one has to be careful when applying the convolution theorem. Utilizing that  $F[s(-t)](k) = S(-k)$  it holds:

$$C(\tau) = F^{-1} [ 2\pi F[f(t)](k)F[g(t)](-k)](-\tau) \quad (96)$$

### (c) Reverse Correlation:

A simple model for the response properties of neurons in the visual system assumes that the response  $r(t)$  of a neuron is constructed by a linear superposition of a stimulus  $s(x, y, t)$  at the location  $(x, y)$  on the retina, multiplied with a weight function  $w(x, y, \tau)$ .  $\tau$  is the delay between stimulus and neuronal response, and  $g[ ]$  an additional point-non-linearity:

$$r(t) \propto g \left[ \int_x dx \int_y dy \int_\tau d\tau w(x, y, \tau) s(x, y, t - \tau) \right] \quad (97)$$

Again,  $w$  can be interpreted as a filter. With this insight, the inner integral over  $\tau$  is best

numerically solved via the convolution theorem.

When the stimulus  $s(x, y, t)$  and the response  $r(t)$  is known, the unknown filter  $w$  can be determined under special conditions. This becomes specifically easy, when the stimulus is uncorrelated white noise:

$$w(x, y, \tau) \propto \int r(t) s(x, y, t - \tau) dt \quad (98)$$

After application of the convolution theorem, we get:

$$w(x, y, \tau) = F^{-1} [ 2\pi F[r(t)](k) F[s(x, y, t)](-k) ](\tau) \quad (99)$$

**(d) Recurrent Networks:**

Neuronal networks usually have a one- or two-dimensional topology, where a neuron at position  $x$  is coupled to a neuron at position  $x'$  by a weight of magnitude  $w(x, x')$ . The dynamics of such a network are usually described as a differential equation for the neuronal activities  $A(x, t)$  in the following form:

$$\tau \dot{A}(x, t) = -A(x, t) + g[I(x, t)] \quad (100)$$

$I(x, t)$  denotes the incoming current of the neuron at position  $x$ ; this is a sum across the weighted activities of all other neurons:

$$I(x, t) = \int w(x, x') A(x', t) dx' \quad (101)$$

If the weight are invariant to translation via  $w(x, x') = w(x - x')$ , the solution of this integral is again a case for the well-known convolution theorem.

**(e) Wavelet Transformations:**

Fourier transformations are ill suited for signals, whose spectrum changes over time. This also includes signals which do contain periodic oscillations, but where the frequency is subject to fluctuations.

In wavelet transformations, a signals is convolved with a filter  $g$ , that only contains few oscillations of a specific period and decays to 0 outside of this range. A spectrum is obtained that not only depends on the frequency, but also from time. A much used wavelet is the Morlet-wavelet, which is a harmonic oscillation multiplied with a Gaussian curve. We will discuss this technique in the subsequent section.

## 4.3 Wavelet Analysis

**Disadvantage of Fourier analysis:** bad for detecting **changes** in signal decomposition.

- (a) **Idea:** Use (sine) waves localized in time (i.e. “wavelets”) to assess the spectral content of a signal in a time-resolved manner.

Examples:

- Haar wavelets (box functions with negative and positive part)
- Morlet wavelets (Gaussian envelope on sine wave, Gabor function!)

- (b) Applying the wavelet transform:

- compare the wavelet with the signal, note the result, shift the wavelet, etc....
- repeat for as many frequencies as you like

**Remark:** again, the wavelet is complex-valued for finding the best phase of the wave that matches the signal.

- (c) Mathematically:

$$X_a(t) = \frac{1}{a} \int_{-\infty}^{\infty} \overline{\Psi\left(\frac{t' - t}{a}\right)} x(t') dt' \quad (102)$$

Here  $a$  represents the *scale* at which the transform is applied. For example, the Morlet wavelet is defined as

$$\Psi(t) := \pi^{-\frac{1}{4}} \exp\left(-\frac{t^2}{2}\right) \exp(i\omega_0 t) \quad (103)$$

When scaled, the width  $\sigma$  of its Gaussian envelope is identical to  $a$ . Furthermore, the parameter  $a$  is related to the circular frequency  $\omega$  at which the signal  $x$  is analyzed by the relation  $\omega = \omega_0/a$ .  $\omega_0$  defines how many cycles of the wavelet are within the Gaussian envelope.

The transform is mathematically very similar to the Fourier transform – try to insert the following, infinitely extended 'wavelet' into the equation:

$$\Psi(t) = \exp(it) \quad . \quad (104)$$

From this, we obtain:

$$X_{\frac{1}{\omega}}(0) = \omega \int_{-\infty}^{\infty} x(t) \exp(-i\omega t) dt \quad . \quad (105)$$

- (d) Tradeoff:

- the more cycles a Wavelet contains  $\implies$  the better the frequency resolution  $\Delta f$
  - the less cycles a Wavelet contains  $\implies$  the better the temporal resolution  $\Delta t$
- $\implies$  Uncertainty principle:  $\Delta f \Delta t \geq \text{const.}!$

## 5 GENERALIZED LINEAR MODELS (GLMs)/LINEAR-NONLINEAR-POISSON MODELS (LNPs)

Generalized linear models (GLMs) can be used to model neural responses to arbitrary stimuli. GLMs employ a combination of linear operations (such as summation and convolution) with non-linear operations (such as gain functions and feedback loops) to map a stimulus to a response. The linear-nonlinear Poisson model is a simplified form of a GLM, whose parameters (summation kernel, gain function) can be estimated elegantly by computing a spike-triggered average or reverse correlation.

### Idea:

We consider 1-D, but non-stationary stimuli  $s(t)$ : interpret neuron's response as the waves that emerge if something falls on the water:

- (a)  $s(t)$  : things we drop into the water (stimulus)
- (b)  $r(t)$  : wave pattern that emerges (response)
- (c)  $f$  : BlackBox (the water or the brain)

### But how to capture time dependence??

Idea: Think of  $s(t)$  composed of  $s_1 = s(t_1)$ ,  $s_2 = s(t_2)$ , ...

## 5.1 Linear Superposition of Elementary (Pulse-)Responses

### The L-part:

- (a) First, we start with a response  $r_1(t)$  to a pulse stimulus  $s(t_1)$  at time  $t_1$ , e.g.  $s_1 D(t - t_1)$ .  $D(t - t_1)$  normally is 0 for  $t < t_1$  (causality!). Suppose response scales linearly with stimulus value  $s_1$ .
- (b) Second, we sum up all of these responses:

$$\sum_i s_i D(t - t_i) \quad (106)$$

This becomes an integral for a function  $s(t)$ , giving the linear response  $L(t)$  via

$$L(t) = \int_{-\infty}^{t'=t} s(t') D(t - t') dt' \quad (107)$$

With a change of variables,  $\tau = t - t'$ , and with introducing a “baseline activation” or

offset  $r_0$ , we obtain

$$r(t) = r_0 + L(t) = r_0 + \int_0^\infty s(t - \tau)D(\tau)d\tau \quad (108)$$

### Consistency check:

What happens for one drop of water, e.g.  $s(t) = s_1\delta(t - t_1)$  and no offset,  $r_0 = 0$ ?

$$r(t) = s_1D(t - t_1) = r_1(t) \quad \text{okay, is well-defined!} \quad (109)$$

## 5.2 Non-linear Transfer Function and Stochastic Output Spikes

**The N-part:** Responses are in general not linear: Rectify! Pass output  $L(t)$  of linear stage through neural transfer/gain function such that  $r(t) = r_0 + g[L(t)]$ .

**The P-part:** Rates become converted to spikes by means of a Poisson process (or, if more appropriate to describe an experiment, a different stochastic process!).

**Together, the linear convolution, the non-linear transfer function, and the Poisson process define the LNP model.**

### Remarks:

- Integration boundaries  $0 \dots \infty$  if kernel defined for positive  $\tau$  only, or  $-\infty \dots + \infty$  otherwise.
- Good description for neuronal responses in periphery of the brain.
- $D$  might be referred to in different terms: kernel, impulse response, Greens' function...
- Extension: higher-order Kernels (Wiener/Volterra series).
- The convolution operation is similar to synaptic transmission (earlier example from exercises/lecture):

$$I_{total}(t) = \int I_{resp}(\tau)\rho(t - \tau)d\tau \quad (110)$$

$$L(t) = \int D(\tau)s(t - \tau)d\tau \quad (111)$$

Input:  $s, \rho$ ; Kernel:  $I_{resp}, D$ ; Output:  $I_{total}, L$

### 5.3 Spike-triggered average and reverse correlation

**Problem:** How do we find the “right” kernel for a LNP model?

To solve this problem, consider we have shown some stimulus, and collected spikes from many trials. Now we want to find out: What was the average stimulus preceding these spikes a time  $\tau$  (i.e., the **Spike-Triggered-Average**)?

Let the stimulus be  $s(t)$ , and spikes be observed at  $t_i$  in an interval from 0 to  $T$ :

$$\text{STA}(\tau) = \left\langle \frac{1}{N} \sum_i^N s(t_i - \tau) \right\rangle \quad (112)$$

$\langle \dots \rangle$  indicates an average over trials. If  $N$  is large, the total number of spikes per trial is well approximated by the average number  $\langle N \rangle$  of spikes per trial:

$$\text{STA}(\tau) \approx \frac{1}{\langle N \rangle} \left\langle \sum_i^N s(t_i - \tau) \right\rangle \quad (113)$$

By replacing the sum over spikes with an integral over time, and by using the relation  $\langle r \rangle T = \langle N \rangle$ , we can relate  $\text{STA}$  to the instantaneous firing rate  $r(t)$ :

$$\text{STA}(\tau) = \frac{1}{\langle N \rangle} \int_0^T s(t - \tau) \left\langle \sum_i^N \delta(t - t_i) \right\rangle dt \quad (114)$$

$$\begin{aligned} &= \frac{1}{\langle N \rangle} \int_0^T s(t - \tau) \langle \rho(t) \rangle dt \\ &= \frac{1}{\langle r \rangle T} \int_0^T s(t - \tau) r(t) dt =: \text{RC}(\tau) \end{aligned} \quad (115)$$

which is termed **Reverse Correlation (RC)**.

Next, we show that under certain conditions the STA/RC gives us an estimate for the kernel  $D$  in a LNP-model:

For this purpose, let us assume a rate  $r(t)$  was generated by a LNP (...and let us forget the non-linearity  $g[\dots]$  for a moment):

$$r(t) = r_0 + \int D(\tau) s(t - \tau) d\tau \quad (116)$$

Insert  $r(t)$  into equation (115) for the RC:

$$\text{RC}(\tau) = \frac{1}{\langle r \rangle T} \int \left( r_0 + \int D(\tau') s(t - \tau') d\tau' \right) s(t - \tau) dt \quad (117)$$

Now let us assume that for the stimulus  $s(t)$ , we used uncorrelated noise with zero mean. Such a signal has the (intuitive!) properties:

$$\frac{1}{T} \int s(t) s(t - \tau) dt = \sigma^2 \delta(\tau) \quad \text{and} \quad \int s(t) dt = 0 \quad (118)$$

where  $\sigma^2$  is the signal's variance over time. Since this signal has zero mean, the term with  $r_0$  in the equation above vanishes. We now substitute  $z := t - \tau'$  for  $t$  and obtain:

$$\text{RC}(\tau) = \frac{1}{\langle r \rangle T} \int \left( \int s(z) s(z + \tau' - \tau) dz \right) D(\tau') d\tau' \quad (119)$$

$$= \frac{\sigma^2}{\langle r \rangle} \int D(\tau') \delta(\tau - \tau') d\tau' \quad (120)$$

$$= \frac{\sigma^2}{\langle r \rangle} D(\tau) \quad (121)$$

This holds in more general terms, too, if we do **not** ignore the point nonlinearity or gain function of the model,  $r(t) = r_0 + g[L(t)]$  (Bussgang's Theorem).

### Extension:

If the stimulus is NOT white noise, we have to incorporate the inverse of the autocorrelation matrix into the equation for computing the STA/RC, in order to “take out” the correlations over time (or space...) contained in the stimulus  $s(t)$  which could “tarnish” our estimate for  $D$ .

## 5.4 Multivariate Inputs/Multidimensional Receptive Fields

LNP models can easily be extended to more than one input dimension:

$$r(t) = r_0 + g \left[ \int_0^\infty s(t - \tau) D(\tau) d\tau \right] \quad (122)$$

$$\Rightarrow r(t) = r_0 + g \left[ \int \int \dots \int s(t - \tau, x, y, \dots) D(\tau, x, y, \dots) dx dy d\tau \right] \quad (123)$$

Conversely, STA and RC become

$$\text{STA}(\tau) = \frac{1}{N} \sum_i^N s(t_i - \tau) \quad (124)$$

$$\Rightarrow \text{STA}(\tau, x, y, \dots) = \frac{1}{N} \sum_i^N s(t_i - \tau, x, y, \dots) \quad (125)$$

$$\text{RC}(\tau) = \frac{1}{\langle r \rangle T} \int_0^T s(t - \tau) r(t) dt \quad (126)$$

$$\Rightarrow \text{RC}(\tau, x, y, \dots) = \frac{1}{\langle r \rangle T} \int_0^T s(t - \tau, x, y, \dots) r(t) dt \quad (127)$$

$$(128)$$

### (a) Retinal Ganglion Cells, LGN cells

- ON-OFF structure
- similar to Mexican Hat function (difference of Gaussians, extended from 1D to 2D)
- How do ON-OFF cells emerge?  $\Rightarrow$  Lateral Inhibition!

### (b) Cortical Simple Cells (Separability)

- Gabor functions
- How do simple cells emerge?  $\Rightarrow$  Oriented Superposition of ON-OFF-cells!
- **Separability:**

$$D(x, y, \tau) = D_{\text{spatial}}(x, y) D_{\text{temporal}}(\tau)$$

- Simple cells have separable RFs!

### (c) Cortical Complex cells

- **Problem:** they are not a linear combination of simple cells...
- **Solution:** describe as a non-linear combination of simple cells!

### (d) Inseparable receptive fields

- Can not be described by a space-time separable kernel  $D$
- They explain direction selectivity (e.g. in area MT)

(e) Higher-order kernels: Volterra and Wiener series

- Needed, for example, if a response to two dots at different positions in RF is NOT a superposition of the single responses
- However, they are difficult to estimate,  $D(x, \dots) \implies D(x, x', \dots)$ , since much more statistics is needed, potentiating experimentation time

## 5.5 Abstracting Receptive Fields: Tuning Curves

**We observe:**

Neurons respond to certain “patterns” in the sensory input (e.g. ON-OFF, Gabors). In everyday use, one narrows down the notion of receptive fields to particular features of these patterns. There are plenty of putative features: orientation, direction, depth (binocular vision), pixel luminance, frequency/pitch of a sound, pressure on a part of the skin, etc...

**Hence:** Parametrize these patterns by their key features, e.g. contrast  $c$ , orientation  $\phi$ ...

$\implies$  stimulus  $s \longrightarrow$  feature  $\phi \longrightarrow$  tuning function  $f \longrightarrow$  rate  $r$ : response  $r = f(\phi)$ .

**Take care! Noise:**

Normally, we do not observe  $r$ , but some spike count per time interval  $T$ , maybe repeatedly.

$\implies$  Luckily, we know already a lot about the statistics of those spikes! Thus we can identify potentially underlying processes, we know about their distributions, etc.

**Example:** Orientation tuning in visual cortex

Poisson  $\longrightarrow$  approximate by Gaussian  $\longrightarrow$  variability higher around maximum  $\longrightarrow$  hint to problem when reconstructing

**Observation:** Again: Features for which neuron is more active, features for which it is silent.

**Receptive Field:** Ensemble of all possible features  $\phi^*$  to which a neuron responds.

**Preferred Feature:** The “feature”  $\phi_{\text{optimal}}$  for which the neuron responds most strongly.

## 6 FROM MEMBRANE POTENTIALS TO THE INTEGRATE-AND-FIRE NEURON

Let us consider the dynamics of a neuron: e.g., what happens with an input current  $I$  into a neuron with membrane potential  $V$ ?

**Needs:** Novel mathematical concept. Membrane potential  $V$  not only depends on input current,  $V(t) = g[I(t)]$ , but also on itself:

$$V(t) = g[V(t), I(t)] \quad \text{This is a dynamical system!!!} \quad (129)$$

### 6.1 The integrate-and-fire neuron

The integrate-and-fire (IAF) neuron is described by an ordinary, inhomogeneous, linear DEQ, similar to equations describing decay processes in nature (e.g. radioactive decay):

$$\tau \frac{dV(t)}{dt} = -V(t) + V_{rest} + RI(t) \quad (130)$$

If  $V(t)$  crosses the threshold  $V_{thr}$ , it is reset to  $V_{rest}$  (membrane's resting potential, typically about -70mV), and an action potential is transmitted (to all neurons connected to the axon).

#### 6.1.1 Analytical Solution

Analytical solution is possible to obtain for a constant input current  $I$  (neglect threshold for a moment)

$$\frac{dV}{V - V_{rest} - RI} = -\frac{dt}{\tau} \quad (131)$$

$$\ln(V - V_{rest} - RI) = \frac{-t}{\tau} + const. \quad (132)$$

$$V - V_{rest} - RI = \exp(const.) \exp\left(-\frac{t}{\tau}\right) \quad (133)$$

$$\longrightarrow V(t) = V_{rest} + RI + \exp(const.) \exp\left(-\frac{t}{\tau}\right) \quad (134)$$

Initial conditions: Let's assume  $V(t = 0) = V_0$ :

$$V(t = 0) = V_{rest} + RI + \exp(const.) = V_0 \quad (135)$$

$$\longrightarrow \exp(const.) = V_0 - V_{rest} - RI \quad (136)$$

$$\longrightarrow V(t) = V_{rest} + RI + (V_0 - V_{rest} - RI) \exp\left(-\frac{t}{\tau}\right) \quad (137)$$

### 6.1.2 Examples – How the IAF neuron works

- (a) constant  $I$
- (b) consider  $t \longrightarrow \infty$ :  $V_\infty := V(t \longrightarrow \infty) = V_{rest} + RI$ , substitute  $V_\infty$  into equation (137)
- (c) draw dynamics for  $V_0 > V_\infty$ ,  $V_0 < V_\infty$
- (d) consider different  $\tau$ :  $\tau$  is the time  $t$  it takes for  $V_0 - V_\infty$  to be reduced to  $1/e$  (draw for different  $\tau$ )
- (e) consider spiking threshold  $V_{thr}$ :

$$V_\infty \leq V_{thr} \quad (\text{e.g., } I \text{ too small}) \longrightarrow \text{no spikes} \quad (138)$$

$$V_\infty > V_{thr} \quad (\text{e.g., } I \text{ large enough}) \longrightarrow \text{regular spikes} \quad (139)$$

There is a “normalized” version of the IAF neuron: Introduce the relative membrane potential:

$$v = V - V_{rest}:$$

$$\tau \frac{dV}{dt} = -(V - V_{rest}) + RI(t) \quad (140)$$

$$\longrightarrow \tau \frac{dv}{dt} = -v + RI(t) \quad (141)$$

The threshold is now  $v_{thr} = V_{thr} - V_{rest}$ , and  $v = 0$  means that the membrane is at rest,  $V = V_{rest}$ .

With  $\tau = RC$ , a different form of the DEQ results:

$$C \frac{dV}{dt} = -\frac{1}{R}(V - V_{rest}) + I(t) \quad (142)$$

$$= -g(V - V_{rest}) + I(t) \quad (143)$$

When handling equations with many different variables and parameters, it's always wise to do a

consistency check by having a look at the physical units (on the left and right hand sides):

$$v : [V] \quad \text{Volts} \quad (144)$$

$$I : [A] \quad \text{Amps} \quad (145)$$

$$g : [A/V] \quad \text{from: Ohm's Law} \quad (146)$$

$$C : [As/V] \quad \text{from: } Q = CU \quad (147)$$

In this chapter, the IAF neuron “fell from the sky” ... Questions:

- How does the IAF neuron derive from neuroelectronics?
- Why is the resting potential at about -70mV?
- Can we describe the spiking mechanism explicitly, without referring to an “extra” threshold mechanism?

To answer these questions, we have to understand the neuronal “electronics”!

## 6.2 Cell membranes, ions, currents and potentials

For a deeper understanding how neurons work, one must have a closer look at the exchange processes of charged particles (ions) across a cell's membrane. These will tell us how the membrane potential changes over time, how action potentials are generated, and how synaptic inputs are integrated.

Parts of a cell membrane:

- (a) Lipid double layer (hydrophobic-inside/hydrophilic-outside)
- (b) Protein molecules as “channels”, “gates” or “receptors”

### 6.2.1 Leakage currents

#### a) All cells are polarized:

In general, there exists a potential difference  $V$  between the inside and outside of a cell:

$$V = j_{inside} - j_{outside} \quad j: \text{electrical potential} \quad (148)$$

When the cell is currently not active and does not receive external inputs, there is an excess of positive charges on the outside. The corresponding membrane potential  $V = V_{rest}$  is called the *resting potential*.

## b) The resting potential:

How is it generated, and which dynamical processes contribute?

The resting potential is caused by the interplay of different dynamical processes acting on sodium ions ( $Na^+$ ) and potassium ions ( $K^+$ ) – numbers in brackets give the typical concentrations in the squid giant axon at temperature of  $T = 20^\circ$ , which was used by Hodgkin and Huxley ([?]) to study the dynamics of neurons and generation of action potentials:

- Outside: more  $Na^+$  than inside (440:50) mM/l
- Inside: more  $K^+$  than outside (20:400) mM/l

Due to the different concentrations and electric charges, there are two forces pushing these ions:

⇒ diffusion tries to equilibrate concentrations

⇒ electrical field causes ionic currents

The sum of these currents is named **leakage** or **resting current**. It would normally equilibrate the concentrations until  $V = 0$ .

⇒ *Ion pumps* must counteract these two processes ('electrodiffusive currents') to keep up the resting potential, i.e. by exchanging 3  $Na^+$  against 2  $K^+$

## c) Quantitative description of the electrodiffusive currents:

Ion movement, or in other words a current, will be determined by an existing potential difference  $V$  and a concentration gradient between  $c_o$  and  $c_i$ :

- Units:
  - larger surface, larger currents ⇒ current density [ $A/m^2$ ]
  - also: specific conductance [ $S/m^2$ ] or [ $1/Ohms\ m^2$ ]
- Ionic current  $J$  → electric current  $I$  (for one type of ions)

$$J = Pc \left[ \frac{zF}{RT} V + \ln \left( \frac{c_i}{c_o} \right) \right] \quad (149)$$

$$I = PczF \left[ \frac{zF}{RT} V + \ln \left( \frac{c_i}{c_o} \right) \right] \quad (150)$$

$P$  = Permeability [ $m/s$ ]

$c, c_i, c_o$  = mean/inside/outside concentration [ $M/m^3$ ]

$z$  = valency [1]

$F$  = Faraday-constant,  $9.6e4$  [ $C/M$ ] = [ $As/M$ ]

$R$  = gas constant,  $8.3$  [ $J/MK$ ] = [ $VAs/MK$ ]

$T$  = temperature [ $K$ ]

## d) Interpretation:

- o  $I$  can be considered as composed of two forces: diffusive force and electrical force:
- o electrical force:
  - positive  $V \implies$  current to outside, independent of valency (positive ions to outside, negative ions to inside)
  - negative  $V \implies$  current to inside, independent of valency (positive ions to inside, negative ions to outside)
- o diffusive force:
  - $c_i > c_o \implies$  diffusion to outside
  - $c_i < c_o \implies$  diffusion to inside

### e) The Nernst equation:

Under which conditions is the electrodiffusive current  $I = 0$ ?

$$\frac{zF}{RT}V + \ln\left(\frac{c_i}{c_o}\right) = 0 \quad (151)$$

$$\longrightarrow \frac{c_i}{c_o} = \exp\left(-\frac{zF}{RT}V\right) \quad (152)$$

$$\longrightarrow V = -\frac{RT}{zF} \ln\left(\frac{c_i}{c_o}\right) = E(c_i, c_o) \quad (153)$$

This equation is termed the **Nernst equation** for pure electrodiffusion through a cell membrane. For a given ion concentration gradient, it defines a voltage  $V$  for which there would be no net overall current (diffusive and electric forces equilibrate).

This condition is certainly not fulfilled in the resting state of a neuron, because ion concentrations for the various sorts of ions are all different.

### f) Conductances and reversal potentials:

We will now simplify the expression for the electrodiffusive current:

$$I = PczF \left[ \frac{zF}{RT}V + \ln\left(\frac{c_i}{c_o}\right) \right] \quad (154)$$

$$\longrightarrow I = PczF \frac{zF}{RT} \left[ V + \frac{RT}{zF} \ln\left(\frac{c_i}{c_o}\right) \right] \quad (155)$$

By defining  $g := PczF(zF/(RT))$ , and substituting expression  $E(c_i, c_o)$  from **e)**, we obtain:

$$\longrightarrow I = g[V - E(c_i, c_o)] \quad (156)$$

The first term is actually Ohm's law,  $I = V/R = gV$ ,  $g : [S/m^2]$ , while the second term defines the so-called **reversal potential**  $E(c_i, c_o)$ .

**Interpretation of  $E(c_i, c_o)$ :**

It is a (virtual) **equilibrium potential** with which the current concentrations would be in balance (no net overall current, diffusive and electric forces equilibrate), if it would be the current potential. The name **reversal potential** was given since the net overall current reverses its flow if the actual potential crosses the reversal potential.

**g) Approximation by constant ion concentrations:**

$E(c_i, c_o)$  are dependent on  $c_i$  and  $c_o$ , which in turn change with the membrane potential...

...but there is the following **observation**: While the total balance between all ions is subject to large changes, the **absolute concentrations** do not vary strongly.

$$\longrightarrow E(c_i, c_o) \approx E(c_{i,rest,X}, c_{o,rest,X}) =: E_X \quad (157)$$

$$\longrightarrow I_X(t) = g_X * [V(t) - E_X] \quad (158)$$

For the squid giant axon, the conductances are approximately  $g_K = 0.008 \text{ S/cm}^2$  and  $g_{Na} = 0.001 \text{ S/cm}^2$ .

**WE HAVE SO FAR...:**

...understood leakage currents, now we have to include **ion pumps** and **gated currents** to arrive at a complete description of a neuron:

## 6.2.2 Kirchhoff's Law and Membrane Potential Dynamics

There are three currents through a membrane:

(a) leakage currents,  $I_{Na}^L, I_K^L$  ( $I_{Ca}^L$ )

(b) pump currents,  $I_{Na}^P, I_K^P$

(c) gated currents (voltage-gated, ligand-gated),  $I_{Na}^V, I_K^V$  ( $I_{Ca}^V$ ) and  $I_{Na}^T, I_K^T$  ( $I_{Ca}^T$ )

One can summarize all currents of a specific ion type  $X$ :

$$I_X(t) = I_X^P + (g_X^L + g_X^V + g_X^T)[V(t) - E_X] \quad (159)$$

$$= I_X^P + g_X[V(t) - E_X] \quad (160)$$

...with the total conductance  $g_X := g_X^L + g_X^V + g_X^T$ .

**a) Dynamics:**

Kirchhoff's law: conservation of electric charge:

- at a junction, all input currents must equal all output currents:

$$\sum_i I_{input}^i = \sum_i I_{output}^i \quad (161)$$

- ...or phrased differently: all currents must sum to 0!

$$\sum_i I_{input}^i - \sum_i I_{output}^i = 0 \quad (162)$$

We consider three currents:

- $I_{ionic}^{total}$ : total ionic currents (leakage, pump, gated/synaptic, of all sorts of ions)
- $I = I_{inject}$ : current injected into a neuron (e.g. during a current clamp experiment)
- $I_C$ : current charging the membrane (capacitor)

$$\longrightarrow I = I_{inject} = I_C + I_{ionic}^{total} \quad (163)$$

How does a capacitor behave? A capacitor has a capacity  $C$ , and the charge  $Q$  is related to voltage  $V$  via  $Q = CV$ :

Take derivative wrt. time:  $dQ/dt = I_C = C dV/dt$ :

$$\longrightarrow C \frac{dV}{dt} = - \sum_X [I_X^P + g_X^L(V(t) - E_X) + (g_X^V + g_X^T)(V(t) - E_X)] + I \quad (164)$$

Now, remember that for each sort of ions  $X$ , the pump currents and leakage currents **must compensate if**  $V = V_{rest}$ :

$$\longrightarrow I_X^P + g_X^L(V_{rest} - E_X) = 0 \quad (165)$$

From this equation follows:

$$\longrightarrow I_X^P + g_X^L(V(t) - E_X) = I_X^P + g_X^L(V(t) - V_{rest} + V_{rest} - E_X) \quad (166)$$

$$= g_X^L(V(t) - V_{rest}) + I_X^P + g_X^L(V_{rest} - E_X) \quad (167)$$

$$= g_X^L(V(t) - V_{rest}) \quad (168)$$

In words: the pump currents vanish if we express the leakage currents w.r.t. to the resting

potential  $V_{rest}$ !

$$\longrightarrow C \frac{dV}{dt} = - \sum_X [g_X^L (V(t) - V_{rest}) + (g_X^V + g_X^T)(V(t) - E_X)] + I \quad (169)$$

$$= - \left( \sum_X g_X^L \right) (V(t) - V_{rest}) - \sum_X (g_X^V + g_X^T)(V(t) - E_X) + I \quad (170)$$

with  $g_X^{total} := \sum_X g_X^L = 1/R$ , we obtain:

$$\longrightarrow \frac{dV}{dt} = - \frac{1}{R} (V(t) - V_{rest}) - \sum_X (g_X^V + g_X^T)(V(t) - E_X) + I \quad (171)$$

This is identical to the Integrate-and-Fire-neuron, except for the gated currents:

$$\longrightarrow C \frac{dV}{dt} = - \frac{1}{R} (V(t) - V_{rest}) + I(t) \quad (172)$$

### 6.2.3 Computing the resting potential

How do we compute the resting potential  $V_{rest}$ ? Let us first remember that the ion pumps in the cell membrane pump 3  $Na^+$  ions outside for each 2  $K^+$  ions pumped inside. The resting potential will be the potential at which the electrodiffusive currents  $I_{Na}$  and  $I_K$  balance the pump currents, i.e.  $I_K = -2/3 I_{Na}$ :

$$I_{Na}(V) = g_{Na}(V - E_{Na}) \quad (173)$$

$$I_K(V) = g_K(V - E_K) \quad (174)$$

$$\longrightarrow I_K(V_{rest}) = -\frac{2}{3} I_{Na}(V_{rest}) \quad (175)$$

A short calculation yields:

$$V_{rest} = \frac{\frac{3}{2} g_K E_K + g_{Na} E_{Na}}{\frac{3}{2} g_K + g_{Na}} \quad (176)$$

## 7 CONDUCTANCE-BASED NEURON MODELS: FROM HODGKIN-HUXLEY TO FITZHUGH-NAGUMO

### 7.1 Hodgkin-Huxley Model

The DEQ

$$C \frac{dV}{dt} = -\frac{1}{R}(V(t) - V_{rest}) - \sum_X (g_X^V + g_X^T)(V(t) - E_X) + I \quad (177)$$

is the fundamental dynamical equation for a neuron. But we still need to describe the dynamics of voltage-gated ion channels:

#### 7.1.1 Voltage-gated ion channels

Consider the voltage-dependent dynamics of the sodium and potassium channels  $g_X^V$  ( $g_X^T = 0$ ):

$$g_K^V = g_K^V(V) = g_K^{max} n(V)^4 \quad (178)$$

$$g_{Na}^V = g_{Na}^V(V) = g_{Na}^{max} m(V)^3 h(V) \quad (179)$$

Its dynamics are described by three **gating variables**  $h, m, n \in [0, 1]$ . Interpretation:  $k$  independent events have to occur for opening the channel. The dynamics of gating variables are again described by DEQs:

$$\frac{dh}{dt} = f_h(V, h) \quad (180)$$

$$\frac{dm}{dt} = f_m(V, m) \quad (181)$$

$$\frac{dn}{dt} = f_n(V, n) \quad (182)$$

In consequence, our neuron model becomes a system of four first-order, non-linear DEQs:

$$\frac{dV}{dt} = f_V(V, h, m, n) \quad (183)$$

$$\frac{dh}{dt} = f_h(V, h, m, n) \quad (184)$$

$$\frac{dm}{dt} = f_m(V, h, m, n) \quad (185)$$

$$\frac{dn}{dt} = f_n(V, h, m, n) \quad (186)$$

...can easily be solved numerically, although analytical solution is not possible!

### Dynamics of gating variables:

For any gating variable  $x$ , the DEQ takes the following form ( $\alpha$  = **opening rate**,  $\beta$  = **closing rate**):

$$\frac{dx}{dt} = \alpha(1 - x) - \beta x \quad (187)$$

The steady-state solution  $x_{Inf}$  is given by:

$$\alpha = (\alpha + \beta)x_{Inf} \quad (188)$$

$$x_{Inf} = \frac{\alpha}{\alpha + \beta} \quad (189)$$

Thus, if we write the DEQ in terms of  $x_{Inf}$ :

$$\frac{dx}{dt} = \alpha - (\alpha + \beta)x \quad (190)$$

$$\frac{1}{\alpha + \beta} \frac{dx}{dt} = -x + x_{Inf} \quad (191)$$

### Remark:

This DEQ has a formal equivalence to the integrate-and-fire neuron without threshold!

Of course,  $\alpha(V)$  and  $\beta(V)$  depend on voltage  $V$ . Otherwise  $x$  would not depend on  $V$  anymore...

Examples of model neurons using these equations: Hodgkin-Huxley, Connor-Stevens.

## 7.1.2 Hodgkin-Huxley Parameters/Equations

The dynamics of an action potential:

- ...sodium channels open first  $\implies$  push towards  $E_{Na}$  (positive!)
- ...potassium channels open next  $\implies$  pull towards  $E_K$  (negative!)
- ...action potential suppressed, sodium closes again, then also potassium.

## 7.1.3 Simplifying Hodgkin-Huxley

Focus on changes of conductances during action potential:

- ...changes are very fast compared to usual membrane potential dynamics: gating variable

$m$  ( $Na^+$ ) is fastest!

- o ...gating variables  $n$  and  $h$  lag behind.

**First approximation:** Replace  $m$  by  $\langle m \rangle$ .

**Second approximation:** Should we also replace  $n, h$  by  $\langle n \rangle, \langle h \rangle$ ?

- o ...no, because action potential would be terminated the first time it would “try” to occur...
- o ...better: introduce a second, “dummy”, voltage variable  $U$  which lags behind the original voltage  $V$ . Write  $n$  and  $h$  as functions (not DEQs!) of  $U$ .

This procedure leads **generally** to two-dimensional models with variables  $(V, U)$  whose dynamics are described by two nonlinear DEQ's. We will study an example in the next subsection!

## 7.2 The Fitzhugh-Nagumo neuron

As an example, introduce the Fitzhugh-Nagumo (FHN) model. It has the same flavour as the approximation introduced above, but was derived differently. For historical reasons, this model is considered here as an example of many 2D-spiking models.

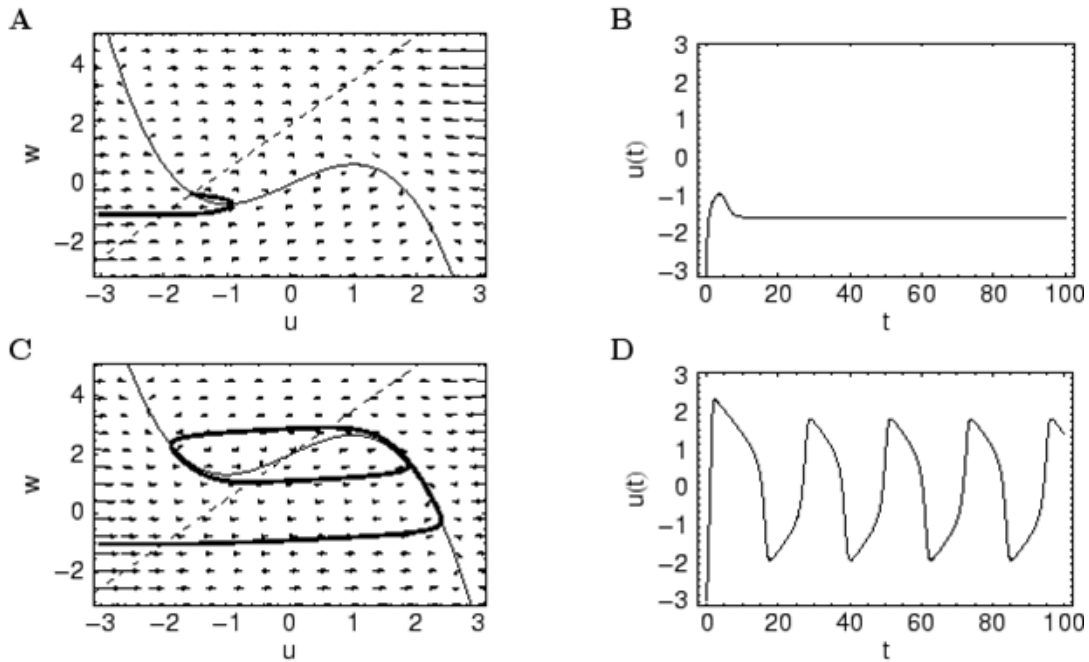
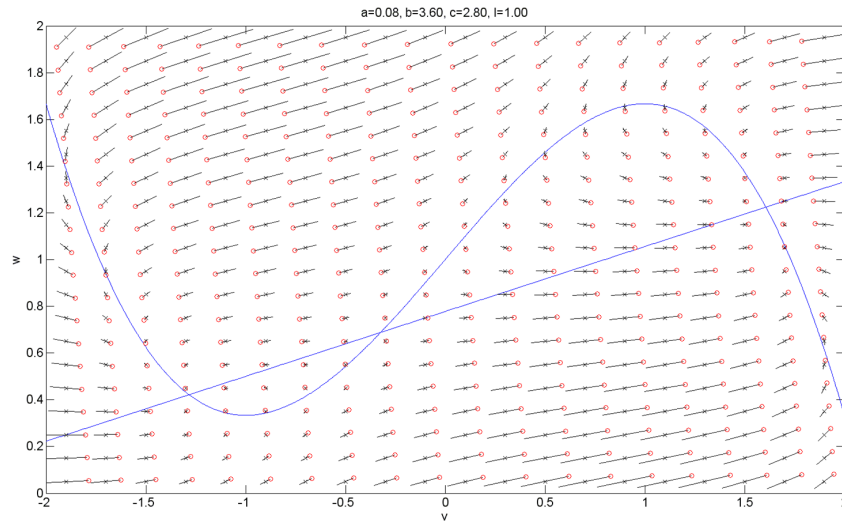
$$\frac{dv(t)}{dt} = v(t) - \frac{1}{3}v^3(t) - w(t) + I(t) \quad (192)$$

$$\frac{dw(t)}{dt} = a(v(t) - bw(t) + c) \quad (193)$$

Let's solve it! Hmmmmm... difficult, since the equations are nonlinear... is it impossible?

No! The dynamics of the FHN-model can be analyzed by performing a **linear stability analysis** (see Appendix). Let's assume for that  $I = \text{const.} = 1$ . We obtain two **nullclines** that depend on the parameters  $a, b$ , and  $c$ . Here we can distinguish between several generic cases:

- (a)  $\dot{w}$ -nullcline steep or shallow
- (b)  $\dot{v}$ -nullcline low or high 'amplitude'



Correspondingly, we have three qualitatively different modes:

- (a)  $a = 0.08, b = 3.6, c = 5.6$ : initial activation decays, or neuron fires one 'spike'  $\Rightarrow$  excitable regime
- (b)  $a = 0.08, b = 0.8, c = 0.7$ : system goes into limit cycle  $\Rightarrow$  regular spike trains
- (c)  $a = 0.08, b = 3.6, c = 2.8$ : existence of two stable and one unstable fixed points  $\Rightarrow$  goes into either high or low voltage state, thus multistability in dependence on initial conditions

## 8 AXONS, DENDRITES AND SYNAPSES

Literature:

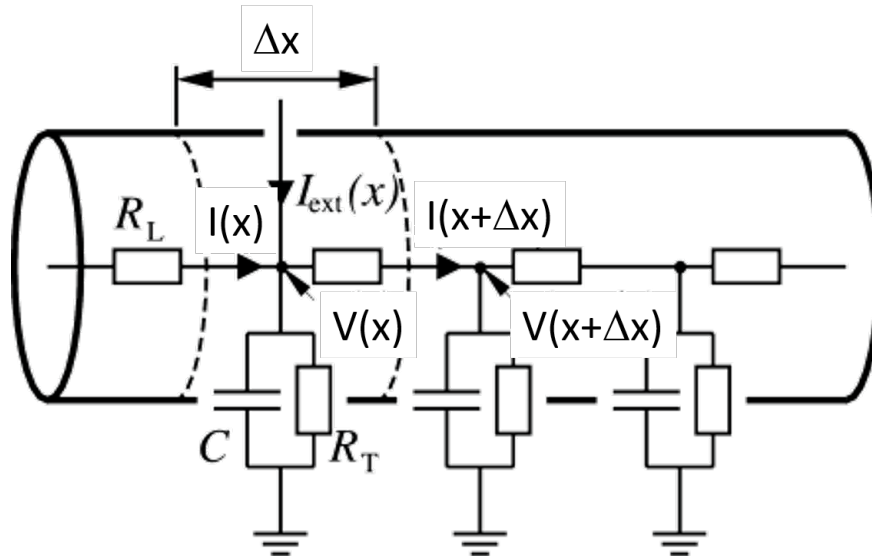
Gerstner et al., from page 58

Abbott & Dayan, from page 203

### 8.1 Axonal and Dendritic Transmission

#### 8.1.1 Passive cable equations

Dendrite or axon can be considered as a cylindric membrane which is composed of different segments.



(modified from Gerstner book)

Mathematically, the membrane potential dynamics of each segment (length  $\Delta x$ ) is described by equations similar to the Hodgkin-Huxley model. Each segment has a longitudinal resistance  $R_L$ , currents flow along the segment according to Ohm's law.

Approach: Kirchhoff's law: incoming and outgoing currents must sum to 0:

Incoming: current from the left (negative  $x$ -axis)  $I(t, x)$  and external current  $I_{ext}$  Outgoing: current to the right (positive  $x$ -axis)  $I(t, x + \Delta x)$ , membrane charging current  $I_m$  and ionic currents  $\sum_x I_x$

Take care: Current densities **across** membrane have to be multiplied by  $\Delta x$  to yield total

current, currents **along** the cable are total currents:

$$I(t, x) + \Delta x I_{ext}(t, x) = I(t, x + \Delta x) + \Delta x C \frac{\partial V(t, x)}{\partial t} + \Delta x \sum_X I_X(t, x)$$

Put currents along cable to one side:

$$-\frac{I(t, x + \Delta x) - I(t, x)}{\Delta x} = -I_{ext}(t, x) + C \frac{\partial V(t, x)}{\partial t} + \sum_X I_X(t, x)$$

Thus:

$$-\frac{\partial I(t, x)}{\partial x} = -I_{ext}(t, x) + C \frac{\partial V(t, x)}{\partial t} + \sum_X I_X(t, x)$$

Finally, with  $I = -1/R_L \partial V / \partial x$  (current is positive if flowing into positive  $x$ -direction):

$$\frac{1}{R_L} \frac{\partial^2 V(t, x)}{\partial x^2} = -I_{ext}(t, x) + C \frac{\partial V(t, x)}{\partial t} + \sum_X I_X(t, x)$$

Or, rearranged w.r.t. the temporal derivative:

$$C \frac{\partial V(t, x)}{\partial t} = \frac{1}{R_L} \frac{\partial^2 V(t, x)}{\partial x^2} - \sum_X I_X(t, x) + I_{ext}(t, x)$$

**For a passive membrane, we can assume that the ionic currents are  $1/R_T(V - V_L)$ , where  $R_T$  is the transversal resistance per unit length:**

$$C \frac{\partial V(t, x)}{\partial t} = \frac{1}{R_L} \frac{\partial^2 V(t, x)}{\partial x^2} - \frac{1}{R_T} (V(t, x) - V_L) + I_{ext}(t, x)$$

Then the equation can nicely be scaled to unit-free coordinates:

1. Introduce electrotonic length scale  $\lambda^2 = R_T/R_L$  and membrane time constant  $\tau = R_T C$ :

$$\tau \frac{\partial V(t, x)}{\partial t} = \lambda^2 \frac{\partial^2 V(t, x)}{\partial x^2} - (V(t, x) - V_{rest}) + R_T I_{ext}(t, x)$$

2. Scale  $x \rightarrow x/\lambda$ ,  $t \rightarrow t/\tau$ ,  $I_{ext} \rightarrow R_T I_{ext}$ , and  $V \rightarrow V - V_{rest}$ :

$$\frac{\partial V(t, x)}{\partial t} = \frac{\partial^2 V(t, x)}{\partial x^2} - V(t, x) + I_{ext}(t, x)$$

### 8.1.2 Solving the passive cable equations

Intuitively, the dynamics of the passive cable equation can easily be understood: temporal change in  $V$  is composed of a diffusive term, a decay term, and an external source term.

For analytically solving the equation, we have to find its Green's function (solution to a 'kick', or  $\delta$ -pulse at  $t = 0$  and  $x = 0$ ):

$$I_{ext}(t, x) = \delta(x)\delta(t)$$

Now we employ an often-used trick when confronted with partial differential equations: transformation into Fourier space w.r.t.  $x$ ,

$$v(t, k) := \frac{1}{\sqrt{2\pi}} \int V(t, x) \exp(-ikx) dx$$

Remember,  $t$  is untouched by Fourier transform (FT) w.r.t.  $x$ . Now,  $I_{ext}$  gives a constant times  $\delta(t)$ . FT of  $\partial V/\partial x$  gives  $ikFT(V)$ . Thus we obtain:

$$\frac{\partial v(t, k)}{\partial t} = -k^2 v(t, k) - v(t, k) + \delta(t)/\sqrt{2\pi}$$

Let's assume  $v$  is zero for negative  $t$ . At  $t = 0$ , it is then instantly raised to  $v(t^+, k) = 1/\sqrt{2\pi}$ . This is the initial condition for the now again homogeneous differential equation, which has the following solution for  $t > 0$ :

$$v(t, k) = \delta(t)/\sqrt{2\pi} \exp(-(1 + k^2)t)$$

Inverse Fourier transform leads to the solution:

$$V(t, x) = \frac{1}{4\pi t} \exp\left(-t - \frac{x^2}{4t}\right) = G_{inf}(t, x)$$

**Interpretation:** Gaussian profile decaying and broadening over time.

A Green's function tells us how a system behaves if 'kicked' once. To obtain the solution for an

arbitrary input, we need to superimpose the system's reaction to a 'series' of kicks – best done by a corresponding convolution:

$$V(t, x) = \int_{-\infty}^t dt' \int_{-\infty}^{+\infty} dx' G_{inf}(t - t', x - x') I_{ext}(t', x')$$

### 8.1.3 Nonlinear extensions

Ionic currents  $I_X$  can be used to model synaptic inputs to the dendrite via

$$I_{syn}(t, x) = g_{syn}(t, x)(V(t, x) - E_{syn})$$

with appropriate (time-varying) conductances  $g_{syn}$  and reversal potentials  $E_{syn}$ .

### 8.1.4 Unmyelinated axons

Unmyelinated axons can be modelled like dendrites with active ion channels. A similar calculation as for the Hodgkin-Huxley model gives (for the squid giant axon):

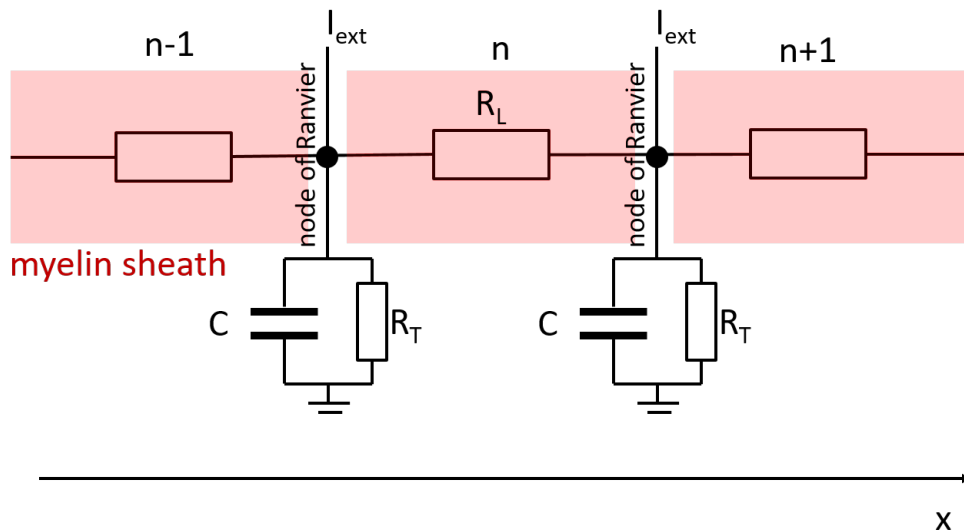
$$C \frac{\partial V(t, x)}{\partial t} = \frac{1}{R_L} \frac{\partial^2 V(t, x)}{\partial x^2} - \frac{1}{R_T} (V(t, x) - V_L) + g_{Na}^{max} m^3(t, x) h(t, x) (V(t, x) - E_{Na}) + g_K^{max} n^4(t, x) (V(t, x) - E_K) + I_{ext}(t, x) \quad (194)$$

### 8.1.5 Myelinated axons

Myelinated axons:  $K$ - and  $Na$ -channels only at nodes of Ranvier (0.2% of axonal length). Everything else covered in myelin sheath (lipid-rich substance). Decreases membrane capacitance  $C$  and increases resistance  $R_T$ . Thus increases conduction velocity from 0.25 m/s to 70 – 80 m/s.

Mathematically, discretization  $dx$  into finite segments of length  $L$ , running index  $n$ :

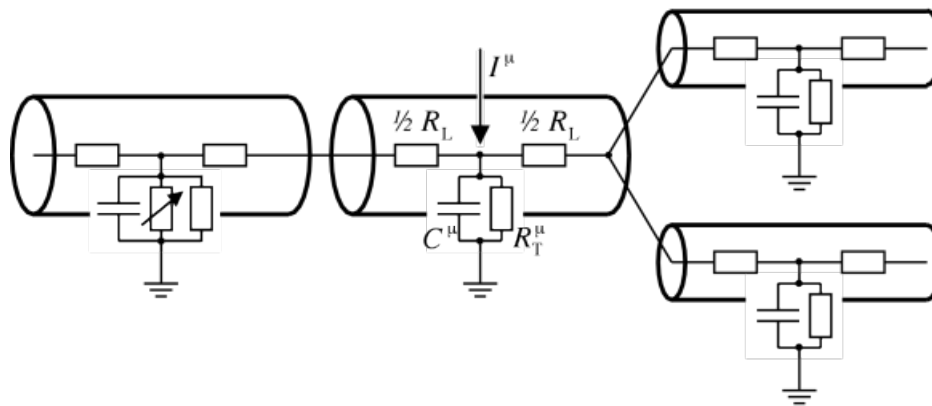
$$C \frac{dV_n(t)}{dt} = \frac{1}{LR_L} (V_{n+1}(t) - 2V_n(t) + V_{n-1}(t)) - \sum_X I_{n,X}(t) + I_{n,ext}(t)$$



### 8.1.6 Multicompartmental models, numerical methods, and boundary conditions

#### 1. Setup

Multicompartmental models composed of multiple segments of passive or active cables. Makes finding analytical solutions difficult.



(from Gerstner book)

Technically: discretize model into many small compartments. These are described by coupled, but ordinary differential equations in time.

Also problematic: explosion in number of free parameters.

#### 2. Numerical solution

For the numerical solution of cable equations, one has to discretize in time and space. Second

derivatives are approximated by differential quotients of first derivatives.

Temporal derivatives can still be approximated by a simple differential quotient and solved, for example, for the (forward) Euler method.

However, often algorithms derived in this simplistic manner are not numerically stable. That means, small perturbations such as imprecision of a numerical computation can exponentially increase and quickly render a solution evolving in time unusable.

For using the above example for  $\partial^2 V / \partial x^2$  with the Euler method, a condition for stability for solving the dimensionless passive cable equation is  $\Delta t < \frac{1}{2} \Delta x^2$ .

If such a condition is unfeasible to implement, because of limited storage space or processing power, more sophisticated methods have to be used. This includes, for example, the *Crank-Nicolson-Method* which can efficiently be implemented using the *Thomas-Algorithm*.

### 3. Boundary conditions

Real axons/dendrites, and simulated ones are not infinitely long. Therefore it is important to specify what happens at the borders of a cable.

For example, sealed ends:  $I(t, x = 0) = I(t, x = L) = 0$  (equivalent statement on  $\partial V / \partial x$ ).

This situation can be solved by a trick from electrostatics (mirror charges): Extend finite cable of length  $L$  by adding an infinite number of chunks with mirrored current sources.

On the ends of the chunks, currents from left and right must cancel such that there is a net current of 0 as required by the sealed-ends condition. Assume a current pulse at  $t_0$  and  $x_0$ :

$$G_{0,L}(t_0, x_0; t, x) = \sum_{n=-\infty}^{+\infty} G_{inf}(t - t_0, x - x_0 - 2nL) + G_{inf}(t - t_0, x + x_0 - 2nL)$$

Again, the solution to an arbitrary input current is given by a superposition:

$$V(t, x) = \int_{-\infty}^t dt' \int_0^L dx' G_{0,L}(t', x'; t, x) I_{ext}(t', x')$$

## 8.2 Synapses

Synapses have a complicated microscopic dynamics, whose formal description is beyond the scope of this course. However, the effect of synaptic transmission onto currents in dendrites (or in the soma, if dendrites are not explicitly modeled), can be described by an ionic conductance

with corresponding reversal potential. This conductance is activated by the transmitters released from the presynaptic terminals:

$$I_{syn}(t) = g_{syn}(t)(V(t) - E_{syn}) \quad (196)$$

$g_{syn}$  is often modeled as a superposition of exponentially decaying functions, e.g.

$$g_{syn}(t) = g_{syn}^{max} \exp(-(t - t_{spike})/\tau)$$

for  $t > t_{spike}$ .

For a more detailed description leading to a smoothly rising and decaying current, an exponential increase is combined with a fast and a slow exponential decrease:

$$g_{syn}(t) = g_{syn}^{max} [1 - \exp(-(t - t_{spike})/\tau_{rise})] \quad (197)$$

$$[a \exp(-(t - t_{spike})/\tau_{fast}) + (1 - a) \exp(-(t - t_{spike})/\tau_{slow})] \quad (198)$$

Examples:

- **GABA<sub>A</sub>** (inhibitory):  $a = 1$ ,  $\tau_{rise} = 1$  ms,  $\tau_{fast} = 6$  ms,  $E_{syn} = -70 \dots -75$  mV
- **GABA<sub>B</sub>** (inhibitory):  $a = 0.8$ ,  $\tau_{rise} = 25 \dots 50$  ms,  $\tau_{fast} = 100 \dots 300$  ms,  $\tau_{slow} = 500 \dots 1000$  ms,  $E_{syn} = -70 \dots -75$  mV
- **AMPA** (excitatory):  $\tau = 2 \dots 5$  ms (equation with exponential decay only),  $E_{syn} = 0$  mV.

Under certain conditions, synaptic interactions can be simplified and lead to neural network models which are easier to analyze and to understand formally: for example, one can make the assumption that transmission of a spike to a postsynaptic cell occurs on a much faster time scale than the membrane potential  $V$  changes. By assuming  $V(t) \approx V(t_{spike})$  during spike transmission, lasting an effective time  $\Delta t$ , the integral over equation (196) becomes:

$$\int_{t_{spike}}^{t_{spike} + \Delta t} I_{syn}(t) dt = (V(t_{spike}) - E_{syn}) \int_0^\infty g_{syn}(t) dt \quad (199)$$

$$= const. (V(t_{spike}) - E_{syn}) \quad (200)$$

That means, the effect of a presynaptic spike on the postsynaptic (or dendritic) cell can be described by an increase (or decrease, for inhibitory couplings) of the membrane potential by an amount proportional to the difference between actual potential and synaptic reversal potential.

If one assumes in addition that  $V(t)$  is most of the time not far from the resting potential (or close to some average membrane potential  $\bar{V}$  above the resting potential, e.g. between  $-60$  and  $-50$  mV), while  $E_{syn}$  is far bigger or smaller than  $\bar{V}$ , one can describe the effect of a presynaptic spike as a *constant and instantaneous increase or decrease* in membrane potential  $V$ , modelled by a  $\delta$ -distribution for the synaptic current (here the equation is given for a series of spikes at times  $t_{spike}$ ):

$$I_{syn}(t) = I_{spike} \sum_{t_{spike}} \delta(t - t_{spike}) \quad (201)$$

## 9 COLLECTIVE PHENOMENA IN SPIKING NETWORKS

### 9.1 Synchronization of pulse-coupled oscillators

#### 9.1.1 Motivation

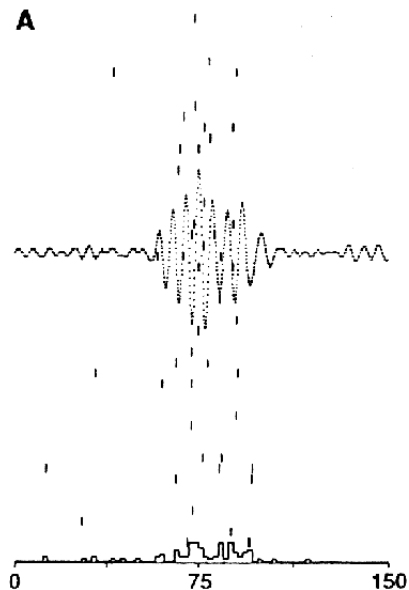
Synchronization of coupled oscillators is a widespread phenomenon:

##### **Physics:**

Coupled pendula (Huygens, Horologium oscillatorium 1673)

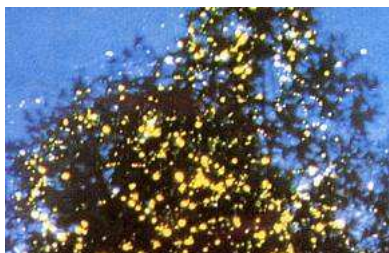
##### **Neurophysiology:**

High-frequency oscillations (Buzsaki et al.)

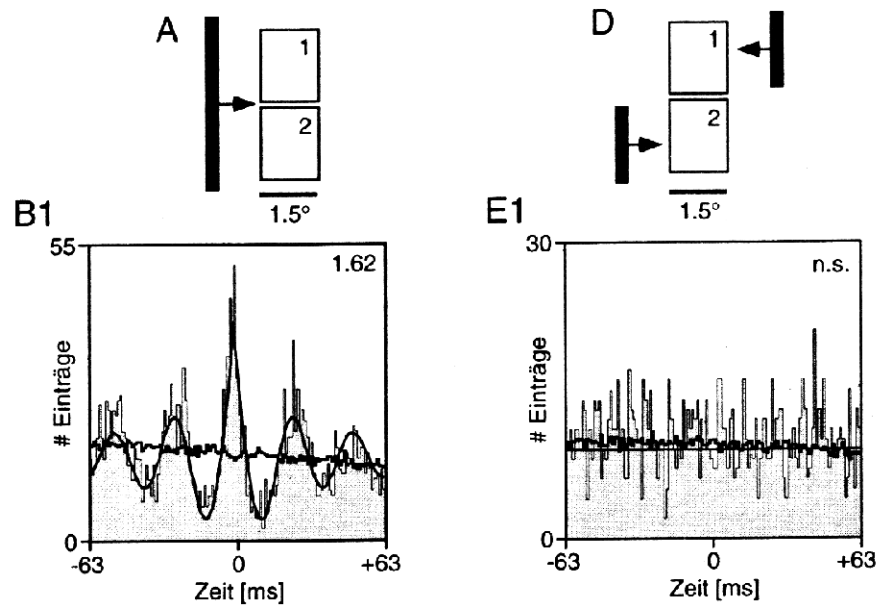


##### **Biology:**

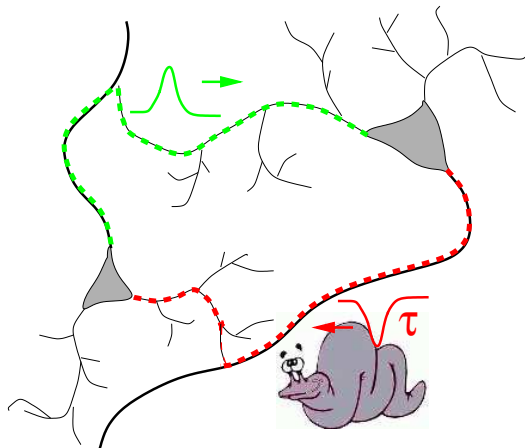
Groups of fireflies (Buck, Copeland, ...)



Synchronization may be related to perception:



Okay, let's analyze pulse-coupled neurons with delays!



### 9.1.2 Oscillator model

(Normalized) integrate-and-fire (IAF) neuron with constant, suprathreshold synaptic input realizes neuronal oscillator:

$$\begin{aligned} \tau \frac{dV}{dt} &= -V + RI; & V(t=0) &= 0 \\ \rightarrow V(t) &= RI(1 - \exp(-t/\tau)) \end{aligned}$$

The neuron fires if  $V = 1$ , firing period  $T$  is given by

$$T = -\tau \log(1 - 1/(RI)). \quad (202)$$

Time axis can be rescaled  $t \rightarrow \Phi = t/T$ , such that neuron fires at  $\Phi = 1$  ( $\Phi$  defines the phase of the neural oscillator).

**Goal:** What happens if two identical oscillators  $A$  and  $B$  are coupled, either excitatorily or inhibitorily, either with or without delay?

Mathematical description: use a generalized oscillator model. Requirements:

- One oscillator described by function

$$f : \Phi \in [0, 1] \mapsto f(\Phi) \in [0, 1] \quad (203)$$

(for IAF neurons:  $f$  describes evolution of membrane potential).

- $f$  monotonically increasing,  $f' > 0$ .
- $f$  concave up,  $f'' < 0$ .

$\rightarrow$  inverse  $g := f^{-1}$  exists!

Assume simple pulse-coupling to the other oscillator:

**Example:**

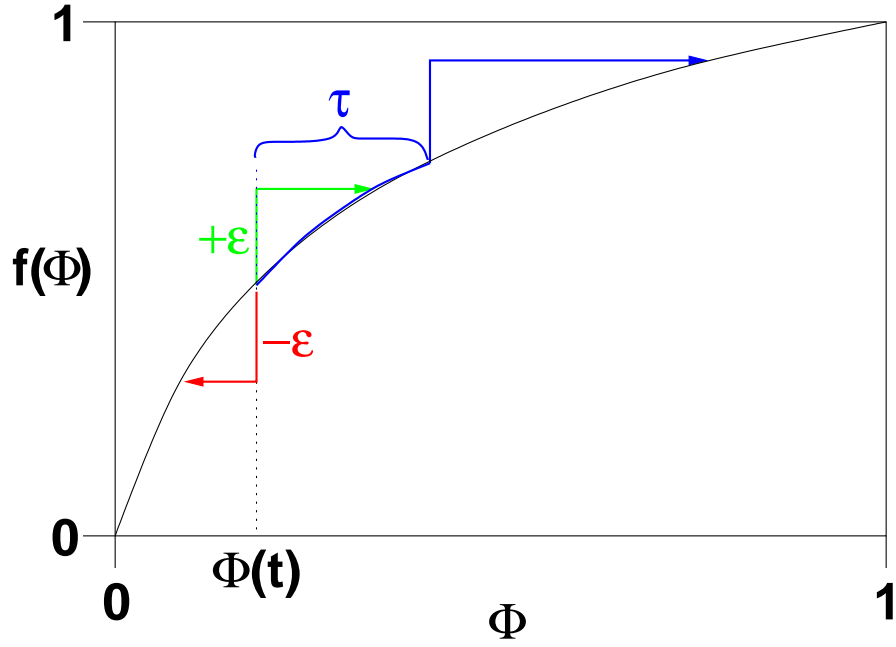
$$\frac{df}{dt} = c + \epsilon \delta(t - t_{fire})$$

Solution obtained by separation of variables, elementary integration:

$$f(t) = f(0) + \begin{cases} ct & : t < t_{fire} \\ ct + \epsilon & : t > t_{fire} \end{cases}$$

Pulse of strength  $\epsilon$  emitted at time  $t$

- **Excitatory** coupling:  $f = f + \epsilon$   
 $\Phi(t) \rightarrow g(f(\Phi(t)) + \epsilon)$
- **Inhibitory** coupling:  $f = f - \epsilon$   
 $\Phi(t) \rightarrow g(f(\Phi(t)) - \epsilon)$



- **Delayed** coupling:  
 $\Phi(t + \tau) \longrightarrow g(f(\Phi(t + \tau)) \pm \epsilon)$

Without delays, state of the system is fully determined by the phases  $\{\Phi_A, \Phi_B\}$  of the two oscillators  $A$  and  $B$ .

### 9.1.3 Fixed points

Due to the pulselike coupling, it is difficult to find a fixed point and determine its stability by a linear stability analysis. Instead, we construct a *Poincaré section* or *return map*  $R$

$$R(\Delta\Phi) : \Delta\Phi(t_k) \longrightarrow \Delta\Phi(t_{k+1}) = R(\Delta\Phi(t_k)) \quad (204)$$

$R$  maps the phase difference  $\Delta\Phi$  between the oscillators at times  $t_k$  when one oscillator fires, onto the phase difference  $\Delta\Phi$  when the *same oscillator* fires again at  $t_{k+1}$ .

One constructs  $R$  by first defining a *fire map*  $h$ :

$h$  maps the phase difference  $\Delta\Phi$  between the oscillators at times  $t_k$  when one oscillator fires, onto the phase difference  $\Delta\Phi$  when the *other* oscillator fires the next time.

Thus,  $h$  is given by:

$$h(\phi) = g(\epsilon + f(1 - \phi)) \quad (205)$$

Suppose oscillator  $A$  fires, and oscillator  $B$  is at phase  $\phi$ . Applying the fire map yields:

$$\{0, \phi\} \longrightarrow \{h(\phi), 0\} \quad (206)$$

Applying the fire map again yields:

$$\{h(\phi), 0\} \longrightarrow \{0, h(h(\phi))\} \quad (207)$$

Thus, return map  $R$  is given by iterated fire map  $h$ ,  $R = h \otimes h$ .

**Remark:**  $R$  is only defined on the smaller interval  $[\delta, h^{-1}(\delta)]$ , because neurons in  $[0, \delta]$  are *absorbed* or synchronized after one iteration of  $h$ ,  $h(\delta) = 1$ , and neurons in  $[h^{-1}(\delta), 1]$  are *absorbed* after two iterations of  $h$ .

**Lemma 1:**  $h'(\phi) < -1$  for  $\epsilon > 0$ .

**Proof:** With  $f'(1 - \phi) = 1/g'(f(1 - \phi))$ ,

$$h'(\phi) = g'(\epsilon + f(1 - \phi))f'(1 - \phi)(-1) \quad (208)$$

$$= -\frac{g'(\epsilon + f(1 - \phi))}{g'(f(1 - \phi))} \quad (209)$$

$$= -\frac{g'(\epsilon + u)}{g'(u)} \quad (210)$$

$$< -1 \quad (211)$$

The last relation holds true, because of  $f'' < 0 \longrightarrow g'' > 0$ .

**Lemma 2:**  $R'(\phi) > 1$  for  $\epsilon > 0$ .

**Proof:**  $R'(\phi) = h'(h(\phi))h'(\phi) > (-1)(-1) = 1$ .

**Lemma 3:**  $R$  has exactly one instable fixed point  $\phi^*$  in  $[\delta, h^{-1}(\delta)]$ . We consider  $\phi - h(\phi)$ :

**Proof:** At the definition boundaries

$$h(\delta) = 1 > \delta \longrightarrow \delta - h(\delta) < 0 \quad (212)$$

$$h(h^{-1}(\delta)) = \delta < h^{-1}(\delta) \longrightarrow h^{-1}(\delta) - h(h^{-1}(\delta)) > 0, \quad (213)$$

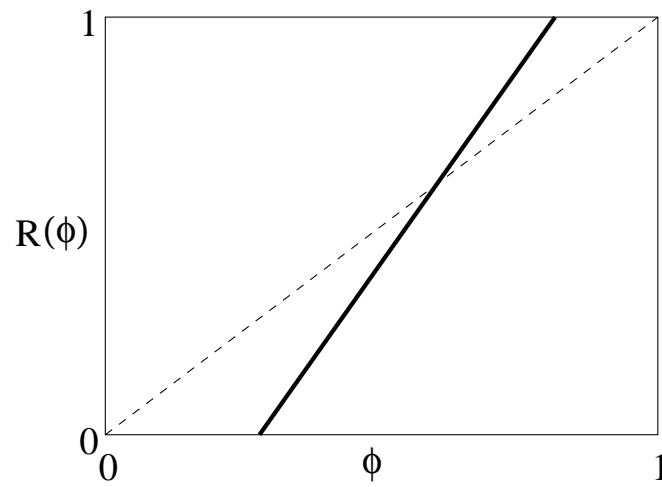
and  $d(\phi - h(\phi))/d\phi > 2$  (Lemma 1).

Therefore, it exists one  $\phi^*$  with  $h(\phi^*) = \phi^*$ , and thus  $R(\phi^*) = \phi^*$ .

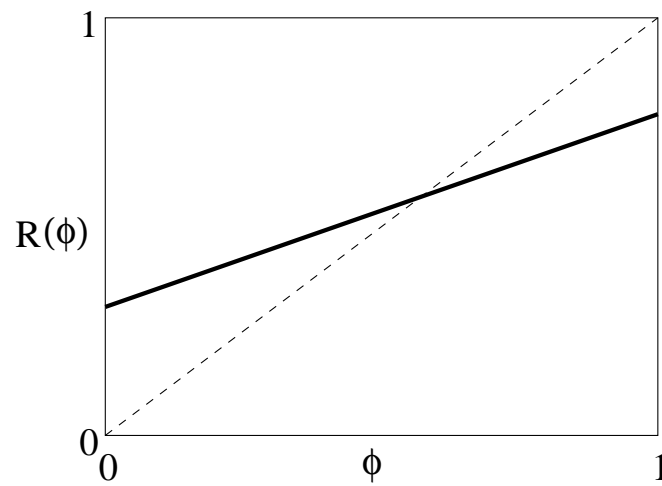
From Lemma 2 we have also  $R'(\phi^*) > 1$ , so  $\phi^*$  is a fixed point of  $R$ , and it is instable.

**Fact:** Synchronization for almost all initial conditions:

Excitatory couplings, one unstable fixed point:  $\rightarrow$  oscillators synchronize [?].

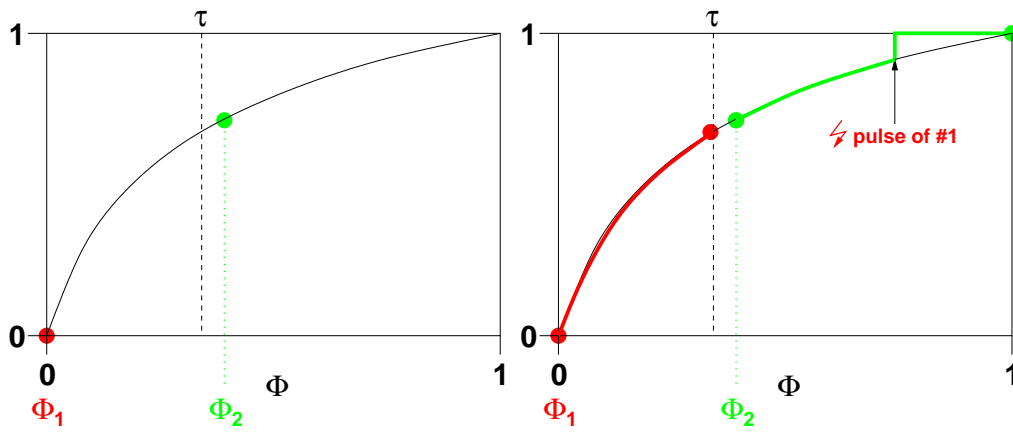


Inhibition? Just say  $\epsilon < 0$ ! One stable fixed point, oscillators fire in anti-phase.

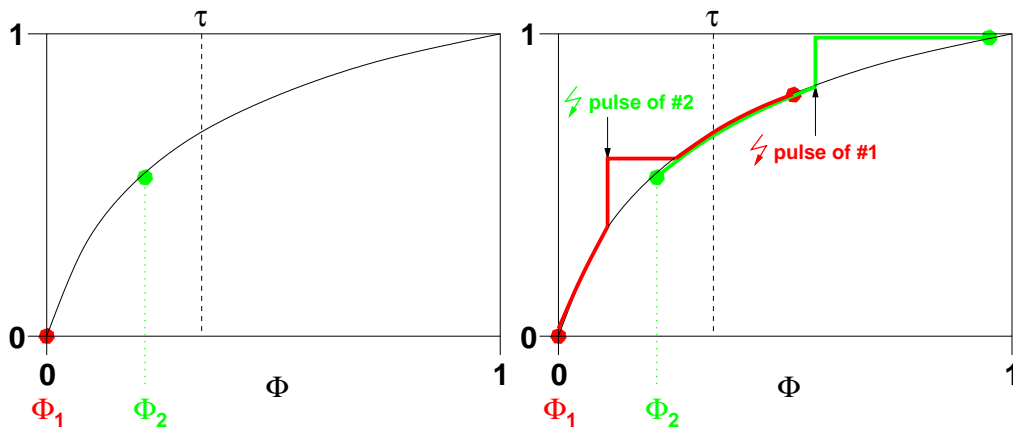


### 9.1.4 Considering delays $\tau$

$\Delta\Phi \geq \tau$ : For  $\Delta\Phi \geq \tau$ , only the first oscillator's pulse is on its way to oscillator two.



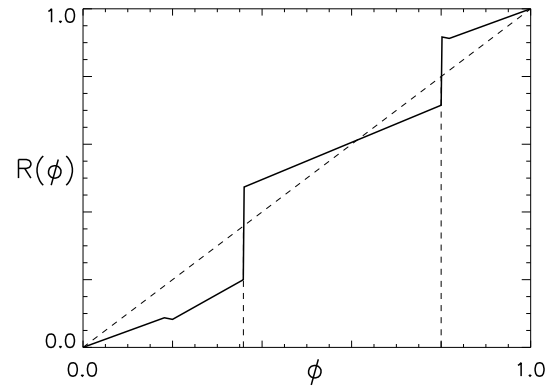
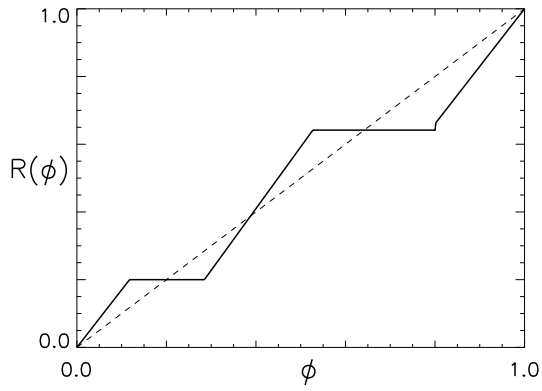
$\Delta\Phi < \tau$ : For  $\Delta\Phi < \tau$ , there is an additional pulse from oscillator two which will reach oscillator one after a time  $\tau - \Delta\Phi$ .



Tedious calculation with several case distinctions, but possible to derive expressions for  $R$  for all  $\tau \in [0, 0.5]$ ,  $\epsilon < 1$ .

**Excitation:** Out-of-phase synchronization (left)

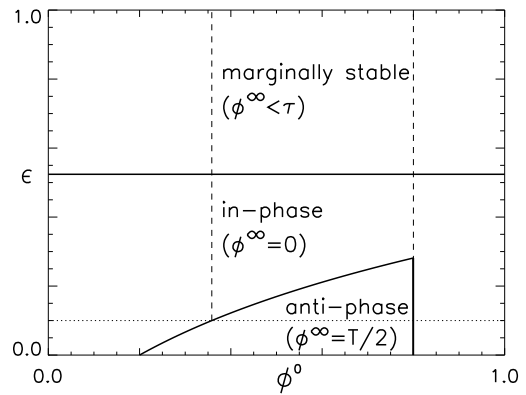
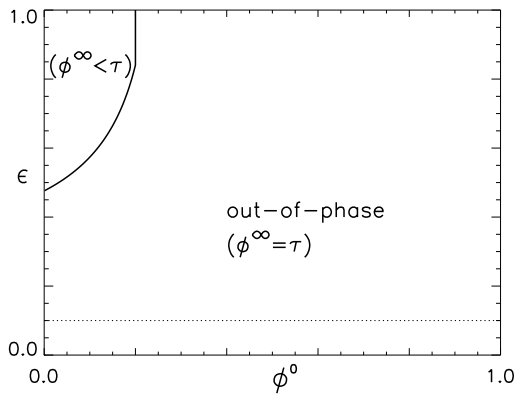
**Inhibition:** In- or out-of-phase synchronization (right)



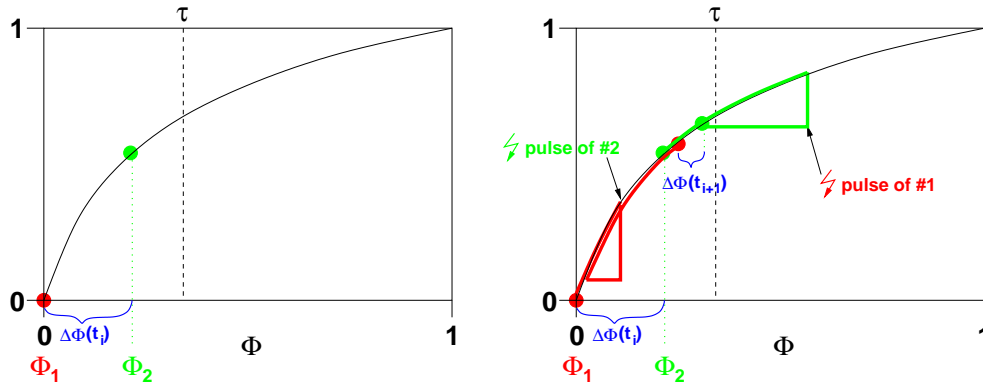
Phase diagrams for excitatory and inhibitory couplings,  $\epsilon < 1$ , and  $\tau < 0.5$ .

**Excitation:** (left)

**Inhibition:** (right)



## Why synchronization with inhibition and delay?:



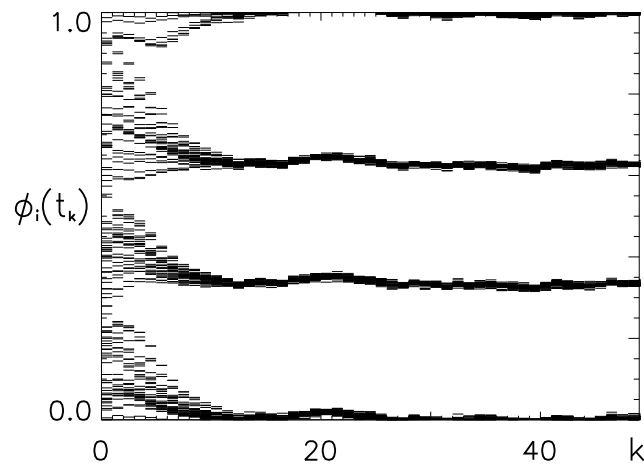
$f$  concave up:

- oscillator #2 with larger phase is more retarded than oscillator #1 with smaller phase
- phase difference decreases.

## Inhibition:

In-phase synchronization, stable phase clustering.

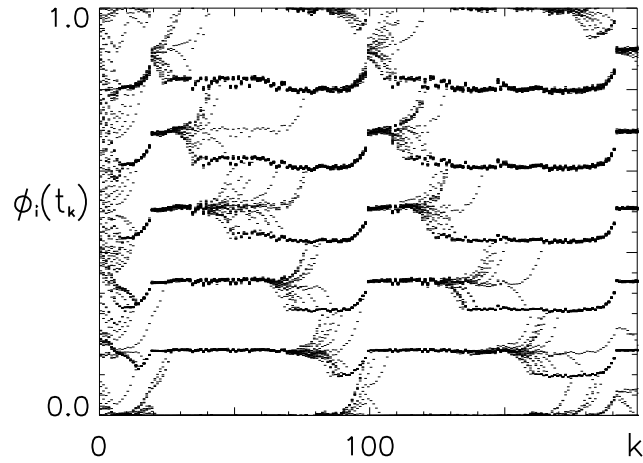
Okuda, *Physica D* **63** (1993); Golomb and Rinzel, *Physica D* **72** (1994).



## Excitation:

In-phase synchronization, unstable phase clustering.

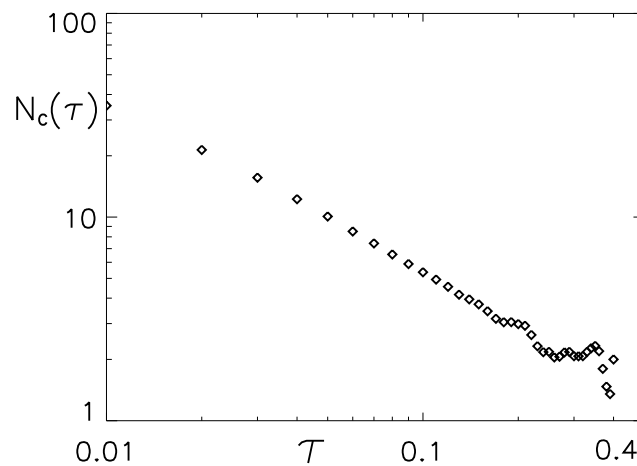
Ernst, Pawelzik, and Geisel, *Phys. Rev. Letters* (1995), and Timme, Wolf, and Geisel, *Phys. Rev. Letters* (2002).



### Multistability:

Inhibition and delay:

- With fixed delay  $\tau$ , from one up to  $N_c(\tau)$  clusters can be synchronized.
- $N_c(\tau)$  increases if the delay  $\tau$  decreases,  $N_c(\tau) = c/\tau$ .
- Behaviour can be understood in terms of phase diagrams for  $N = 2$  oscillators.



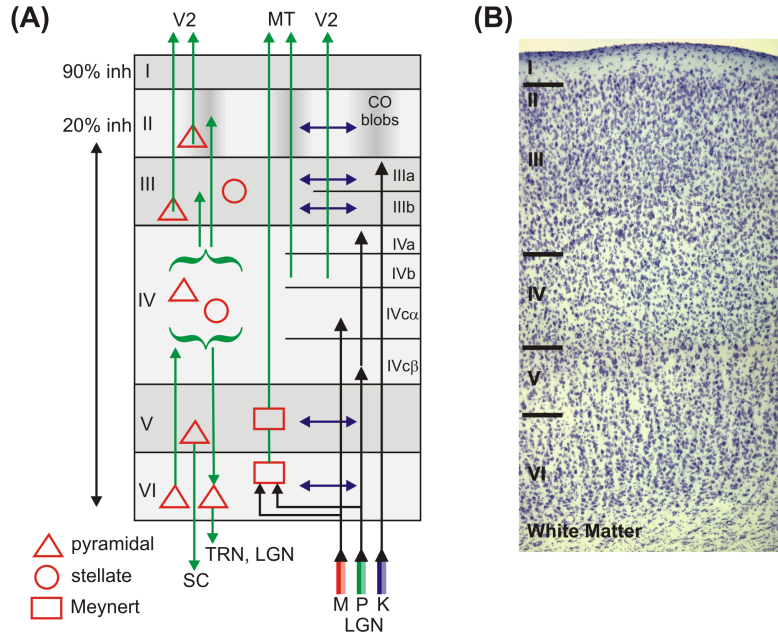
*Ernst, Pawelzik, and Geisel, PRL* **74** (1995);

*van Vreeswijk, PRE* **54** (1996);

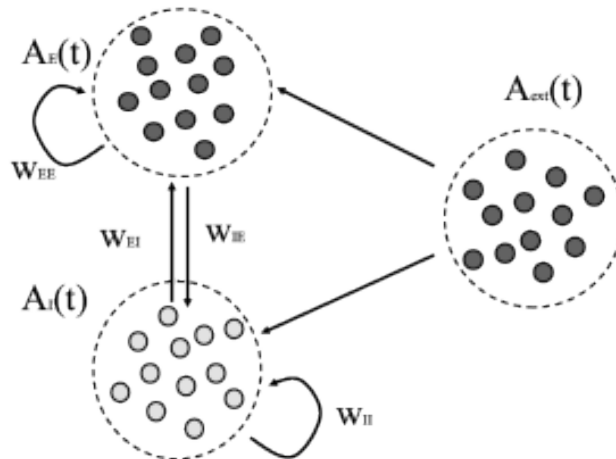
## 9.2 Stochastic drive and irregular firing

Synchronization in the previous example critically relies on a more or less constant external drive to the neurons, thus ensuring they fire at a particular frequency and have a well-defined phase.

In addition, we have neglected that typical cortical networks consist of mixed excitatory-inhibitory couplings, and focused on exclusively excitatory or exclusively inhibitory neurons only.



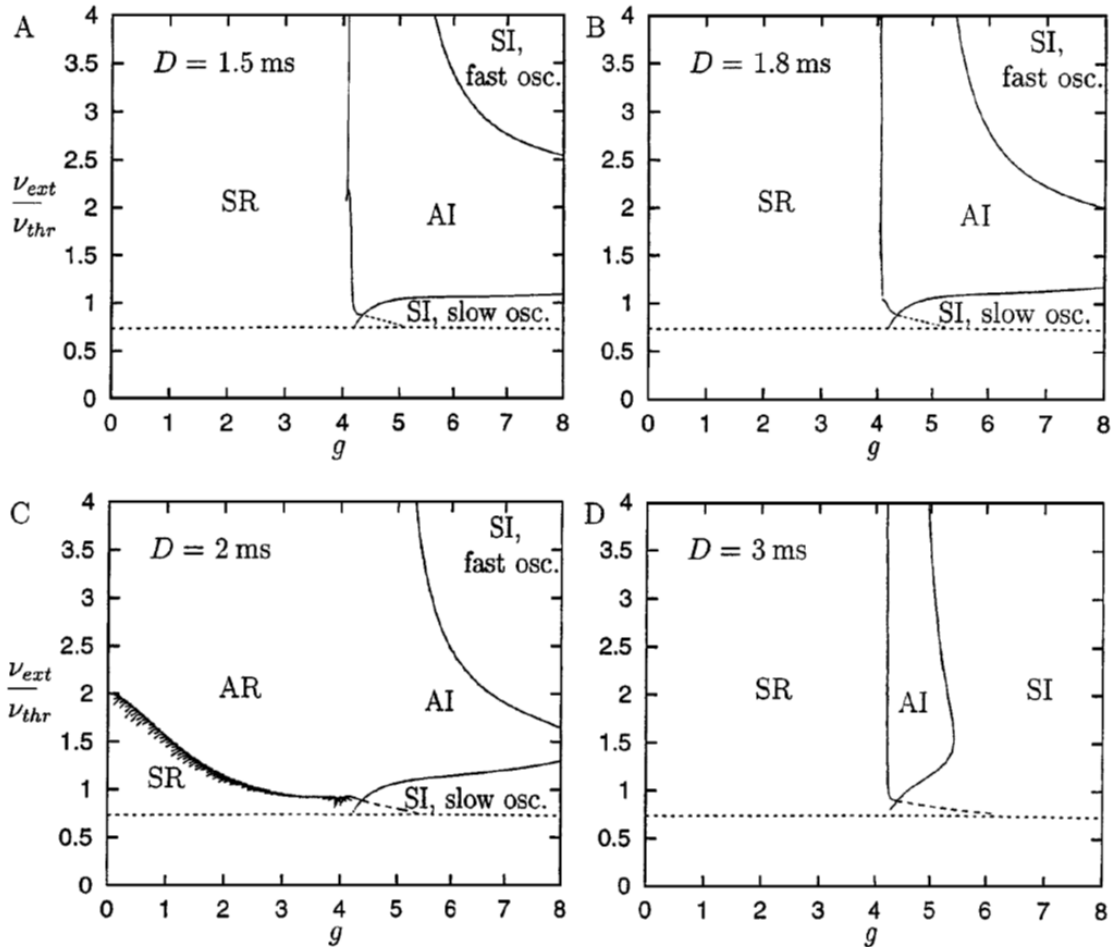
Subsequent modelling and theoretical studies addressed these issues, revealing an interesting and rich phenomenology of mixed networks subject to a stochastic drive. We here report the seminal work by Nicolas Brunel and coworkers [?, ?].



- Two populations: one excitatory with efficacy  $w_0$ , one inhibitory with efficacy  $-gw_0$  ( $N_E = 4N_I$ ).
- Mutual, recurrent couplings: sparse connectivity (=low probability for connection to exist).
- External stochastic drive, realized by Poissonian process.
- Network is perfectly balanced for  $g = 4$ .

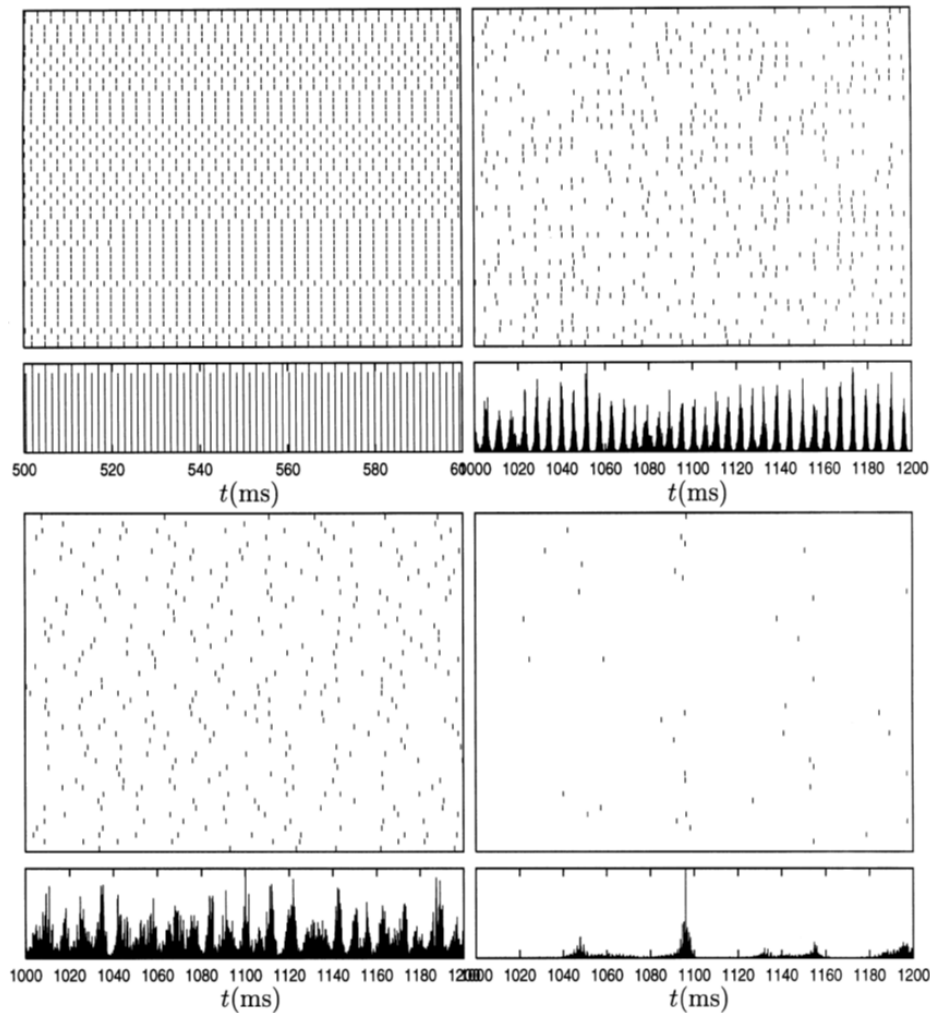
Reminder: Poisson distribution:  $P(k, \lambda T) = \frac{(\lambda T)^k}{k!} \exp(-\lambda T)$ , approximated by Bernoulli process with small  $p(\text{outcome} = \text{'Spike'})$ .

The network shows different regimes, or synchronization states, which can be visualized as regions in the phase space spanned by the coupling constant  $g$  and external input strength:



- Synchronous regular (SR) activity: excitation dominates, network synchronizes, splits into several subpopulations.

- Synchronous irregular (SI), fast or slow activity: In the fast regime, strong input excites the excitatory population, which in turn excites the inhibitory population, which in turn shuts down the excitatory population. In the slow regime, the network is occasionally put into an excited state, but shuts down quickly.
- Asynchronous irregular (AI) activity: Inhibition dominates in this regime, and leads to an asynchronous firing.



How can this network be analyzed? Brunel used the so-called *Fokker-Planck formalism*. Its idea is the following:

- Consider a very large number of neurons with membrane potentials  $V$ . At any moment in time, the state of the ensemble is characterized by a distribution or density  $\rho(V, t)$ .
- Construct a partial differential equation for the temporal change of the density  $\rho$ ,  $\partial\rho(V, t)/\partial t$  (continuity equation). It is composed of different terms:

- Due to the dynamics of the neurons themselves, and due to external factors such as an input current, there will be a *flux*  $J(V, t)$  of membrane potentials: for example, a strong input current would drive the neurons towards threshold (positive flux), while the leak conductance will drive the neurons back towards the resting potential  $V_{rest}$  (negative flux).

If the flux is constant across  $V$ , the density  $\rho$  will not change: there is an equal amount of neurons driven towards  $V$ , and away from  $V$ . In consequence, the temporal change in  $\rho$  will depend on the negative gradient in flux (i.e., when the flux is slowing down at  $V$ , there will be an increase in the number of neurons with that particular membrane voltage:

$$DRIFT = \frac{-\partial J(V, t)}{\partial V} \quad (214)$$

- Due to interactions with other neurons or different external sources, spikes will arrive at the neurons in our ensemble. If each spike makes the membrane potential increase or decrease by an amount  $\epsilon$ , there will be a decay in  $\rho$  at  $V - \epsilon$ , while there will be an equivalent increase at  $V$  (sinks and sources). We can easily accommodate multiple interaction strengths  $\epsilon_k$  with time-varying Poissonian(!) rates  $\lambda_k(t)$ :

$$INTERACTIONS = \sum_k \lambda_k(t)(\rho(V - \epsilon_k, t) - \rho(V, t)) \quad (215)$$

- Finally, we have to take care of boundary conditions: Neurons reaching  $V_{thresh}$  will fire and be reset to  $V_{rest}$ . These are again sinks and sources which are modeled by the following terms, where  $A(t)$  denotes the flux *at threshold*:

$$SPIKES = A(t)(\delta(V - V_{rest}) - \delta(V - V_{thresh})) \quad (216)$$

The drift term for typical neuron models at potential  $V$  is given by the change in membrane potential (provided by the DEQ for that particular neuron) times the actual density of states  $\rho(V, t)$  and takes the form

$$J(V, t) = \frac{1}{\tau}(f(V) + RI_{ext}(t))\rho(V, t) \quad , \quad (217)$$

which for the integrate-and-fire neuron is particularly simple:

$$J(V, t) = \frac{1}{\tau}(-(V - V_{rest}) + RI_{ext}(t))\rho(V, t) \quad , \quad (218)$$

The final expression for the continuity equation then reads:

$$\frac{\partial \rho(V, t)}{\partial t} = \frac{-\partial J(V, t)}{\partial V} + \sum_k \lambda_k(t) (\rho(V - \epsilon_k, t) - \rho(V, t)) \quad (219)$$

$$+ A(t) (\delta(V - V_{rest}) - \delta(V - V_{thresh})) \quad (220)$$

If the 'jumps'  $\epsilon_k$  due to arriving spikes are very small, this equation can be approximated by a Taylor series expansion in  $\epsilon_k$  in the input term  $I_{rec}(\epsilon) := \lambda(t) (\rho(V - \epsilon, t) - \rho(V, t))$ :

$$\begin{aligned} I_{rec}(\epsilon) &= I_{rec}(0) + \left. \frac{\partial I_{rec}}{\partial \epsilon} \right|_{\epsilon=0} \epsilon + \frac{1}{2} \left. \frac{\partial^2 I_{rec}}{\partial \epsilon^2} \right|_{\epsilon=0} \epsilon^2 + O(\epsilon^3) \\ &= 0 + \lambda \frac{\partial \rho}{\partial V}(-1) \epsilon + \frac{\lambda^2}{2} \frac{\partial^2 \rho}{\partial V^2}(-1)^2 \epsilon^2 + O(\epsilon^3) \end{aligned} \quad (221)$$

Neglecting terms of order  $\epsilon^3$  yields the Fokker-Planck equation:

$$\frac{\partial \rho(V, t)}{\partial t} = -\frac{\partial}{\partial V} \left\{ \left[ \frac{1}{\tau} (f(V) + RI_{ext}(t)) + \sum_k \lambda_k(t) \epsilon_k \right] \rho(V, t) \right\} \quad (222)$$

$$+ \frac{1}{2} \left[ \sum_k \lambda_k(t) \epsilon_k^2 \right] \frac{\partial^2}{\partial V^2} \rho(V, t) \quad (223)$$

$$+ A(t) (\delta(V - V_{rest}) - \delta(V - V_{thresh})) + O(\epsilon_k^3) \quad (224)$$

Here,  $\mu(t) = 1/\tau RI_{ext}(t) + \sum_k \lambda_k(t) \epsilon_k$  is the total drive, and  $\sigma^2(t) = \sum_k \lambda_k(t) \epsilon_k^2$  is the amount of diffusive noise.

In recurrent networks,  $\lambda_k(t)$  is substituted by total activity (flux at threshold)  $A(t)$ . This can also be done to have a Fokker-Planck equation for the network studied by Brunel. The dynamics of the network was then analyzed by first finding the steady-state solution  $\rho_0(V)$  and corresponding activity  $A_0$ . Then stability of that solution was probed by the perturbation ansatz  $A(t) = A_0 + \delta A \exp(\lambda t) \cos(\omega t)$ , looking for parameters which lead to a  $\lambda > 0$ , indicating instability of a solution with the particular frequency  $\omega$ .

# 10 POPULATION DYNAMICS

**Motivation:** Understand interplay between large networks in the brain.

**Problem:** High degree of complexity, cortex:  $10^7$  neurons, receiving input from  $10^4$  synapses.

Large networks of realistic neurons...

- (a) ...are **computationally expensive**,
- (b) ...their biological **details obscure** basic mechanisms,
- (c) ...the **size of their parameter space** makes any systematic approach impossible.

**Solution:**

Consider the activation dynamics of **neuronal populations**:

- (a) Subsume a multitude of single neurons into a **neural population** with appropriate dynamics.
- (b) Why making everything too complicated? **Minimalistic model** could suffice to explain observed phenomena to a large extent.
- (c) Population dynamics may be open to an analytic approach, which is often not feasible for networks of elements with differing characteristics.

## 10.1 Simplified Wilson-Cowan Dynamics

### 10.1.1 Heuristic derivation

We will present a heuristic derivation of the activation dynamics of a neural population first, and later focus on a more thorough derivation:

The ingredients of a minimal population dynamics:

- ...an activation variable  $A(t)$ , which is proportional to the mean firing rate of a population of **independent, uncoupled** units.
- ...a synaptic input  $I(t)$ , which consists of feedforward/feedback/recurrent contributions.
- ...a nonlinear gain function  $g$ , which associates a synaptic input with an activation.
- ...a time constant  $\tau$  which describes how fast a population reacts to a synaptic activation.

Without input ( $I = 0$ ), the activation of the population decays to zero:

$$\tau \frac{dA}{dt} = -A \quad (225)$$

With constant input, the activation should reach a constant level after some time (steady state), which is described by the gain function  $g$ :

$$\tau \frac{dA}{dt} = -A + g(I) \quad (226)$$

This is the simplest form of a population dynamics – this model can be extended to an arbitrary complexity. As a first extension, we consider a recurrent self-coupling with constant interaction strength  $w$ . Thus the input consists of an external drive  $I_{ext}$  and recurrent feedback  $I_{rec}$  which is given by  $I_{rec}(t) = wA(t)$ ,  $I(t) = I_{ext}(t) + I_{rec}(t)$ :

$$\tau \frac{dA}{dt} = -A + g(I) = -A + g(wA + I_{ext}) \quad (227)$$

This is a population dynamics that describes neuronal dynamics in terms of (output) activation/firing rate. There is a second, but less commonly used form, which describes neuronal dynamics in terms of (input) activation/total synaptic current  $I(t)$ :

$$\tau \frac{dI}{dt} = -I + I_{ext} + wg(I) \quad (228)$$

#### Remark:

The population dynamics described above is often termed **simplified** or **time coarse-grained Wilson-Cowan dynamics** [?]. We will learn more about the full, non-simplified Wilson-Cowan dynamics in the extended lecture.

### 10.1.2 Gain functions

Gain functions describe how a neuron or neuronal population responds to a constant input current. Typically, these functions are non-linear: neurons need a non-negative current (“threshold”)  $I_f$  to start firing, and their maximum firing rate is limited. In between these extremes, gain functions will increase with an (average or maximum) slope  $s$ . Therefore, the gain function will often have a **sigmoidal** shape and we will indicate this fact by using the symbol  $S$ .

#### Shapes of gain functions $S$ :

$S$  increases monotonically and (normally) saturates above a certain activation level. In a population dynamics,  $S$  summarizes characteristics and parameters of neuronal responses  $A$  to input currents  $I$  originating from...

- ...distributions of neuronal thresholds,
- ...distributions of synaptic weights,

- ...and other variations caused by noise.

**Unimodality:** of distributions:

$\Rightarrow S$  has the shape of a sigmoid function.

Examples for  $S$ :

(a) **Sigmoidal function** (logistic curve):

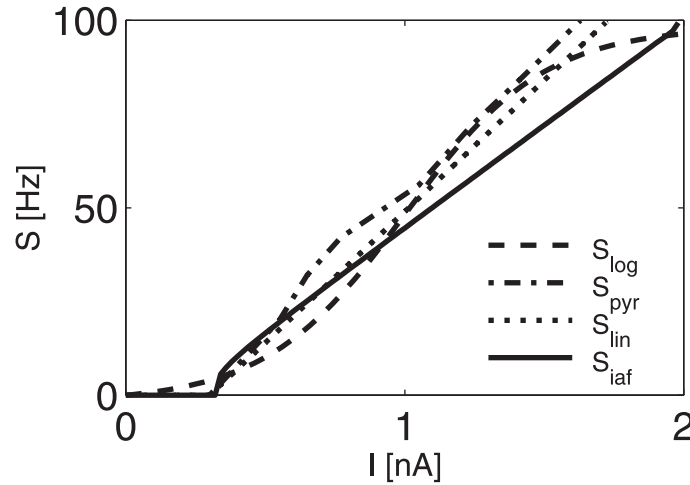
$$S_{\log}(I) := \frac{1}{1 + \exp[-2s(I - I_f)]} - \frac{1}{1 + \exp(2sI_f)} \quad (229)$$

(b) **Threshold-linear** function  $S_{\text{lin}}$

$$S_{\text{lin}}(I) := s \cdot (I - I_f) \quad \text{for } I > I_f, \text{ and } 0 \text{ otherwise.} \quad (230)$$

(c) **Integrate-and-fire** response curve  $S_{\text{iaf}}$

$$S_{\text{iaf}}(I) = \frac{-1}{\tau \log(1 - \frac{1}{I})} \quad \text{for } I > 1, \text{ and } 0 \text{ otherwise.} \quad (231)$$



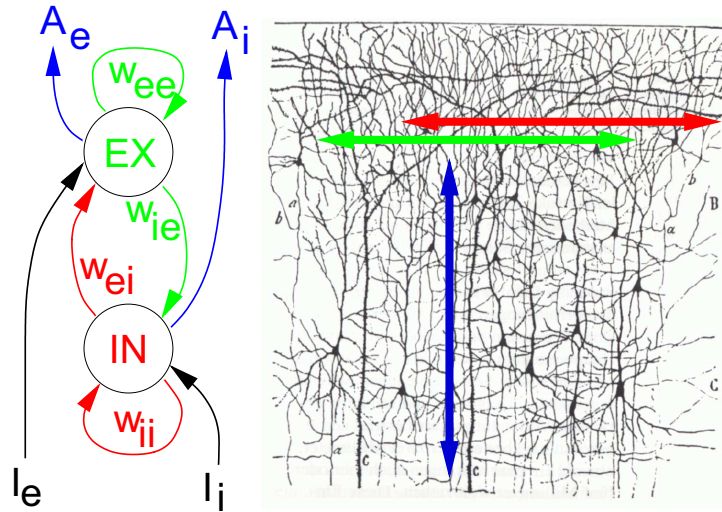
**Remark:**

The response curve typical for cortical neurons is surprisingly linear – also, most cortical neurons normally do not operate close to their saturation level. So a threshold-linear function is often a good choice in models/simulations.

### 10.1.3 Example: Cortical Microcircuits

**Aims:**

- (a) ...understand more about the dynamics of simple circuits with neural populations
- (b) ...apply the concepts to understand the behaviour of neocortical circuits



Properties of the system we want to describe and investigate:

- o Elementary microcircuit = **excitatory** and **inhibitory** population  
 $\Rightarrow$  two **differential equations** for activities  $A_e, A_i$ .
- o Coupling: each population coupled to each other (mutual coupling)  
 $\Rightarrow$  four **coupling coefficients**  $w_{ee}, w_{ie}, w_{ei}, w_{ii}$ .
- o Input-Output: gain functions for the response  $S(I)$  of the neurons to input  $I$ .
- o (peripheral) stimulus: apply **external input**  $I(t)$

Population dynamics :

$$\tau_e \frac{dA_e}{dt} = -A_e + S_e(w_{ee}A_e - w_{ei}A_i + I_e) \quad (232)$$

$$\tau_i \frac{dA_i}{dt} = -A_i + S_i(w_{ie}A_e - w_{ii}A_i + I_i). \quad (233)$$

Before we are discussing the dynamics of this cortical microcircuit, we will make an excursion for the mathematically interested reader and introduce the full Wilson-Cowan dynamics:

## 10.2 Wilson-Cowan equations

### 10.2.1 Full dynamics

The simplified population dynamics described in the previous sections was only derived heuristically. Wilson and Cowan [?] properly derived delayed integral equations for  $A_e(t)$  and

$A_i(t)$ .

In the original work,  $A$  is the **fraction** of neurons being active at time  $t$  (and therefore of course proportional to **total activity** or **mean firing rate** of a neuronal population). The basic form of the equations is given by the following considerations:

- (a) Neurons are refractory: At an arbitrary time  $t$ , there are always some neurons which just have fired and are in a **refractory (non-excitabile)** state for a time  $r$ . The remaining neurons (fraction  $F_{exc}$  of all neurons) are in an excitable state.
- (b) The number of neurons which will become active in an excitable state depends on the amount of excitation  $E(t)$  they receive. This dependency can be described by a logistic function  $S$  which starts from 0 (no neuron becomes active) for low  $E$  and saturates at 1 for high  $E$  (all neurons will become active).
- (c) It will take a time  $\tau$  for a neuron to become active (e.g. initiation of an action potential).

In consequence, the fraction of neurons  $A(t + \tau)$  active at  $t + \tau$  is given by the product  $F_{exc}(t) \cdot S(E(t))$ .

Now we have to find expressions for the terms  $F_{exc}(t)$  and  $E(t)$ :

- (a) Fraction of excitable neurons is 1 minus total fraction of neurons that have been active between  $t - \tau$  and  $t$  and are thus in refractory period:

$$F_{exc}(t) := 1 - \int_{t-\tau}^t A(t') dt' \quad (234)$$

- (b) Excitation level  $E$  is given by the total synaptic input  $I_{tot}(t)$  convolved with a temporal kernel  $\alpha$  subsuming the temporal dynamics of synapses and synaptic transmission:

$$E(t) := \int_{-\infty}^t \alpha(t - t') I_{tot}(t') dt' \quad (235)$$

Typical choices for  $\alpha$  include:

- Simplest choice, instantaneous transmission:  $\alpha(t) \approx \delta(-t)$ .
- Wilson and Cowan:  $\alpha(t) \approx \exp(-t)$ .
- Most commonly used, has smooth increase:  $\alpha(t) \approx t \exp(-\alpha_0 t)$ .
- Also found:  $\alpha(t) \approx t \exp(-\alpha_0 t^2)$ .

If we put these expressions together, and assume that we focus on an excitatory and inhibitory

population that are mutually coupled, we arrive at:

$$A_e(t + \tau_e) = \left[ 1 - \int_{t-r_e}^t A_e(t') dt' \right] \cdot S_e \left( \int_{-\infty}^t \alpha(t-t') [w_{ee} A_e(t') - w_{ei} A_i(t') + I_e(t')] dt' \right) \quad (236)$$

$$A_i(t + \tau_i) = \left[ 1 - \int_{t-r_i}^t A_i(t') dt' \right] \cdot S_i \left( \int_{-\infty}^t \alpha(t-t') [w_{ie} A_e(t') - w_{ii} A_i(t') + I_i(t')] dt' \right) \quad (237)$$

### 10.2.2 Time coarse-grained dynamics

Problem: Integral equations equation (236) and equation (237) are still too unhandy. But it is possible to apply a **time-coarse graining** technique by performing a **time average** over  $A$ :

$$\bar{A}(t) := \frac{1}{r} \int_{t-r}^t A(t') dt' \quad (238)$$

If we assume that function  $\alpha(t)$  is close to unity for  $0 \leq t \leq r$  and then drops off fairly quickly to 0, we can also write:

$$k\bar{A}(t) := \int_{-\infty}^t \alpha(t-t') A(t') dt' \quad (239)$$

Furthermore, we can express the left hand sides of Eqs.(236-237) by a Taylor expansion in terms of the coarse-grained variable  $\bar{A}$ :

$$A(t + \tau) \approx \bar{A}(t) + \tau \left. \frac{d\bar{A}}{dt} \right|_t + O(\tau^2) \quad (240)$$

Taking together Eqs.(238-240), we finally arrive at:

$$\tau_e \frac{d\bar{A}_e}{dt} = -\bar{A}_e + (1 - r_e \bar{A}_e) S_e(kw_{ee} \bar{A}_e - kw_{ei} \bar{A}_i + kI_e) \quad (241)$$

$$\tau_i \frac{d\bar{A}_i}{dt} = -\bar{A}_i + (1 - r_i \bar{A}_i) S_i(kw_{ie} \bar{A}_e - kw_{ii} \bar{A}_i + kI_i) \quad (242)$$

Without loss of generality, one can rescale the gain functions in Eqs.(241-242) such that the time coarse-graining constant  $k = 1$ . Also, we normally use  $A_e$  ( $A_i$ ) instead of the terms  $\bar{A}_e$  ( $\bar{A}_i$ ).

## 10.3 Analyzing simple population models

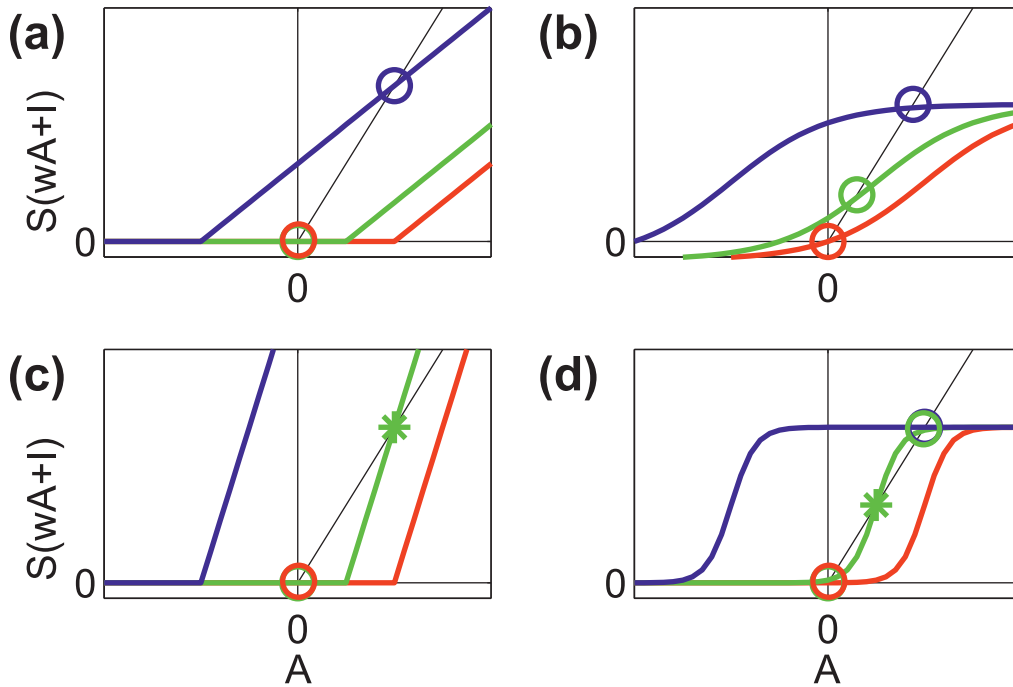
### 10.3.1 Population dynamics: one subpopulation

We first consider just one population:

$$\tau \frac{dA}{dt} = -A + S(wA + I) \quad . \quad (243)$$

**Fixed points:**

$dA/dt = 0$ . Solve  $A_0 = S(wA_0 + I)$  for  $A_0$ :



(a,c) threshold-linear gain function  $S_{lin}$ , (b,d) sigmoidal gain function  $S_{log}$  in dependence on input current (red=low, green=average, blue=high current).

Two different regimes depending on gain constant  $s$

(a) **Weak coupling** regime  $sw < 1$ :

With both  $S = S_{lin}$  and  $S = S_{log}$ , existence of one stable fixed point  $A_0$  whose absolute value increases monotonically with increasing Input  $I$ .

With  $S = S_{log}$ ,  $A_0$  saturates at higher values of  $I$ .

(b) **Strong coupling** regime  $sw \geq 1$ :

With  $S = S_{lin}$ , either one stable fixed point at  $A_0 = 0$ , or the activity increases beyond all limits.

With  $S = S_{\log}$ , and depending on  $I$ , existence of either one stable fixed point near 0, or one stable fixed point near maximum activity, or coexistence of both fixed points.

### Hysteresis:

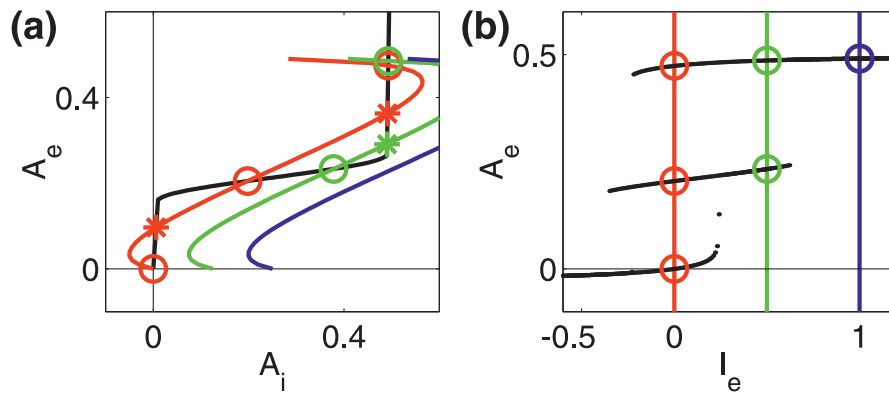
With intermediate  $I$ , one of the two fixed points is chosen, depending on the initial or previous activation level  $A$ .

### Functional consequence of hysteresis:

Hysteresis can implement a form of **short-term memory** – brief pulses of external input can excite a column, which remains activated after the input has decayed, due to the dynamics of the internal couplings ('one-bit memory').

### 10.3.2 Population dynamics: two coupled subpopulations

Eqs.(232-233) yield two nullclines intersecting at the fixed points of the activation dynamics. Stability assessed by linearization of Eqs.(232-233) and solving the characteristic equation.



(a) nullclines of the two population system, (b) fixed points in dependence on input current (red=low, green=average, blue=high current).

Choice of gain function  $S$  makes huge differences:

- $S_{\text{lin}}$ : no hysteresis, either stable fixed point at  $A_e \geq 0$  and  $A_i \geq 0$ , or activation diverges because interaction is too strong.
- $S_{\log}$ : possible multiple hysteresis phenomena. With increasing constant input, either **one**, **two**, or **three** stable fixed points coexist.

**Limit cycles:**

For some range of parameters, model can exhibit (damped) oscillations. These solutions of the DEQs correspond to the existence of **limit cycles** in phase space – requiring a single unstable fixed point and sufficiently high input.

**Relation of limit cycles to physiological experiments:**

Rhythmicity in certain frequency bands of EEG recordings, or other oscillatory phenomena in brain activity.

### 10.3.3 Limitations of the simplified dynamics

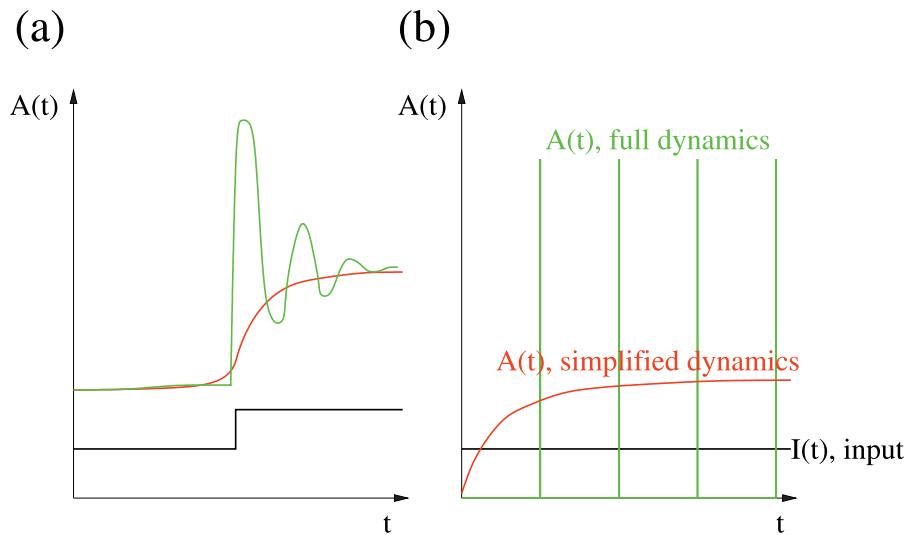
Disadvantage of simplified population dynamics considered so far:

- (a) ...response is slow (determined by time constant  $\tau$ !)
- (b) ...activity only reflects mean rate (correlations between spikes averaged out, no synchronization!)

#### Example:

Population of integrate-and-fire neurons with

- (a) ...stochastic input and without coupling, and
- (b) ...with constant input and global coupling:



## 10.4 Spatially extended systems

A “microcircuit” as investigated above describes the dynamics of a small cortical “column”: a few thousand neurons at one cortical location with largely overlapping receptive fields. Still, such a system already displays nice phenomena: fixed points, multistability/hysteresis, memory, oscillations... However, brain dynamics gets **really** interesting if we consider spatially extended neuronal populations:

**Activation  $A$  will now be a function of time  $t$  and unit index  $i$ , or space  $\mathbf{r}$ :**

$$A(t) \rightarrow A_i(t) \quad (\text{discrete systems}) \quad (244)$$

$$A(t) \rightarrow A(\mathbf{r}, t) \quad (\text{continuous systems}) \quad (245)$$

Recurrent synaptic input now depends also on activities **in all other columns**. We neglect delays of synaptic transmission from  $\mathbf{r}$  to  $\mathbf{r}'$ . In consequence, we have to replace terms like  $wA$  with:

- (a) ...**summation** of products of activities with coupling weights  $w_{ij}$  from unit  $j$  to unit  $i$  (discrete systems):

$$wA \rightarrow \sum_j w_{ij} A_j \quad (246)$$

...or by

- (b) ...**integration** of the activities with the corresponding coupling functions  $W(\mathbf{r}, \mathbf{r}')$  specifying the interactions from the unit at location  $\mathbf{r}'$  to the unit at location  $\mathbf{r}$ :

$$wA \rightarrow \int_{CTX} W(\mathbf{r}, \mathbf{r}') A(\mathbf{r}') d\mathbf{r}' \quad (247)$$

In the following, we will focus on continuous systems since we all like to solve integrals (believe me, you **won't** like to solve the corresponding sums in discrete systems!)...

Often we can make the assumption that populations and couplings are **spatially homogeneous**, and that the corresponding couplings are **translationally invariant**. Thus we can write  $W(\mathbf{r}, \mathbf{r}') = W(\mathbf{r} - \mathbf{r}')$ , the coupling function is now a **kernel**, and recurrent synaptic integration of inputs becomes a **convolution** with the neural activities:

$$I_{rec}(\mathbf{r}, t) := \int_{CTX} W(\mathbf{r}, \mathbf{r}') A(\mathbf{r}', t) d\mathbf{r}' = \int_{CTX} W(\mathbf{r} - \mathbf{r}') A(\mathbf{r}', t) d\mathbf{r}' =: [W * A](\mathbf{r}, t) \quad (248)$$

where the last expression is a shorthand notation for the convolution operation.

If we want to model a system with excitatory and inhibitory spatially extended populations, we have to introduce the four kernels  $W_{ee}(\mathbf{r} - \mathbf{r}')$ ,  $W_{ei}(\mathbf{r} - \mathbf{r}')$ ,  $W_{ie}(\mathbf{r} - \mathbf{r}')$ , and  $W_{ii}(\mathbf{r} - \mathbf{r}')$ . For convenience, we assume that coupling functions are *normalized*:

$$\int_{CTX} d\mathbf{r}' W_{xx}(\mathbf{r} - \mathbf{r}') = w_{xx} \quad (249)$$

For describing the temporal dynamics of spatially extended populations, we have to use **partial differential equations** instead of **ordinary differential equations**: Since  $A$  depends on both space and time, temporal changes are described by the derivative **with respect to time  $t$** :

$$\tau_e \frac{\partial A_e(\mathbf{r}, t)}{\partial t} = -A_e(\mathbf{r}, t) + (k_e - r_e A_e(\mathbf{r}, t)) \quad (250)$$

$$\times S_e([W_{ee} * A_e](\mathbf{r}, t) - [W_{ei} * A_i](\mathbf{r}, t) + I_e(\mathbf{r}, t)) \quad (251)$$

$$\tau_i \frac{\partial A_i(\mathbf{r}, t)}{\partial t} = -A_i(\mathbf{r}, t) + (k_i - r_i A_i(\mathbf{r}, t)) \quad (252)$$

$$\times S_i([W_{ie} * A_e](\mathbf{r}, t) - [W_{ii} * A_i](\mathbf{r}, t) + I_i(\mathbf{r}, t)) \quad (253)$$

These **spatially extended Wilson-Cowan equations** [?] can also be time-coarse-grained and then become (with renormalization  $kS \rightarrow S$ ):

$$\tau_e \frac{\partial A_e(\mathbf{r}, t)}{\partial t} = -A_e(\mathbf{r}, t) + S_e([W_{ee} * A_e](\mathbf{r}, t) - [W_{ei} * A_i](\mathbf{r}, t) + I_e(\mathbf{r}, t)) \quad (254)$$

$$\tau_i \frac{\partial A_i(\mathbf{r}, t)}{\partial t} = -A_i(\mathbf{r}, t) + S_i([W_{ie} * A_e](\mathbf{r}, t) - [W_{ii} * A_i](\mathbf{r}, t) + I_i(\mathbf{r}, t)) \quad (255)$$

Before we discuss specific spatially extended systems for modelling and understanding important dynamic phenomena in visual cortex, let us discuss a further simplification of the simplified Wilson-Cowan equations:

- (a) Axons originating from one population should contact excitatory and inhibitory neurons in a given distance with equal relative probability

$$\Rightarrow W_{ee} \propto W_{ie} \text{ and } W_{ei} \propto W_{ii}.$$

- (b) Population parameters and gain functions are identical for inhibitory and excitatory populations

$$\Rightarrow \tau_e = \tau_i \text{ and } S_e = S_i.$$

For this special case, Eqs.(254-255) are redundant and can be replaced by a single partial differential equation

$$\tau \frac{\partial A(\mathbf{r}, t)}{\partial t} = -A(\mathbf{r}, t) + S([W_e * A](\mathbf{r}, t) - [W_i * A](\mathbf{r}, t) + I(\mathbf{r}, t)) \quad (256)$$

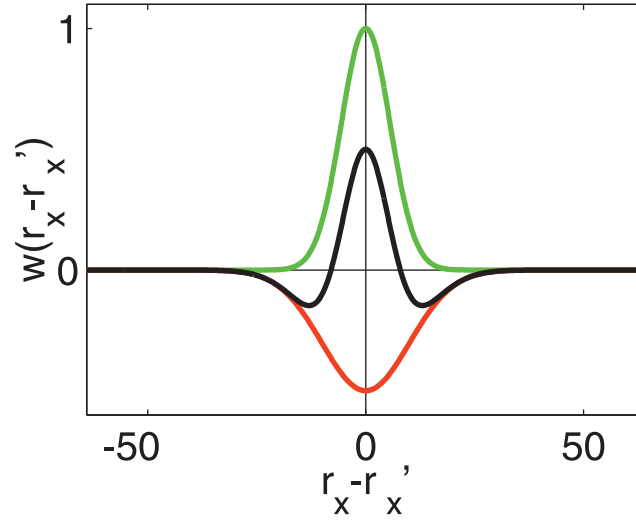
### 10.4.1 Cortical dynamics

We now apply this formalism to describe population dynamics in the visual cortex. Plausible assumptions are:

- (a) Threshold-linear gain function  $S = S_{\text{lin}}$
- (b) Gaussian coupling functions  $W_e$  and  $W_i$ , with inhibitory couplings having a longer range  $\sigma_i$  than excitatory couplings  $\sigma_e$ :

$$W_e(\mathbf{r} - \mathbf{r}') = \frac{w_e}{(2\pi)^{d/2} \sigma_e^d} \exp\left(-\frac{\|\mathbf{r} - \mathbf{r}'\|^2}{2\sigma_e^2}\right) \quad (257)$$

$$W_i(\mathbf{r} - \mathbf{r}') = \frac{w_i}{(2\pi)^{d/2} \sigma_i^d} \exp\left(-\frac{\|\mathbf{r} - \mathbf{r}'\|^2}{2\sigma_i^2}\right) \quad (258)$$



#### Parameters:

- $s$ : gain,  $I_f$ : firing threshold,
- $w_e$ : excitatory coupling strength,
- $w_i$ : inhibitory coupling strength
- $\sigma_e$ : excitatory coupling length scale,
- $\sigma_i$ : inhibitory coupling length scale,
- $d$ : dimensionality of neuronal tissue.

### 10.4.2 Constant input

We first consider a spatially and temporally constant input. Also, we consider a one-dimensional population “chain” of length  $l_x$  ( $d = 1$ ) with periodic boundary conditions:

$$I(\mathbf{r}, t) = I_0 = \text{const.} > I_f . \quad (259)$$

Under these conditions, there exists a single homogeneous fixed point  $A_0 > 0$ , for  $w_i > w_e - 1/s$  (for arbitrary coupling functions):

$$A_0(\mathbf{r}, t) = A_0 = \frac{s(I_0 - I_f)}{1 - s(w_e - w_i)} .$$

Now we consider the stability of this fixed point. We first linearize around  $A_0$ ,  $A(\mathbf{r}, t) = A_0 + \delta A(\mathbf{r}, t)$ , obtaining:

$$\tau \frac{\partial \delta A(\mathbf{r}, t)}{\partial t} = -\delta A(\mathbf{r}, t) + s([W_e * \delta A](\mathbf{r}, t) - [W_i * \delta A](\mathbf{r}, t)) . \quad (260)$$

This partial DEQ still looks bad; the activity at location  $\mathbf{r}$  is still dependent on the activities at other locations  $\mathbf{r}'$  (via the coupling kernels). But there’s a **trick**! We can apply a Fourier transformation to the activation pattern  $\delta A(\mathbf{r})$ , thus obtaining the spectral decomposition  $\delta \tilde{A}(\mathbf{k})$  in dependence on the **wave vector**  $\mathbf{k}$ . According to the convolution theorem, the integral  $\int W(\mathbf{r} - \mathbf{r}') A(\mathbf{r}') d\mathbf{r}'$  now becomes a multiplication  $2\pi \tilde{W}(\mathbf{k}) \tilde{A}(\mathbf{k})$ :

$$\tau \frac{\partial \delta \tilde{A}(\mathbf{k}, t)}{\partial t} = -\delta \tilde{A}(\mathbf{k}, t) + 2\pi s(\tilde{W}_e(\mathbf{k}) \delta \tilde{A}(\mathbf{k}, t) - \tilde{W}_i(\mathbf{k}) \delta \tilde{A}(\mathbf{k}, t)) . \quad (261)$$

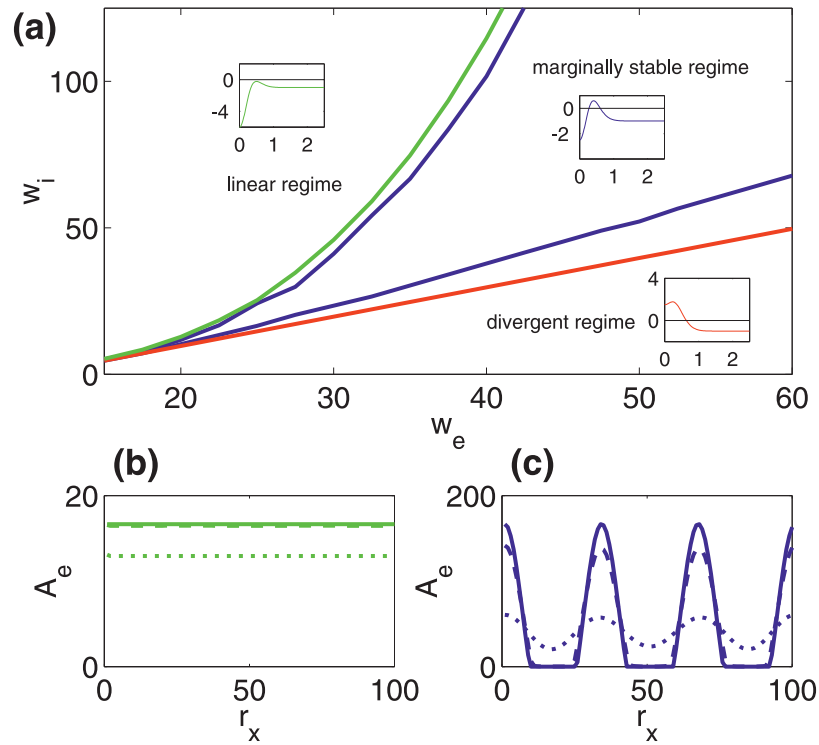
In frequency space, the differential equations are now **decoupled**, and they have the form

$$\frac{\partial \delta \tilde{A}(\mathbf{k}, t)}{\partial t} = \lambda(\mathbf{k}) \delta \tilde{A}(\mathbf{k}, t) . \quad (262)$$

The exponents  $\lambda$  give us a **spectrum of eigenvalues (modes)** in dependence of the spatial mode  $\mathbf{k}$  of the perturbation. If for one specific  $\mathbf{k}$  the exponent is positive, the homogeneous fixed point is not stable and the corresponding mode will start to increase exponentially.

These conditions can be visualized in a **phase diagram** in dependence on certain **system or control parameters**. A phase diagram displays regions in which the dynamics of a system is qualitatively different. Here, our control parameters are the total recurrent coupling strengths:

$\implies$  We distinguish three different phases or **regimes**



- (a) **strong inhibitory coupling**: homogeneous fixed point is stable, every perturbation decays exponentially (termed **linear** regime).
- (b) **medium inhibitory coupling**: fixed point becomes unstable and small perturbations lead to exponentially increasing activity. Due to nonlinearity in  $S$ , activity converges into inhomogeneous stable state. The particular form of this state depends on initial conditions/perturbations (termed **marginally stable** regime).
- (c) **weak inhibitory coupling**: nonlinearity does not suffice to limit diverging activity (termed **diverging** regime).

### 10.4.3 Spatially inhomogeneous input

Example for **inhomogeneous** input: Consider again neuronal chain of length  $l_x$ , periodic boundary conditions, unimodal input distribution with one maximum at  $r_x' = l_x/2$ , e.g.:

$$I = I_0((1 - \epsilon) + \epsilon \exp(-\|r_x - r_x'\|/2\sigma_I)) \quad (263)$$

$$\text{or} \quad I = I_0(1 + \epsilon \cos(2\pi(r_x - r_x')/l_x)) \quad (264)$$

Which neural response emerges when applying this input?

- (a) **Linear** regime: afferent input  $\gg$  lateral feedback

⇒ spatially inhom. input leads to a similar activity distribution.

(b) **Marginally stable** regime: lateral feedback  $\gg$  afferent input

⇒ blobs emerge, centered at positions of maximal input.

**Mechanism:** for last case (**Pattern Formation**):

Excitatory interactions prevail on short distances

⇒ **perturbation enhanced** by excitation

⇒ surround activity **suppressed** by inhibition

⇒ localized **activation blob** centered around  $l_x/2$ .

Assume afferent input  $I$  is suprathreshold everywhere – do one or more blobs emerge? Rule of thumb:

(a)  $\sigma_i < l_x$

⇒ other blobs appear in a specific distance determined by the **length scale**  $\sigma_i$  of the **inhibitory** interaction.

(b)  $\sigma_i > l_x$

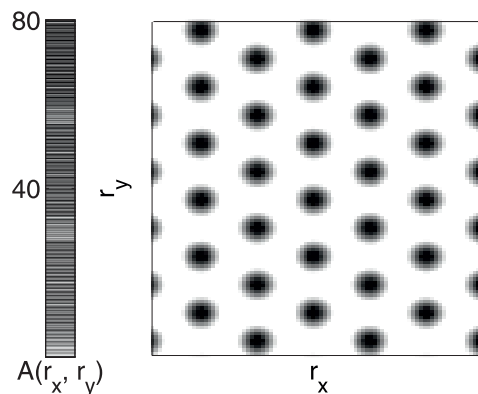
⇒ **Winner-takes-all network**, single blob appears where feedback and external input is strongest.

#### 10.4.4 Two-dimensional activation dynamics

Going from one to  $N$  dimensions, we typically find the same qualitative behaviour – however, parameters at which different regimes in pattern formation emerge might be different (calculate again!).

**Example:**

In a 2D-cortex, the activation clusters typically arrange in a hexagonal pattern:



Two other interesting dynamical states with propagating waves or blobs of velocity  $\Omega_b$  [?, ?, ?]:

(a) Moving periodic stimulus with velocity  $\Omega_s$ , e.g.

$$I = I_0(1 + \epsilon \cos(2\pi(0.5 + \Omega_s t + r_x/l_x))) \quad (265)$$

$\Rightarrow$  drags the blobs into the direction of movement.

Depending on time scales and stimulus amplitudes,

- ...blobs may follow perfectly ( $\Omega_b = \Omega_s$ ),
- ...or may be slower ( $\Omega_b < \Omega_s$ ), skipping stimulus cycles.

(b) Small asymmetry in input leads to self-propagating wave

$\Rightarrow$  necessary conditions are large inhibitory input [?], and  $\epsilon \ll 1$  to prevent pinning at position of maximal input.

#### 10.4.5 Example: Orientation tuning

Neurons in visual cortex are tuned to the orientation of a stimulus within their receptive field (**orientation preference**). The properties of orientation tuning can nicely be studied with population dynamics:

The **model of Hubel and Wiesel** [?] explains orientation preference by linear superposition of ON-OFF responses, like in a LNP model with Gabor-shaped spatial kernel. This provides a cosine-tuned input to a neural **hypercolumn**, but it also creates a problem: The **Iceberg-effect**. In the following, we will discuss this problem in more detail...

##### Hubel and Wiesel meet population dynamics:

We consider a model hypercolumn (1D) of neurons with preferred orientations  $\Phi$

$$I(\Phi_S) = I_0(1 + \epsilon \cos(2(\Phi - \Phi_S))) \quad (266)$$

$$\tau \frac{\partial A(\Phi, t)}{\partial t} = -A(\Phi, t) + S(I(\Phi_S, \Phi)) \quad (267)$$

With no recurrent feedback, just afferent input: steady-state solution of the DEQ is simply:

$$A(\Phi) = S(I(\Phi_S, \Phi)) \quad \text{for } t \rightarrow \infty \quad (268)$$

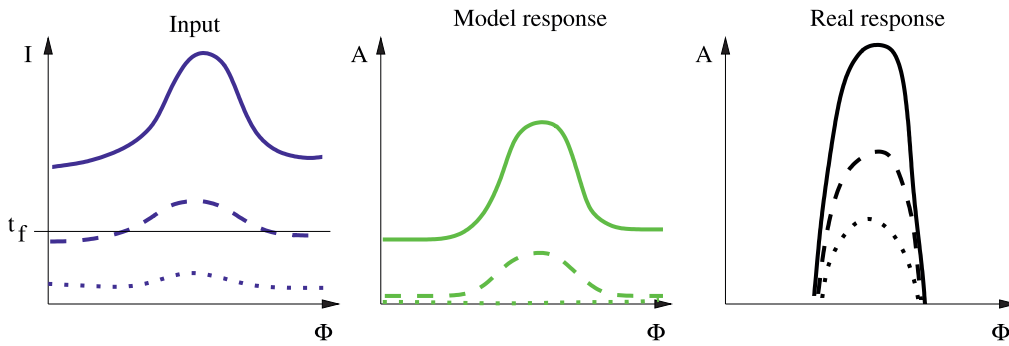
In consequence, resulting tuning curves are **too broad** and **not contrast-invariant**:

What do we do?

##### Solution:

Take recurrent couplings in V1 into account. **First idea**:: add inhibitory interactions only.

**Result**:: tuning curves are sharpened, but still not contrast-invariant.

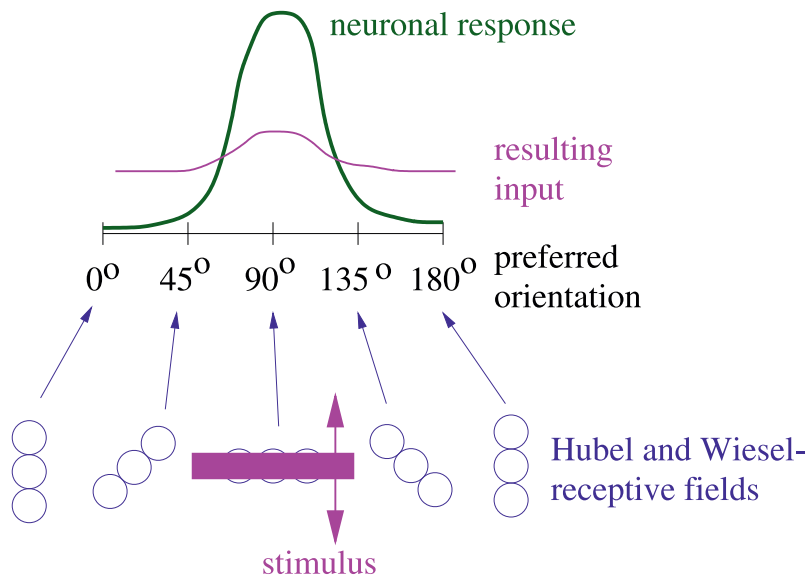


**Second idea::** use excitatory **and** inhibitory couplings, forming a **Mexican hat**. Model is same as in Eq.(256), only  $r$  substituted by  $\Phi$ :

$$\tau \frac{\partial A(\Phi, t)}{\partial t} = -A(\Phi, t) + S([W_e * A](\Phi, t) - [W_i * A](\Phi, t) + I(\Phi, t)) \quad (269)$$

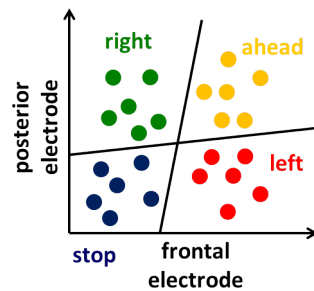
$$I = I_0(1 + \epsilon \cos(2(\Phi - \Phi_S))) \quad (270)$$

The recurrent feedback makes tuning curves **contrast-invariant** [?], as has been observed in numerous experiments!



# 11 COMPUTATION AND CLASSIFICATION

Important task(s) for the brain **and** for machines are **computation** and **classification**.



Aspects of computation/classification:

- o Supervised learning/unsupervised learning
- o Generalization: training set/test set
- o Classification is similar to computation: e.g. Boolean functions
- o Any computation can be performed with a (complex set of) Boolean function (computer!)

In this chapter, we will study classification with three methods: the “classical” (multi-layered) **perceptron**, the **(K)-nearest-neighbor ((K)-NN)** classification, and **support vector machines (SVMs)**.

Although being quite old, the perceptron is still the basis for understanding more elaborate, and even up-to-date methods (e.g. **deep networks**). It is also the basis for one of the most commonly used methods today, the “workhorse” SVM. Finally, (K)NN classification is a method which is extremely simple to use and to implement, and not too bad in its performance.

## 11.1 Single-Layer Perceptrons

### 11.1.1 Feedforward Networks

**Idea:** We will use **feedforward networks** to associate an input with the correct output.

**Constituents of feedforward (FF) networks::**

- o input layer
- o hidden layer
- o output layer

Simple FF networks: consist only of input+output layer, no hidden layer – these are called **simple perceptrons** [?].

### 11.1.2 Definition of the Perceptron

#### Variables/Terminology:

- o  $x_k$ : input on node  $k$  ( $k = 1 \dots N$ )
- o  $O_i$ : output of node  $i$  ( $i = 1 \dots M$ )
- o  $w_{ik}$ : weights
- o  $\theta_i$ : threshold of node  $i$  (also called bias, offset,...)

The 'dynamics' (there is no explicit dependency on time) is simple: sum the synaptic inputs to an activation variable  $h_i$ ,  $h_i := \sum_k w_{ik}x_k$  and pass the result through a non-linearity:

$$O_i := g(h_i - \theta_i) = g\left(\sum_k w_{ik}x_k - \theta_i\right) \quad (271)$$

This is the dynamics of a **McCulloch-Pitts neuron** [?].

$g$  is an activation function like the ones used in the LNP model or in the Wilson-Cowan population dynamics. Typical examples include a binary output of  $-1/1$ , or of  $0/1$ , or a positive output which monotonously depends on the input and saturates for very low/very high activations  $h$ .

**Trick:** The threshold can be elegantly omitted by introducing an additional node  $k = 0$  with weights  $w_{i0} = \theta_i$  to the output units, which receives a constant input of  $x_0 = -1$ . In the following, we will assume this additional node to be present in all of our examples, and come back later for a closer look on its (graphical) interpretation...

### 11.1.3 Computation

Here we will consider computation/classification with a simple perceptron:

#### (a) Computation is Association:

Assume we have a set of  $p = 1 \dots P$  **input pattern**  $x_{kp}$  associated with specific **output pattern**  $y_{ip}$ . Outputs are independent, so consider only one,  $i = 1$ , and omit  $i$  index in

the following.

We also restrict ourselves on binary classification/association problems  $y_p \in [-1, 1]$ , using threshold units:  $g(h) = \text{sgn}(h)$ .

**(b) Patterns and weights can be understood/visualized as vectors:**

- o ...for one pattern  $p \Rightarrow$  one input vector  $\mathbf{x}_p$
- o ...for one output unit  $i \Rightarrow$  one weight vector  $\mathbf{w}$
- o  $\Rightarrow$  thus,  $\sum_k w_k x_{kp} = \langle \mathbf{w}, \mathbf{x}_p \rangle \Rightarrow$  scalar product!

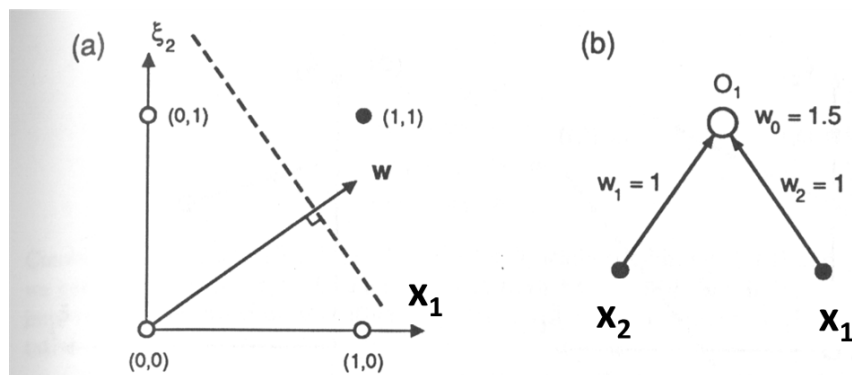
$\Rightarrow$  Conditions to be fulfilled:

- o  $\langle \mathbf{w}, \mathbf{x}_p \rangle > 0$  for a desired output  $O = y_p = 1$
- o  $\langle \mathbf{w}, \mathbf{x}_p \rangle < 0$  for a desired output  $O = y_p = -1$

What is  $\langle \mathbf{w}, \mathbf{x}_p \rangle$  graphically? Remember: it's  $|\mathbf{w}||\mathbf{x}_p| \cos(\text{angle inbetween})$

$\Rightarrow$  thus: Projection of  $\mathbf{x}_p$  on  $\mathbf{w}$  should be positive/negative  $\Rightarrow$  line (hyperplane) perpendicular to  $\mathbf{w}$  separates the two data clouds.

**Which purpose has the threshold?:** The argument of  $g(\dots)$  is exactly zero for all  $\mathbf{x}_{sep}$  which fulfill  $\mathbf{w}\mathbf{x}_{sep} = w_0$ , i.e. for all  $\mathbf{x}_{sep}$  whose projection on  $\mathbf{w}$  is  $w_0$  (separating line!)  $\Rightarrow w_0$  shifts separating line **away from origin**.

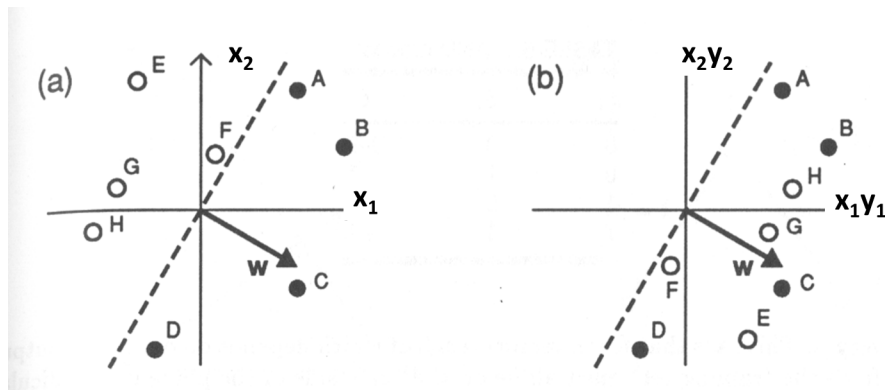
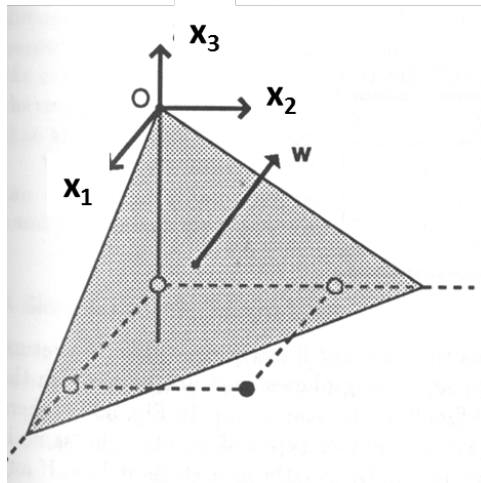


**Insights:**

- o  $N$  input dimensions plus threshold =  $N + 1$  input dimensions without threshold.
- o Classification problem is **equivalent** to mirroring all  $(-1)$ -patterns and searching for a plane through origin such that all points are to one side of plane.

**(c) Problem: linear separability or XOR problem:**

Trying to realize the **XOR-function** in a perceptron leads to the following set of



equations which can not be solved:

$$w_1 + w_2 < w_0 \quad (272)$$

$$-w_1 - w_2 < w_0 \quad (273)$$

$$w_1 - w_2 > w_0 \quad (274)$$

$$-w_1 + w_2 > w_0 \quad (275)$$

E.g., combining first and last:  $w_1 < 0$ , combining the others yields  $w_1 > 0$ . Impossible!

#### 11.1.4 Learning

Simplest form of learning rule  $w_{ik} \rightarrow w_{ik} + \delta w_{ik}$ :

- If  $O_{ip} \neq y_{ip}$ :  $\delta w_{ik} = \eta y_{ip} x_{kp}$ ,
- ...and 0 otherwise.

$\eta$  is the **learning rate**.

Easy to understand: If  $y_{ip}$  has wrong sign, shift weight into direction such that it makes the input in the output layer unit go towards the desired output value (sign).

### Extension of the learning rule/introducing a margin:

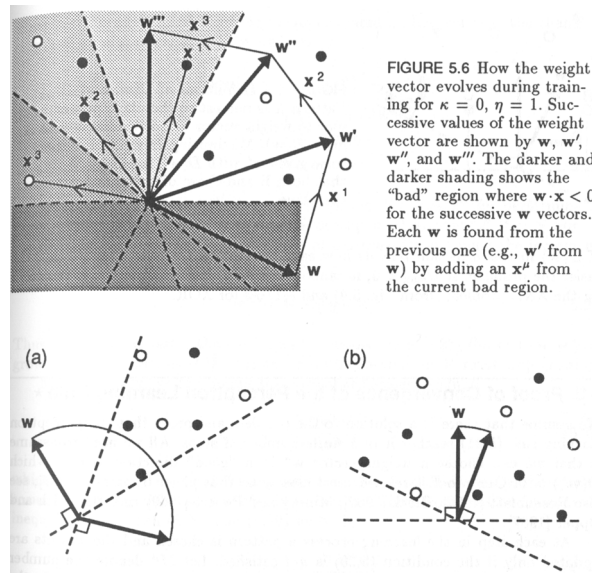
Activation  $h$  should not only have correct sign, but also have some desired magnitude or distance to 0:  $h_{ip} > N\kappa$ .

Thus the learning rule becomes:

$$\delta w_{ik} = \eta \Theta(N\kappa - y_{ip} h_{ip}) y_{ip} x_{kp} \quad (276)$$

Detailed explanation:

- o Assume first  $N\kappa = 0$ : Nothing happens if  $h_{ip}$  has same sign as  $y_{ip}$  (that's the "0 otherwise" in the previous rule...)
- o ...if it has opposite sign, learning rule above is recovered...
- o ...now with  $N\kappa > 0$ : learning still happens even if  $h_{ip}$  **has already the same sign** as  $y_{ip}$ .



### 11.1.5 Continuous output functions and gradient descent

#### Idea:

Change output function from  $\text{sgn}$  to a sigmoidal function with continuous output. This allows to construct a learning rule from a cost function/objective function.

$\Rightarrow O_{ip} = g(\sum_k w_{ik} x_{kp})$  with, e.g.:

(a)  $g(h) := \tanh(\beta h)$  (outputs bounded:  $-1, +1$ )

**(b)**  $g(h) := 1/(1 + \exp(-2\beta h))$  (outputs bounded: 0, 1)

We can now define a **cost function** (over pattern space  $\mathbf{w}$ ): E.g. mean quadratic distance of actual outputs to desired outputs:

$$E(\mathbf{w}) := 1/2 \sum_{ip} (y_{ip} - O_{ip})^2 \quad (277)$$

Now we have to find the  $\mathbf{w}$  that minimizes  $E$  – either compute it directly (normally not feasible), or perform a gradient descent!

Now, let's minimize w.r.t.  $w_{ik} \implies$  the sum over output units collapses to the selected  $i$ :

$$\frac{\partial E}{\partial w_{ik}} = - \sum_p (y_{ip} - O_{ip}) \frac{dO_{ip}}{dw_{ik}} \quad (278)$$

$$= - \sum_p (y_{ip} - O_{ip}) g'(h_{ip}) x_{kp} \quad (279)$$

$\implies$  with  $\delta_{ip} := (y_{ip} - O_{ip}) g'(h_{ip})$ , during presentation of one pattern  $p$ , the learning rule becomes:

$$\delta w_{ik} = \eta \delta_{ip} x_{kp} \quad (280)$$

#### Remark:

This is similar to the SGN-case before, since  $\delta w_{ik} = \eta y_{ip} x_{kp}$  if actual and desired input are different, where  $y_{ip}$  determines the **direction** of learning...

**Technical remark:** Sigmoids **a)** and **b)** are very convenient, since  $g'(h) = \beta(1 - g^2)$  for **a)**, and  $g'(h) = 2\beta g(1 - g)$  for **b)**...

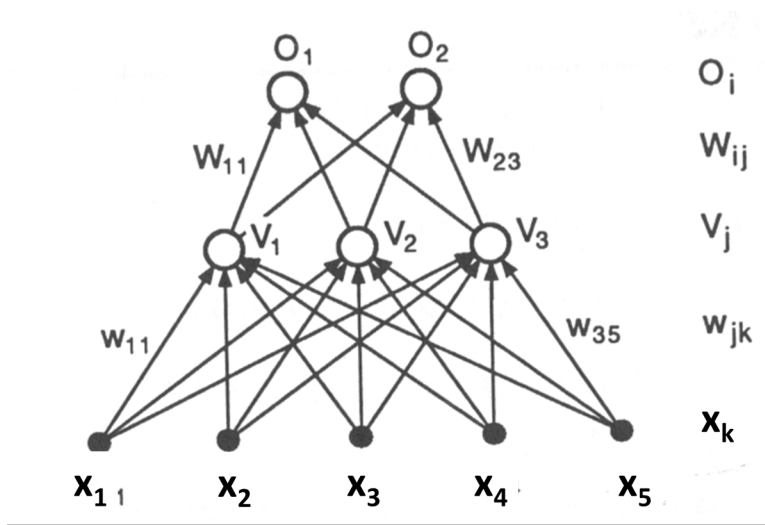
## 11.2 Multilayer Perceptrons

The existence of an error function also allows to derive learning rules for more complex networks: multi-layer perceptrons (MLPs).

MLPs are extensions of SLPs, achieved by adding several hidden layers.

#### Formal description:

- $x_k$ : input on node  $k$
- $w_{jk}$ : weights to HIDDEN layer
- $V_j$ : outputs of HIDDEN layer



- $W_{ij}$ : weights to OUTPUT layer
- $O_i$ : output of OUTPUT layer

### 11.2.1 The Backpropagation Algorithm

The **backpropagation algorithm** is used to learn the weights in a MLP. Derivation of **Backprop learning rule**:

With the new hidden layer, the equations for the perceptron become:

$$h_{jp} = \sum_k w_{jk} x_{kp} \quad (281)$$

$$V_{jp} = g(h_{jp}) = g\left(\sum_k w_{jk} x_{kp}\right) \quad (282)$$

$$H_{ip} = \sum_j W_{ij} V_{jp} = \sum_j W_{ij} g\left(\sum_k w_{jk} x_{kp}\right) \quad (283)$$

$$O_{ip} = g(H_{ip}) = g\left(\sum_j W_{ij} V_{jp}\right) \quad (284)$$

$$= g\left(\sum_j W_{ij} g\left(\sum_k w_{jk} x_{kp}\right)\right) \quad (285)$$

We assume the energy function is defined as:

$$E(\mathbf{w}, \mathbf{W}) = \frac{1}{2} \sum_{p,i} [y_{ip} - O_{ip}(\mathbf{w}, \mathbf{W})]^2 \quad (286)$$

Replacing  $O$  by equation (285) yields:

$$E(\mathbf{w}, \mathbf{W}) = \frac{1}{2} \sum_{p,i} \left[ y_{ip} - g \left( \sum_j W_{ij} g \left( \sum_k w_{jk} x_{kp} \right) \right) \right]^2 \quad (287)$$

For learning the  $w$ 's and  $W$ 's, we perform a gradient descent on  $E$ . For  $W$  we obtain

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} \quad (288)$$

$$= \eta \sum_p [y_{ip} - O_{ip}] g'(H_{ip}) V_{jp} \quad (289)$$

$$= \eta \sum_p \Delta_{ip} V_{jp}, \quad (290)$$

where we introduced the error  $\Delta$ , which is a product of the output error with the slope of the gain function at the current activation  $H$ ,

$$\Delta_{ip} := g'(H_{ip}) [y_{ip} - O_{ip}]. \quad (291)$$

For the  $w$ 's, calculations are a bit more complicated

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \quad (292)$$

$$= -\eta \sum_p \frac{\partial E}{\partial V_{jp}} \frac{\partial V_{jp}}{\partial w_{jk}} \quad (293)$$

$$= \eta \sum_{p,i} [y_{ip} - O_{ip}] g'(H_{ip}) W_{ij} g'(h_{jp}) x_{kp} \quad (294)$$

$$= \eta \sum_{p,i} \Delta_{ip} W_{ij} g'(h_{jp}) x_{kp} \quad (295)$$

$$= \eta \sum_p \delta_{jp} x_{kp} \quad (296)$$

where we introduced the error  $\delta$  via

$$\delta_{jp} := g'(h_{jp}) \sum_i W_{ij} \Delta_{ip}. \quad (297)$$

It is obvious that this scheme can be extended to an arbitrary number of hidden layers.

Interpretation of learning rule:

- **Activations** are propagated forward, and...

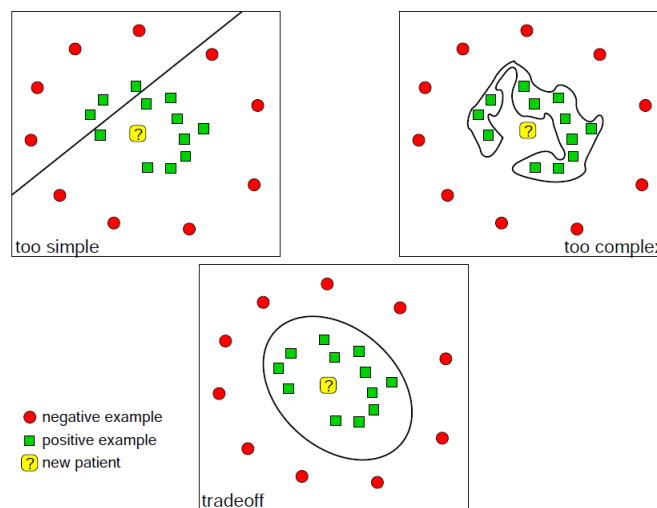
- o ...**Errors** are propagated backwards.
- o The update rule is **local**, involving one forwardly and one backwardly propagated quantity
- o General form:

$$\Delta w_{jk} = \eta \sum_p \delta_{j,output} V_{k,input} \quad , \quad (298)$$

...with  $\delta_{j,output}$  being the error between desired and actual output, and  $V_{k,input}$  the input activation of the corresponding connection  $w_{jk}$ .

### 11.2.2 Some (General) Remarks on Learning

- o **Batch** versus **online** learning:
  - Online: normally stochastic, better exploration of energy landscape, normally better
  - Batch: in certain cases, could learn faster
- o **Generalization** and **overfitting**:



- Learning and computing performance on one data set is very dangerous...
- ...think of just collecting a **dictionary**, on which classification would be 100% perfect
- ...this is called “overfitting”
- Better: separate problem into training and test sets
- Generalization: how well does classifier perform on the whole ensemble?
- Procedures to prevent overfitting: reduce complexity of estimator (i.e. number of hidden units), stop learning before the training data is fitted too well, investigate how classifier changes if you are leaving training data samples out...

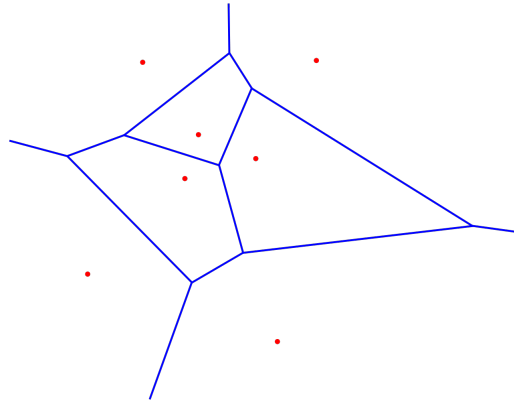
## 11.3 (K)-Nearest-Neighbor Classification

(K)-Nearest neighbor classification requires a set of training samples  $\mathbf{x}_p$  with corresponding outputs  $O_p$ , and a metrics  $d(\mathbf{x}, \mathbf{x}')$  in input space.

For  $K = 1$ , the classification for a test sample  $\mathbf{x}^*$  is given by the output  $O_{p^*}$  of the nearest input vector  $\mathbf{x}_{p^*}$ :

$$\mathbf{x}^* \mapsto O_{p^*} \quad \text{with} \quad p^* := \arg \min_p d(\mathbf{x}^*, \mathbf{x}_p) \quad (299)$$

This classifier imposes a **Voronoi tessellation** on the input space, assigning each test sample a region in which a corresponding test sample would receive the same output:



For  $K > 1$ , the  $K$  nearest neighbors are chosen and the corresponding outputs merged by using an appropriate metrics (e.g. the average for graded outputs  $O$ ).

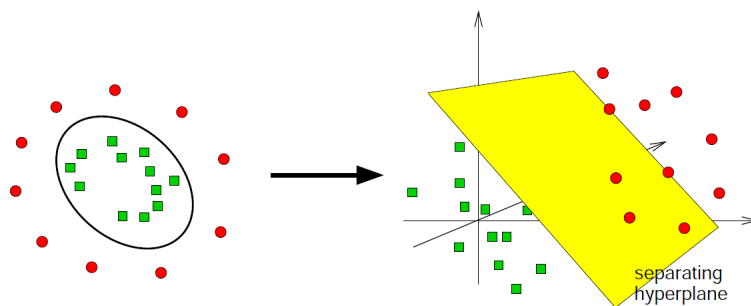
## 11.4 Support Vector Machines

The **support vector machine** (SVM) can be seen as an extension of the perceptron. A SVM can be interpreted as

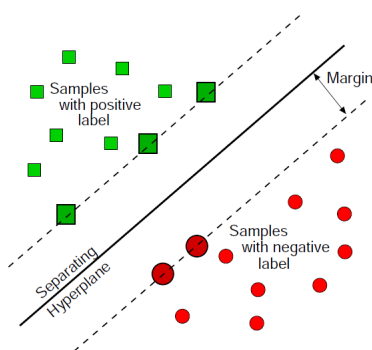
- (a) ...first transforming the data from a low-dimensional input space into a higher-dimensional space,
- (b) ...thus making the classification problem linearly separable, allowing now to use a linear classifier.

Mapped back into input space, the linear classifier becomes a non-linear classifier of arbitrary complexity.

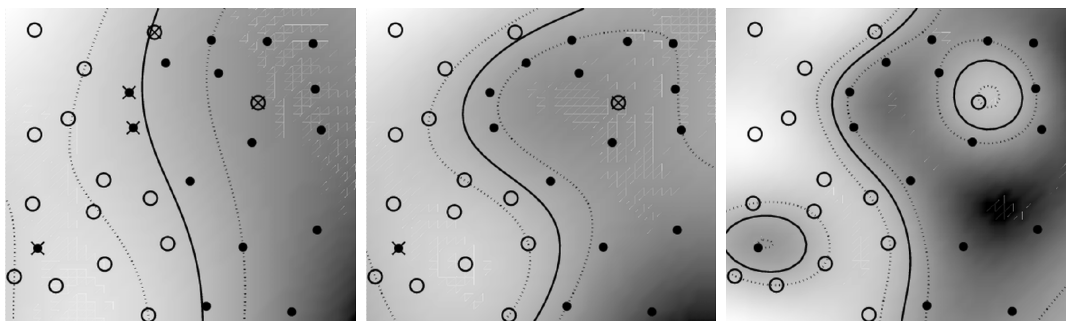
The **support vectors** are the vectors of the training data that are closest to the separating



hyperplane (which is positioned as to maximize their distance to the data clouds).



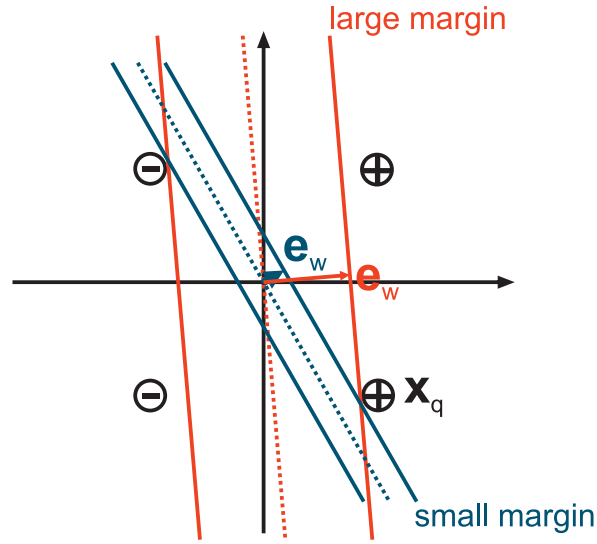
These figures show the same classification problem, with the separating line between the data sets given by three different SVMs with increasing complexity (from left to right).



### 11.4.1 Maximizing the Margin

We will first focus on the last step: linear separation. This problem is identical to perceptron classification. Now, we want to improve our choice of the separation plane and require it to

separate the data with a maximum **margin**  $m$ :



Some considerations:

- (a) Let's first note that the **direction** of  $\mathbf{w}$  is most important. Its **length** can be chosen arbitrarily, i.e., if you scale weight vector and threshold by the same factor  $\lambda$ , it defines the same decision boundary:  $\text{sgn}(\mathbf{w}\mathbf{x} - \theta) = \text{sgn}(\lambda(\mathbf{w}\mathbf{x} - \theta))$  (in the following, we will again omit the explicit threshold and assume it's implicitly defined by an additional input node  $k = 0$ ).
- (b) We now consider all  $\mathbf{w}$ , for which  $\min_p |\mathbf{w}\mathbf{x}_p| = 1$ . Depending on the direction of  $\mathbf{w}$ ,  $\mathbf{e}_w := \mathbf{w}/|\mathbf{w}|$ , the length  $|\mathbf{w}|$  can be different (see figure above).
- (c) If  $q = \arg \min_p |\mathbf{w}\mathbf{x}_p|$ ,  $\mathbf{x}_q$  lies closest to the decision boundary (a "support vector") and the margin is given by its projection  $m = \mathbf{x}_q \mathbf{e}_w$ .
- (d) Since  $\min_p |\mathbf{w}\mathbf{x}_p| = \mathbf{w}\mathbf{x}_q = |\mathbf{w}|m = 1$ , maximizing  $m$  implies minimizing  $|\mathbf{w}|$  or  $|\mathbf{w}|^2$ !

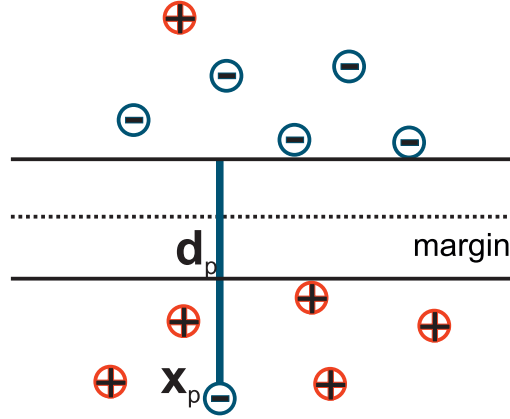
To achieve a good classification performance and a good separation of the data clouds, we therefore combine these two requirements into one **energy function** or **loss function**:

$$E := \frac{1}{P} \sum_{p=1}^P l[\mathbf{w}\mathbf{x}_p, y_p] + \frac{1}{2} |\mathbf{w}|^2 \quad (300)$$

which can be minimized w.r.t.  $\mathbf{w}$  subject to  $\min_p |\mathbf{w}\mathbf{x}_p| = 1$ . The first term of  $E$  is also called the **training error** and the second term the **complexity term**. One straightforward choice of  $l$  would be  $l[\dots] := 0$  if pattern  $p$  is correctly classified, and 1 else (step function).

### 11.4.2 Dealing with Non-Separable Cases

In general, in every state-of-the-art classification problem there are data points which can not be classified correctly (given a SVM of a certain complexity with good generalization properties). How do we handle these “exceptions”?



(a) First we assume we can separate the data perfectly. Thus we can minimize  $|\mathbf{w}|^2$ , subject to

$$\mathbf{w}\mathbf{x}_p \geq 1 \quad \text{if } y_p = 1 \quad (301)$$

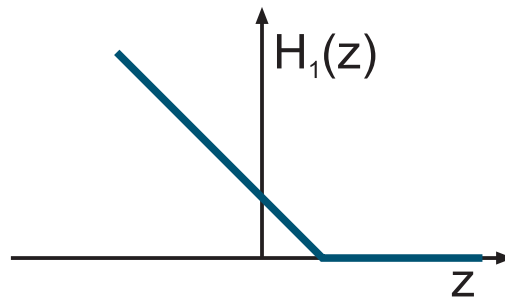
$$\mathbf{w}\mathbf{x}_p \leq -1 \quad \text{if } y_p = -1, \quad (302)$$

which can be compacted to  $y_p(\mathbf{w}\mathbf{x}_p) \geq 1$ .

(b) For dealing with the non-separable case, i.e. patterns that are misclassified with a distance  $d_p \geq 0$  to the decision boundary, we can require  $y_p(\mathbf{w}\mathbf{x}_p) \geq 1 - d_p$  and minimize

$$\frac{1}{2}|\mathbf{w}|^2 + C \sum_p d_p. \quad (303)$$

It is equivalent to choosing  $l[\mathbf{w}\mathbf{x}_p, y_p] := \max(0, 1 - y_p(\mathbf{w}\mathbf{x}_p))$ , the so-called **hinge-loss** function  $H_1(z) := \max(0, 1 - z)$ .



### 11.4.3 The Kernel Trick

Basic idea of an SVM is to transform the classification problem into a higher-dimensional space,  $\mathbf{x} \rightarrow \Phi(\mathbf{x})$ , where the problem is linearly separable.

The idea of the **Kernel Trick** is to choose  $\Phi$  such that for learning and using an SVM, it is not necessary any more to actually perform this transformation. This becomes possible if we can express the equations for learning and using an SVM (normally defined in “ $\Phi$ -space” ...) as operations in the **original input space**.

Basis of the corresponding formalism is the **representer theorem** [?]

$$\mathbf{w} = \sum_p \alpha_p \Phi(\mathbf{x}_p) \quad (304)$$

Thus  $\mathbf{w}$  can alternatively be optimized by optimizing the  $\alpha_p$  instead!

The decision rule now becomes

$$O_p = \text{sgn}(\mathbf{w}\Phi(\mathbf{x})) = \text{sgn}\left(\sum_p \alpha_p \Phi(\mathbf{x}_p)\Phi(\mathbf{x})\right) \quad (305)$$

$$= \text{sgn}\left(\sum_p \alpha_p K(\mathbf{x}_p, \mathbf{x})\right), \quad (306)$$

where the product of the  $\Phi$ 's defines a **kernel function**  $K$ ,

$$K(\mathbf{x}, \mathbf{x}') := \Phi(\mathbf{x})\Phi(\mathbf{x}'). \quad (307)$$

Also the margin becomes a function of  $K$ ,

$$|\mathbf{w}|^2 = \left| \sum_p \alpha_p \Phi(\mathbf{x}_p) \right|^2 \quad (308)$$

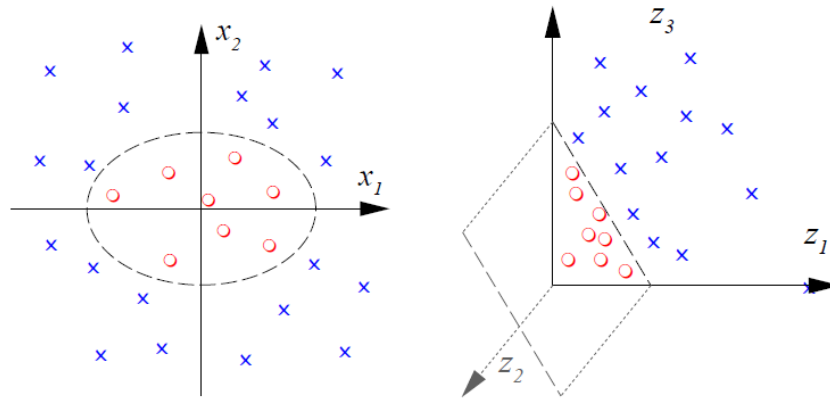
$$= \left| \sum_p \sum_{p'} \alpha_p \alpha_{p'} \Phi(\mathbf{x}_p) \Phi(\mathbf{x}_{p'}) \right| \quad (309)$$

$$= \left| \sum_{p,p'} \alpha_p \alpha_{p'} K(\mathbf{x}_p, \mathbf{x}_{p'}) \right|. \quad (310)$$

In consequence, both learning and classification can be expressed as functions of  $K$ !

### 11.4.4 Examples for Kernels and SVMs

- Polynomial kernel:



$$K(\mathbf{x}, \mathbf{x}') := (\mathbf{x} \cdot \mathbf{x}' + 1)^\gamma \quad (311)$$

Example: Consider the following mapping  $\Phi$  (see figure):

$$\Phi : (x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2) \quad (312)$$

This corresponds to a kernel  $K(\mathbf{x}, \mathbf{x}') := (\mathbf{x} \cdot \mathbf{x}')^2$  (compute!)

- Radial basis function (RBF) kernel:**

$$K(\mathbf{x}, \mathbf{x}') := \exp(-\gamma(\mathbf{x} - \mathbf{x}')^2) \quad (313)$$

This kernel has a nice interpretation in data space if we consider the argument of the decision function

$$\sum_p \alpha_p \exp(-\gamma(\mathbf{x}_p - \mathbf{x})^2), \quad (314)$$

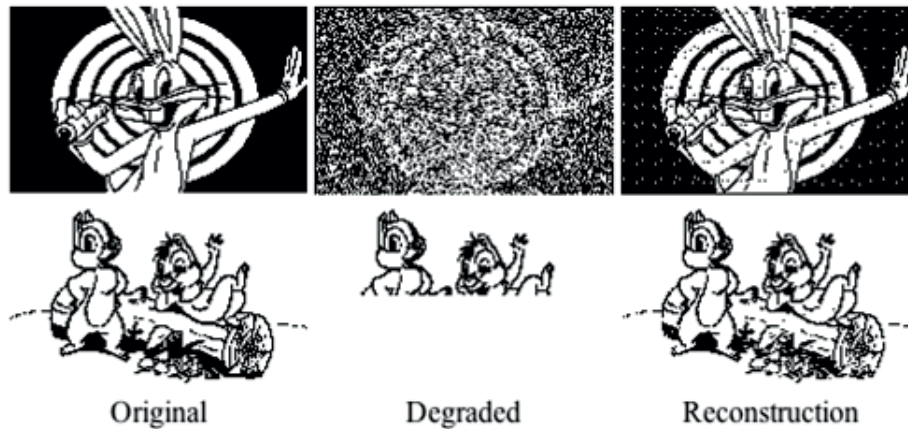
since it “adds” a bump of height  $\alpha$  around each data point and classifies according to the weighted sum of all contributions

An excellent source of information: [www.kernel-machines.org](http://www.kernel-machines.org)!!!

## 12 MEMORY AND LEARNING

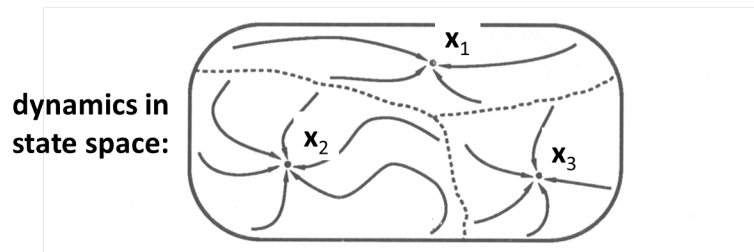
### 12.1 The Hopfield Model

#### 12.1.1 The Paradigm



**Goal:** we want to realize an **associative memory**:

- o Store patterns  $\mathbf{x}_p$  in the network...
- o If stimulated with an input pattern  $\mathbf{x}_0$ , recall pattern  $\mathbf{x}_p$  resembling most closely the input



**Idea:** We do it with a **recurrent network** of McCulloch-Pitts neurons:

- o Patterns defined as  $x_{kp} = +1/-1$ , also state of neuron  $S_i$  as  $-1/+1$ :
- o Update equation:

$$S_i \rightarrow \text{sgn}\left(\sum_j w_{ij} S_j - \theta_i\right) \quad (315)$$

- o It's a **recurrent network** or **iterative scheme**: The next state  $t + 1$  of the network depends on the previous state  $t$ :  $S_i^{t+1} = \text{sgn}(\sum_j w_{ij} S_j^t - \theta_i)$
- o Simplification: we drop  $\theta_i$  for **random patterns** ( $-1$  and  $+1$  occur with same probability), not useful for this problem.

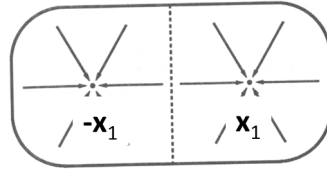
If all  $S_i$  are updated in one iteration step, this is called a **parallel** or **synchronous update**. If only one (e.g. randomly selected)  $S_i$  is updated, this is called a **serial** or **asynchronous** (or **random**) update.

**Question:** How do we choose  $w_{ij}$ ?

### 12.1.2 Choosing the weights

#### (a) One pattern $\mathbf{x}$ – motivation:

- o To be fulfilled for a stable activation:  $x_i = \text{sgn}(\sum_j w_{ij}x_j)$
- o ...is satisfied for  $w_{ij} \propto x_i x_j$
- o ...choose proportionality constant to be  $1/N$ ,  $w_{ij} := 1/N x_i x_j$
- o ...this choice is also **correcting errors**, since:
  - ...if majority of  $S_j = x_j$ ,  $h_i = \sum_j w_{ij} S_j$  has the correct sign
  - ...thus  $\mathbf{x}$  is an **attractor** of the network's dynamics
  - ...but also  $-\mathbf{x}$



#### (b) Many patterns:

- o Superposition:  $w_{ij} := 1/N \sum_p x_{ip} x_{jp}$
- o Resemblance to **Hebb's rule** – what fires together, wires together!
- o Question: Is it still an attractor, and stable? ...to be fulfilled:  $x_{ip} = \text{sgn}(\sum_j w_{ij} x_{jp})$

$$h_{ip} = \sum_j w_{ij} x_{jp} \quad (316)$$

$$= \frac{1}{N} \sum_j \sum_q x_{iq} x_{jq} x_{jp} \quad (317)$$

$$= x_{ip} \quad (\text{for } p = q) \quad (318)$$

$$+ \frac{1}{N} \sum_j \sum_{q \neq p} x_{iq} x_{jq} x_{jp} \quad (\text{for } p \neq q) \quad (319)$$

...second term is the **crosstalk term**, and its magnitude should be smaller than 1!

The crosstalk term gives a means to compute storage capacity.

### 12.1.3 Storage capacity

Here we discuss a method to compute approximately storage capacity:

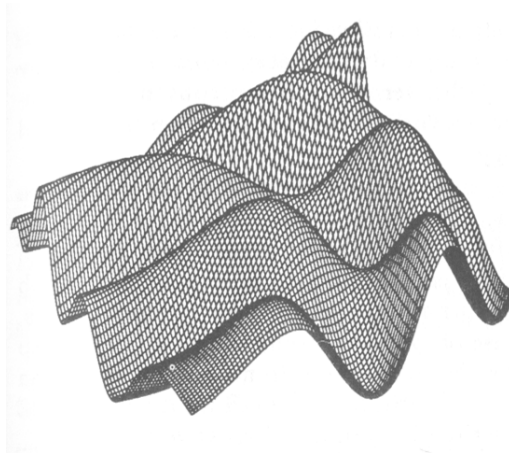
- Consider the crosstalk term multiplied by  $-x_{ip}$ , denote by  $C_{ip}$
- ...if  $C_{ip} > 1$ , the unit/bit will flip
- ...we assume  $N$  and  $P \gg 1$
- ...thus, we have to cope with a sum of  $NP$  independent numbers being  $+1/-1 \Rightarrow$  binomial distribution can be approximated by Gaussian of variance  $\sigma^2 = P/N$
- ...hence,  $P_{err} = 0.5(1 - \text{erf}(\sqrt{N/2P}))$ .

$\Rightarrow$  For a desired maximum flip probability  $P_{err}$  (e.g. 0.05), the number  $P$  of patterns must not exceed a critical value  $P_{max}$  given by this equation.

#### Problem:

This simplified capacity calculations tells us only what happens in one iteration step. However, there's the possibility of an **avalanche** of flips. A more sophisticated calculation reveals  $P_{max} \approx 0.138N$ .

### 12.1.4 Energy function of the Hopfield network



The existence of an **energy function** (or **objective function**) allows us to derive a dynamics from first principles:

$$H := -\frac{1}{2} \sum_{ij} w_{ij} S_i S_j \quad (320)$$

The idea of introducing an energy is that it always decreases under the dynamics, and that

attractors are minima in the energy landscape. A nice analogue is the picture of a particle sliding downhill...

### Why is an energy function useful?

- ...for finding the steady states by analytical methods without “simulating” the network
- ...for deriving a system’s dynamics from first principles, for proposing a computational goal and for deriving rules how it can be reached
- ...for investigating the stability of the system/dynamics

(a) **We first show that  $H$  decreases under the Hopfield dynamics for a serial update:**

$$H = -\frac{1}{2} \sum_{ij} w_{ij} S_i S_j \quad (321)$$

$$= C - \sum_{(ij)} w_{ij} S_i S_j \quad \text{with } (ij) \text{ being distinct pairs, and without } ii \quad (322)$$

- ...if no  $S_i$  changes, there is no change in  $H$
- ...let us assume  $S_i$  changes, thus  $S_i^{t+1} = \text{sgn}(\sum_j w_{ij} S_j^t) = -S_i^t$  (\*)
- ...now consider  $H^{t+1} - H^t$ , but just terms containing  $i$  (the others don't change):

$$H^{t+1} - H^t = \sum_{(j \neq i)} w_{ij} (-S_i^{t+1} S_j^t + S_i^t S_j^t) \quad (323)$$

$$= 2S_i^t \sum_{(j \neq i)} w_{ij} S_j^t \quad (324)$$

$$= 2S_i^t \sum_j w_{ij} S_j^t - 2w_{ii} \quad (325)$$

- ...first term is negative, since  $S_i$  changes sign (\*), thus  $H^{t+1} - H^t < 0$ .

(b)  **$H$  can be derived under condition that patterns be a minimum:**

- We want  $H$  to be minimal if the actual state is a copy (positive or negative) of a pattern:

$$H = -\frac{1}{2N} \sum_p \left( \sum_i S_i x_{ip} \right)^2 \quad (326)$$

- **Proof:** by multiplying out the square and by exchanging the sums over  $p$  and  $i$  ( $i$  first/ $p$  last).
- $\implies w_{ij} = \frac{1}{N} \sum_p x_{ip} x_{jp}$ .

## 12.1.5 Further aspects

### (a) Existence of **spurious states**:

- o ...first, original and inverse pattern is stable – no problem
- o ...second, mixtures of patterns could also be stable, e.g.  

$$x_i^{new} := \text{sgn}(\pm x_{i1} \pm x_{i2} \pm x_{i3}!)$$
- o ...third, there exist **local minima** for large sets of patterns.

### (b) Rules for **updating**:

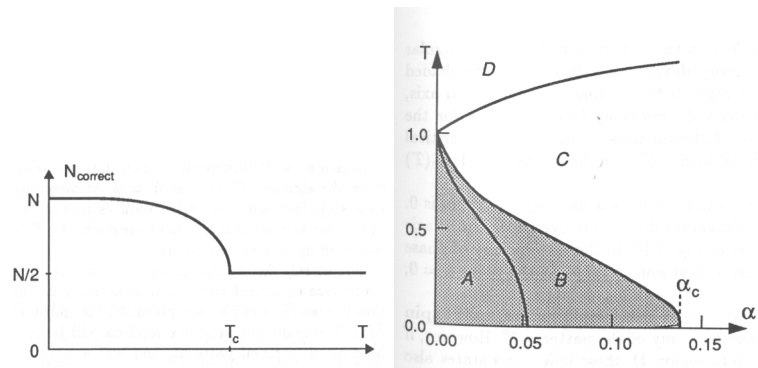
- o ...synchronous: can lead to oscillations, can become stuck in minima
- o ...asynchronous: can be slow

### (c) Sigmoidal **transfer functions** and **stochastic update**:

- o idea (A): relation to spins in a ferromagnet at a finite temperature
- o idea (B): probabilistic synaptic transmission, neural noise
- o ...formally: replace  $\text{sgn}(h)$  by a sigmoidal function, one example for update would then be:  $S_i = +1$  with probability  $g(h_i)$ ,  $-1$  with probability  $1 - g(h_i)$ , and with  $g(\dots)$ , for example,  $g(h_i) := 1/(1 + \exp(-2\beta h))$ .

### (d) **Mean field theory**: The idea:

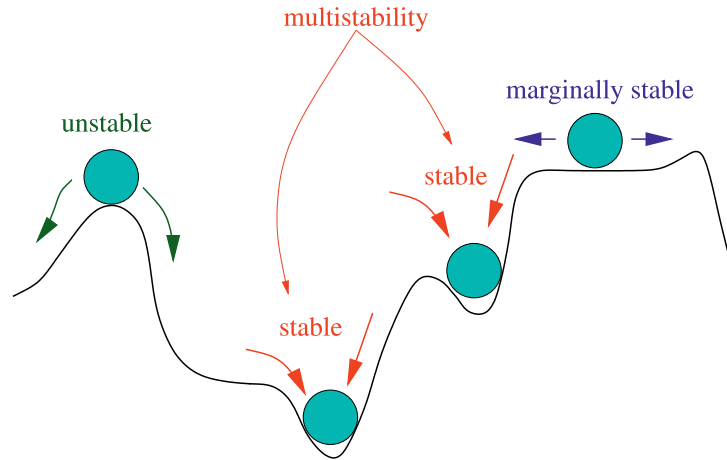
- o ...analyze stochastic systems by replacing state variables by their means
- o ...depending on the **temperature** (i.e., stochasticity), there will be phase transitions



## 13 APPENDIX

### 13.1 Stability analysis

#### 13.1.1 Fixed points



In the long-term limit, some dynamical systems become very boring: they settle into a state where the values of the independent variables remain constant. Such a state is called a **stable fixed point**, and the initial conditions for which this system converges into this state are called its **basin of attraction**.

Generally, there are also unstable, and marginally stable fixed points. Once a fixed point for a dynamical system has been found, it is important to know its properties in order to find out how the system behaves for small perturbations in that state.

If there are more than one fixed point, the system is multistable: depending on its initial conditions, it converges into one of the possible stable fixed points.

Consider a dynamical system obeying the differential equation (DEQ)

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}). \quad (327)$$

For  $\mathbf{x}_f$  being a fixed point, Eq.327 must yield

$$\left. \frac{d\mathbf{x}}{dt} \right|_{\mathbf{x}_f} \equiv 0 = \mathbf{f}(\mathbf{x}_f). \quad (328)$$

This fixed point can either be found by solving equation (328), or by guessing one and showing that it fulfills equation (328).

If we focus on the single components  $f_i$  of an  $N$ -dimensional dynamics  $\mathbf{x}_f = \{f_1, f_2, \dots, f_N\}$ , equation (328) defines  $N$  implicit equations in the variables  $x_1, x_2, \dots, x_N$ , the so-called **nullclines** of a dynamics. The nullclines are  $N - 1$ -dimensional manifolds (for  $N = 2$ : curves), along which one variable is constant over time, i.e. its derivative w.r.t. time is zero. Points where **all** nullclines intersect are fixed points under the dynamics. Stability can be assessed by considering the change of the **other** variable(s) along the nullcline, helping to find out whether fixed points are stable or not.

**Example 1: Finding the fixed point/steady state: Integrate-and-fire neuron with subthreshold input:**

The DEQ for an integrate-and-fire neuron receiving constant synaptic input reads

$$C_m \frac{dV}{dt} = \frac{1}{R_m} (V_{rest} - V) + I_{syn}. \quad (329)$$

For  $dV/dt = 0$ , equation (329) has one fixed point  $V_f$ .

$$0 = \frac{1}{R_m} (V_{rest} - V_f) + I_{syn} \quad (330)$$

$$\rightarrow V_f = R_m I_{syn} + V_{rest}. \quad (331)$$

**Remark:** This fixed point only exists for sufficiently low  $I_{syn}$ , otherwise  $V_f$  will be never reached because it lies higher than the firing threshold  $V_{th}$ , at which the neuron is reset to its resting potential.

### 13.1.2 Stability of $N$ -dimensional systems

The **stability** of a fixed point  $\mathbf{x}_f$  is found doing a **linear stability analysis**. In order to do this, an expansion of the corresponding (nonlinear) DEQ around the fixed point is made, and all higher order terms of this expansion are neglected (like a Taylor series expansion of  $f(x)$  without the terms  $x^2, x^3, \dots$ ).

From this procedure, one obtains an easily solvable linear DEQ, which approximates the behaviour of the 'real' DEQ in the neighbourhood of  $\mathbf{x}_f$  and allows to compute the stability of this fixed point:

Assume  $\mathbf{x}_f$  being a fixed point of equation (327). We substitute  $\mathbf{x}$  by  $\mathbf{x}_f + \delta\mathbf{x}$ , where  $\delta\mathbf{x}$  is a

small deviation from  $\mathbf{x}_f$ , and expand around  $\mathbf{x}_f$ :

$$\frac{d(\mathbf{x}_f + \delta\mathbf{x})}{dt} = \mathbf{f}(\mathbf{x}_f + \delta\mathbf{x}) \quad (332)$$

$$0 + \frac{d\delta\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}_f) + \mathbf{f}'(\mathbf{x}_f)\delta\mathbf{x} + \mathcal{O}(\delta\mathbf{x}^2); \quad (333)$$

Because of  $\mathbf{x}_f$  being a fixed point,  $\mathbf{f}(\mathbf{x}_f) = 0$ .

By neglecting terms of  $\mathcal{O}(\delta\mathbf{x}^2)$ , we obtain the linearized DEQ

$$\frac{d\delta\mathbf{x}}{dt} = \mathbf{f}'(\mathbf{x}_f)\delta\mathbf{x}, \quad (334)$$

whose solution is given by the expression

$$\delta\mathbf{x}(t) = \delta\mathbf{x}(0) \exp(\Lambda t) \quad (335)$$

$$\text{with } \Lambda := \mathbf{f}'(\mathbf{x}_f) = \left( \frac{\partial f_i}{\partial x_j}(\mathbf{x}_f) \right)_{i,j} \quad (336)$$

$$\text{and } \exp(\Lambda t) := \sum_{k=0}^{\infty} \frac{\Lambda^k}{k!} t^k \quad (337)$$

Stability now depends vitally on  $\exp(\Lambda t)$ : if  $\|\exp(\Lambda t)\|$  increases exponentially with time, the fixed point is unstable, if  $\|\exp(\Lambda t)\|$  decreases exponentially, the fixed point is stable against a small perturbation  $\delta\mathbf{x}(0)$ .

### 13.1.3 Stability of one-dimensional systems

In one dimension  $x$ , stability analysis becomes extremely simple. In this case,  $\Lambda = \lambda$  is a scalar, the **dynamical exponent**, whose sign determines stability of the steady state:

$$\frac{d\delta x}{dt} = f'(x_f)\delta x + \mathcal{O}(\delta x^2) \quad (338)$$

$$= \lambda\delta x + \mathcal{O}(\delta x^2) \quad (339)$$

$$\implies \delta x = \delta x(0) \exp(\lambda t) \quad (340)$$

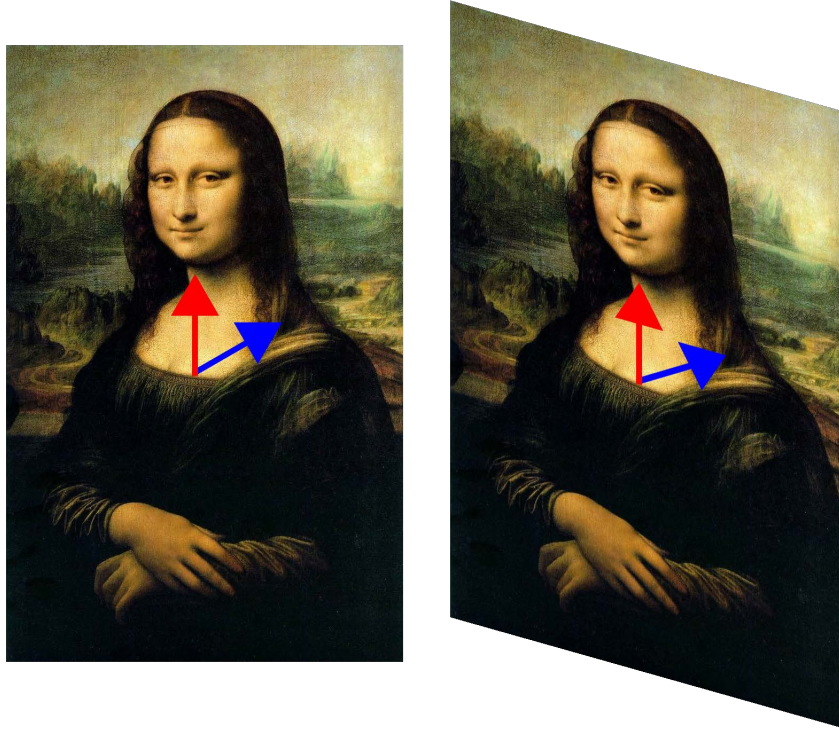
We distinguish between three cases:

- $\lambda < 0$ : stable, perturbation decays exponentially
- $\lambda > 0$ : instable, perturbation grows exponentially
- $\lambda = 0$ : stays where it has been pushed...

### 13.1.4 Stability in more than one dimension

Stability in more than one dimension depends on the properties of the so-called *Jacobi-matrix*  $\Lambda$ . To gain an intuitive understanding we have to consider Eigenvectors  $\mathbf{v}_i$  and Eigenvalues  $\lambda_i$  of this matrix.

Heuristically speaking, an  $N \times N$  matrix  $\Lambda$  defines a linear transformation in an  $N$ -dimensional vector space. Here is one example of a linear transform ('shearing') applied to an image, in a two-dimensional vector space.



The direction in which an Eigenvector  $\mathbf{v}_i$  is pointing remains the same under a linear transformation, this is described by the following equation from vector calculus:

$$\Lambda \mathbf{v}_i = \lambda_i \mathbf{v}_i \quad (341)$$

Let us assume we apply at  $t = 0$  a perturbation which is proportional to an Eigenvector  $\mathbf{v}_i$ ,  $\delta \mathbf{x} = \epsilon \mathbf{v}_i$ . The linearized differential equation then reads

$$\frac{d\delta \mathbf{x}}{dt} = \lambda_i \delta \mathbf{x}, \quad (342)$$

and the solution is  $\delta \mathbf{x}(t) = \epsilon \mathbf{v}_i \exp(\lambda_i t)$ . Here the Eigenvalue  $\lambda_i$  will determine whether the initial, very specific perturbation  $\mathbf{v}_i$ , will grow or decay exponentially.

You can imagine that the same consideration holds also for a *superposition* of Eigenvectors  $\mathbf{v}_i$  and  $\mathbf{v}_j$  as initial perturbation: the solution of the linearized DEQ will also be a superposition of these Eigenvectors decaying/growing with the two Eigenvalues  $\lambda_i$  and  $\lambda_j$ . In particular, the *largest* Eigenvalue will dominate the solution, and if only one of the Eigenvalues is positive the corresponding solution will not be stable (i.e., grow exponentially).

In general, matrices  $\Lambda$  could have two or more Eigenvalues that are identical (termed *multiplicity*), and Eigenvalues are typically not always real numbers but complex-valued. For these reasons, the general solution for a linearized DEQ is composed of terms

$$t^{\mu_i} \exp(\lambda_i) = t^{\mu_i} \exp(\mathcal{Re}\{\lambda_j\}t)(\cos(\mathcal{Im}\{\lambda_j\}t) + i \sin(\mathcal{Im}\{\lambda_j\}t)) \quad , \quad (343)$$

that means, a power in  $t$ , an exponential  $\exp$ , and a harmonics  $\cos$ ,  $\sin$ . For assessing stability, the real part of the Eigenvalue  $\lambda_i$  is essential since it is dominating all other contributions.

So, now that we know that Eigenvalues are important, how do we find them? This is possible by computing the roots for  $\lambda$  of the following equation

$$\det(\Lambda - \lambda E) = 0 \quad , \quad (344)$$

where  $E$  is the unity matrix and  $\det$  the determinant of the matrix enclosed in brackets. For a one-dimensional matrix  $A = [a]$ , the determinant is  $a$  and thus the one and only Eigenvalue is  $\lambda = a$ . For a two-dimensional matrix  $A = [a \ b; c \ d]$  the determinant is  $\det(A) = ad - bc$ . For more than two dimensions, the determinant can be determined (nice alliteration!) by using Laplace's formula.

In the following, some extended information on the formal solution of a linearized DEQ with real-valued components:

#### **Properties of $\exp(At)$ :**

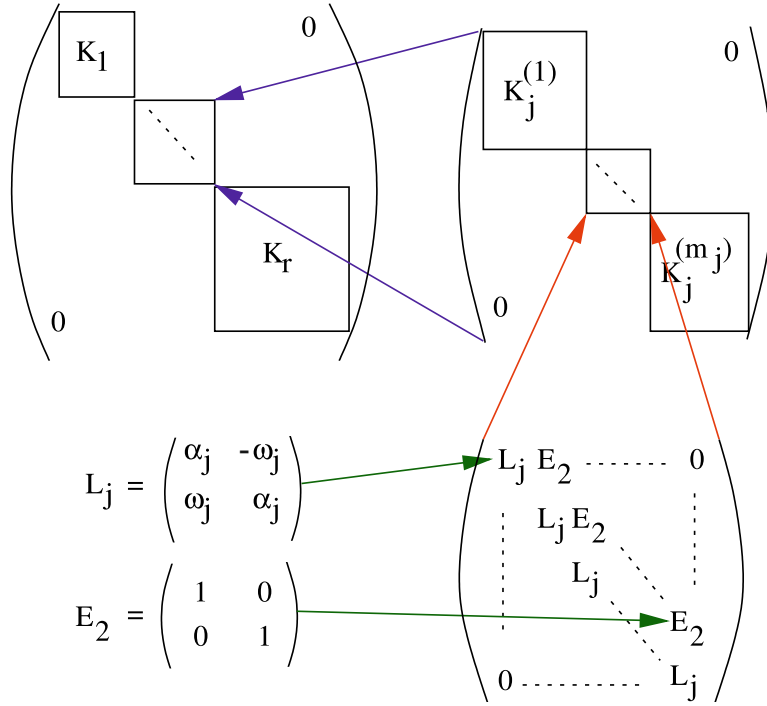
Via a transformation matrix  $Q$ ,  $A$  can be transformed into the **real canonical form**  $Q A Q^{-1}$  with arbitrary complex eigenvalues  $\lambda_j$ . This canonical form is a special type of the Jordan normal form:

On the first view, this looks a bit complex – but this formalism tells us that **all** solutions of equation (334) can be written as linear combinations of the functions

$$t^{l(j)} \exp(\mathcal{Re}\{\lambda_j\}t) \cos(\mathcal{Im}\{\lambda_j\}t) \quad (345)$$

$$t^{l(j)} \exp(\mathcal{Re}\{\lambda_j\}t) \sin(\mathcal{Im}\{\lambda_j\}t) \quad (346)$$

with the eigenvalues  $\lambda_j$  of the matrix  $A$ . As one can see, if at least one of the real parts of the



eigenvalues  $\lambda_j$  is positive,  $\mathbf{x}(t)$  grows exponentially in time. Otherwise, if all  $\mathcal{R}e\{\lambda_j\} < 0$ , every initial perturbation  $\delta\mathbf{x}(0)$  will decrease to zero - this is the criterion for a stable fixed point. If the eigenvalues of  $A$  have nonzero imaginary parts, the solutions of equation (327) will have an oscillatory component. However, this component is dominated by the exponential factors, and therefore does not play a role in determining the fixed points' stability.

### Example 2: Single neuron with feedback:

A neuronal population with positive feedback  $\omega > 0$  is given by the nonlinear DEQ for the population activity  $A$

$$\dot{A} = f(A) := -A + S(wA) \quad (347)$$

$$S(x) := \frac{x}{1+x} \quad (348)$$

The fixed points of the dynamics are found easily by solving for  $A_0$

$$A_0 = \frac{wA_0}{1+wA_0} \quad (349)$$

The first fixed point lies at  $A_{01} = 0$ , the second one at  $A_{02} = 1 - 1/w$  (if  $w \neq 0$ ). To examine the stability of the fixed points, one calculates the linearization of  $f$

$$\frac{\partial f}{\partial A} = -1 + \frac{w}{(1+wA)^2} \quad (350)$$

around the first fixed point,  $(\partial f / \partial A)(A_{01})$

$$\delta \dot{A}_1 = (w - 1)\delta A_1, \quad (351)$$

and the second fixed point,  $(\partial f / \partial A)(A_{02})$

$$\delta \dot{A}_2 = (1/w - 1)\delta A_2. \quad (352)$$

For weak feedback ( $w < 1$ ), the first fixed point is stable and the second one is unstable, and vice versa for strong feedback ( $w > 1$ ). As we require the activation  $A$  to be positive, for  $w < 1$ , the activity either goes to zero or to one, depending on the initial conditions, and for  $w > 1$ , the activity always approaches one.

This result illuminates an interesting property of coupled dynamical system: if the excitatory feedback gets too strong, the system's activity grows exponentially until a saturation level is reached. This saturation level, modeled by the feedback function  $S$ , could be given e.g. by the maximum amount of transmitter a synapse can release, or the maximum firing rate a neuron can have because of other physiological reasons.

### 13.1.5 Stability of two-dimensional systems

For two dimensions, the phenomenology becomes more interesting, but assessing stability is still particularly intuitive and easy to explain. We now consider:

$$\frac{d}{dt}\delta x_1 = a\delta x_1 + b\delta x_2 \quad (353)$$

$$\frac{d}{dt}\delta x_2 = c\delta x_1 + d\delta x_2 \quad (354)$$

Compute the eigenvalues for the matrix  $\Lambda := [a \ b; c \ d]$ :

$$\det(\Lambda - \lambda E) = 0 \quad (355)$$

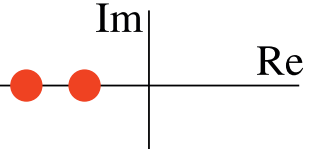
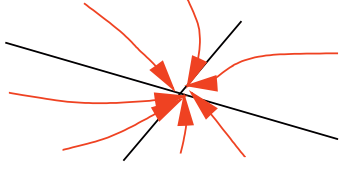
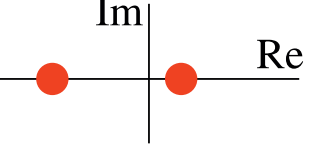
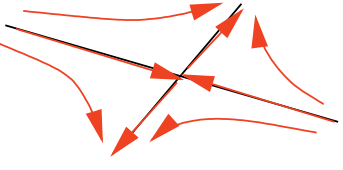
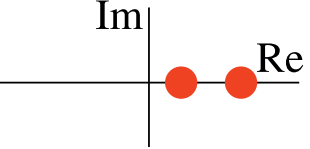
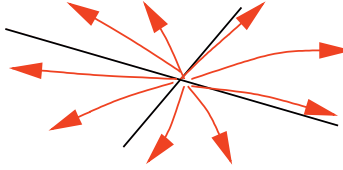
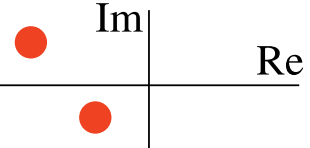
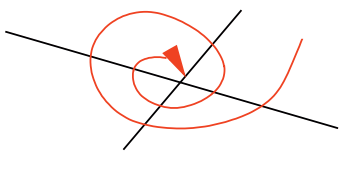
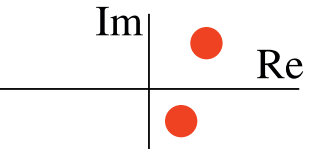
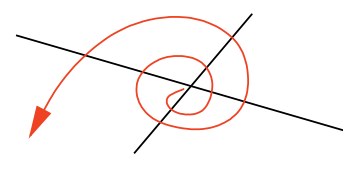
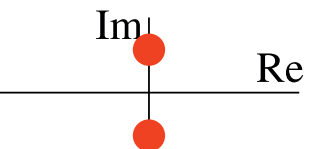
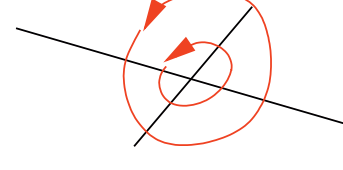
$$(a - \lambda)(d - \lambda) - cb = 0 \quad (356)$$

$$(ad - cb) - \lambda(a + d) + \lambda^2 = 0 \quad (357)$$

$$\det(\Lambda) - \lambda \text{tr}(\Lambda) + \lambda^2 = 0 \quad (358)$$

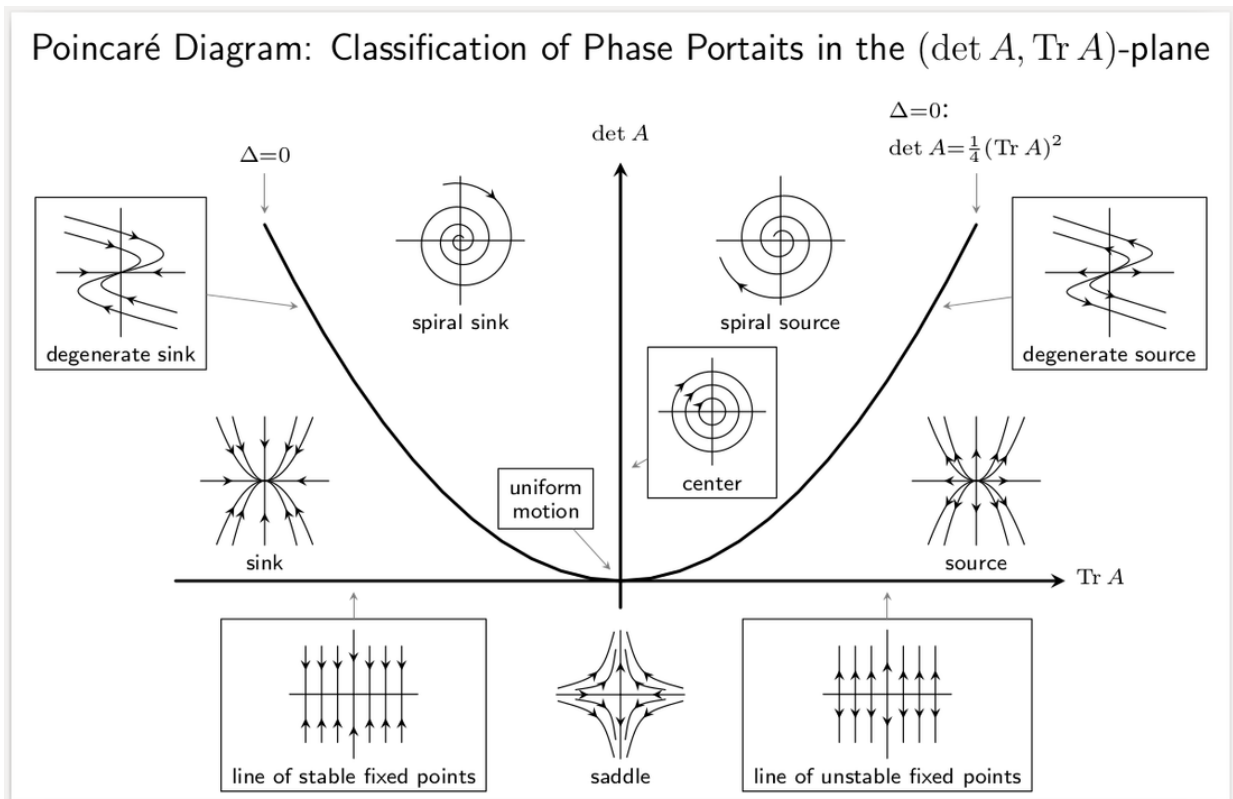
$$\lambda = \frac{1}{2} \left( \pm \sqrt{-4 \det(\Lambda) + \text{tr}(\Lambda)^2} + \text{tr}(\Lambda) \right) \quad (359)$$

**Different types of fixed points:**

Eigenvalues	Type	Trajectories
 <p>Im Re</p>	stable node	
 <p>Im Re</p>	saddle	
 <p>Im Re</p>	instable node	
 <p>Im Re</p>	stable vertex	
 <p>Im Re</p>	instable vertex	
 <p>Im Re</p>	marginal stable vertex	

We can distinguish the following cases:

- (a)  $\det(\Lambda)$  **negative**: The argument of the root is always positive, so there are no imaginary values for  $\lambda$ . The root is also always bigger than  $|\text{tr} \lambda|$ , so there will always be one positive and one negative Eigenvalue  $\Rightarrow$  saddle point!
- (b)  $\det(\Lambda)$  **positive**: We have to distinguish whether the argument of the root will be negative, then the Eigenvalues will be imaginary. This is **not** the case for  $\det(\Lambda) \leq 1/4 \text{tr}(\Lambda)^2$ :
  - (a) ...thus for  $\text{tr}(\Lambda) < 0 \Rightarrow$  two negative Eigenvalues, stable node.
  - (b) ...thus for  $\text{tr}(\Lambda) > 0 \Rightarrow$  two positive Eigenvalues, instable node.
  - (c) ...and if we **have** imaginary Eigenvalues, i.e.  $\det(\Lambda) > 1/4 \text{tr}(\Lambda)^2 \Rightarrow$  instable or stable spiral (for  $\text{tr}(\Lambda) = 0$ : marginally stable limit cycle.)



#### Four simple examples :

##### (a) Decay and Expansion

$$\frac{dx}{dt} = x \quad (360)$$

$$\frac{dy}{dt} = -y \quad (361)$$

Nullclines:  $x = 0, y = 0 \implies$  saddle point (check:  $\det = -1$ )

##### (b) Nonlinear Decay and Expansion

$$\frac{dx}{dt} = x \quad (362)$$

$$\frac{dy}{dt} = -y + x^2 \quad (363)$$

Nullclines:  $x = 0, y = x^2 \implies$  saddle point (check:  $\det = -1$ )

##### (c) Rotation

$$\frac{dx}{dt} = y \quad (364)$$

$$\frac{dy}{dt} = -x \quad (365)$$

Nullclines:  $y = 0, x = 0 \implies$  projections of gradients on nullclines are zero, thus marginally stable states!!! (check:  $\det = 1, \text{tr} = 0$ )

$\implies$  not fixed points, but limit cycles!

##### (d) Rotation with self-interaction, e.g. coupled excitatory-inhibitory neurons, with $0 < \epsilon < 1$ :

$$\frac{dx}{dt} = \epsilon x - y \quad (366)$$

$$\frac{dy}{dt} = x - \epsilon y \quad (367)$$

Nullclines:  $y = \epsilon x, x = \epsilon y \implies$  rotation, marginally stable limit cycle! (check:  $\det = 1 - \epsilon^2, \text{tr} = 0$ )

$\implies$  for  $\epsilon > 1$ , saddle points!

## 13.2 Differential equations

In Computational Neurosciences, differential equations are often used to describe the dynamics (i.e. the temporal evolution) of a variable in a biophysical system (i.e. the membrane voltage of a neuron).

In more general terms, „An ordinary differential equation (ODE) is an equation containing an unknown function of one real or complex variable  $t$ , its derivatives, and some given functions of  $t$ . The unknown function is generally represented by a variable (often denoted  $x$ ), which, therefore, depends on  $t$ . Thus  $t$  is often called the independent variable of the equation. The term "ordinary" is used in contrast with the term partial differential equation, which may be with respect to more than one independent variable.“(cited from Wikipedia, have a look at that site if you need more information!). Note that  $t$  not necessarily describes time.

For simplicity, let us consider first-order, ordinary differential equations (DEQs) of the form:

$$\frac{dx}{dt} = f(x, t). \quad (368)$$

Differential equations are different from equations we encountered previously. Usually, we considered given functions  $x(t)$  defined on an interval  $[a, b]$  and were interested, for example, in computing their derivative  $dx/dt$  in  $[a, b]$ , or their integral  $\int_a^b x(t)dt$ . Now the function  $x(t)$  is unknown, but their derivative  $dx/dt$  is specified for all values of  $x$  and  $t$ . Given an **initial condition** or **starting value**  $x(t_0) = x_0$ , our goal is to find  $x(t)$  by solving the DEQ starting from the initial condition.

Let us consider a simple example: We have a reservoir which is at time  $t_0$  filled with water up to the level  $x_0$ , and there is an outlet at the bottom of the reservoir through which the water flows with a rate proportional to the current level  $x$ . How does the water level changes with time  $t$ ?

The differential equation for this process reads

$$\frac{dx}{dt} = -cx, \quad (369)$$

where  $c$  is a positive proportionality constant. This DEQ can be solved graphically by starting at  $x(t_0 = 0) = x_0$ , computing the r.h.s. of the DEQ, making a small step in direction of the slope of  $x(t)$ , and repeat this procedure over and over.

In this simple case, we can also solve the DEQ analytically by performing a **separation of**

**variables** and an elementary integration,

$$\begin{aligned}
 \frac{dx}{dt} &= -cx \\
 \frac{dx}{x} &= -cdt \\
 \ln(x) &= -ct + c_0 \\
 x(t) &= \exp(-ct) \exp(c_0) \\
 x(t) &= x_0 \exp(-ct).
 \end{aligned} \tag{370}$$

Hereby, the integration constant  $\exp(c_0)$  was easily found by inserting the initial condition,  $x(0) = x_0 = \exp(c_0)$ . This solution tells us that the water reservoir empties with an exponential decay towards the **steady state**  $x(t \rightarrow \infty) = 0$ .

Let us now consider a slightly more involved problem – we add a faucet to the reservoir which adds water with a **constant** inflow rate  $d$ . The modified differential equation now reads

$$\frac{dx}{dt} = -cx + d. \tag{371}$$

It is easy to imagine what happens: if the outflow is smaller than the inflow, the water level will rise. If the outflow is larger than the inflow, the water will fall. Since the outflow depends on the current water level  $x$  while the inflow is constant, there will be a point where in- and outflow are in balance. Actually, this point or **steady-state solution** can be computed **without solving the DEQ** by just searching for a value of  $x$  for which there would be no change in  $x$ . No change means that the slope (i.e., the temporal change  $dx/dt$ ) is zero. By inserting this condition into the DEQ we obtain

$$x_{steady} = \frac{d}{c}. \tag{372}$$

But now let us solve the differential equation. This time we are **guessing** a solution and then check if this solution (with appropriate parameters) would be correct. In this case, we will try  $x(t) = A \exp(-ct) + B$  and substitute this equation into the DEQ:

$$-Ac \exp(-ct) = -Ac \exp(-ct) - Bc + d. \tag{373}$$

By comparing both sides of this equation, we find that indeed our guess is a correct solution if  $B = d/c$ . By using the initial condition from above to also find  $A = x_0 - d/c$  we arrive at the solution

$$x(t) = (x_0 - d/c) \exp(-ct) + d/c. \tag{374}$$

If  $t \rightarrow \infty$ , we confirm that the steady state indeed is  $x_{steady} = d/c$ .

Here we have used two common methods to solve DEQs analytically: separation of variables and guessing. There are a few more specialized methods for obtaining an analytical solution, but, since DEQs can be nonlinear and very complex, the most commonly used method to solve them is **numerical integration**. Numerical integration of the DEQ is similar to the graphical solution outlined above, and the simplest solver is the so-called 'Euler-scheme'. Of course, more precise and more specialized numerical methods exist, too.

#### Further remarks:

- a) The integrate-and-fire neuron was described by the differential equation (130)

$$\tau \frac{dV}{dt} = -V + V_{rest} + RI(t). \quad (375)$$

which is for a constant  $I$  equivalent to the differential equation (371) by means of mapping the parameters  $c = 1/\tau$  and  $d = (V_{rest} + RI)/\tau$ . Thus its solution according to equation (374) is given by:

$$V(t) = V_{rest} + RI + (V_0 - V_{rest} - RI) \exp\left(-\frac{t}{\tau}\right). \quad (376)$$

- b) If a DEQ is written in discrete time  $t_i$ ,  $i = 1, 2, \dots$  with finite step width  $\Delta t$ , the DEQ stated above becomes a discrete, iterative update equation  
 $x(t_{i+1}) = x(t_i) + \Delta t f(x(t_i), t_i)$ .
- c) DEQs can be nonlinear, such as  $dx/dt = x^2 + \sin(cx) + \exp(-x^3)$ . While analytically typically not solvable, the numerical solution can be obtained in the same manner as outlined, and needs only changing the r.h.s. of the DEQ (i.e. change one line in your program code!).
- d) In complex systems such as the brain, one has to solve many DEQs in parallel - for example, one equation for every neuron you want to simulate. Numerically, this is also easy to do - just copy your code for solving **one** DEQ  $N$  times, or put it into a loop, thus integrating  $N$  differential equations **in parallel**.
- e) If you have a DEQ of order  $M$ , such as  $d^2x/dt^2 = -cx$ , simply rewrite this DEQ as a system of  $M$  DEQs of order 1 (this is always possible and there's a recipe for that!) and solve them like described above.

**Bottom line: If you can numerically integrate one linear, ordinary differential equation of first order you can also integrate nonlinear systems of differential equations of arbitrary order – without learning new stuff!**