# Synaptic reliability

*Balazs B Ujfalussy*

*06/11/2017*

This is a demo for illustrating the response of integrate and fire neurons to synaptic input with different release probabilities. We have a single postsynaptic neuron, and $n = \{64, 16384\}$ presynaptic cells, all firing with $r = 2$ Hz and having release probability $p = 0.2$. These values are similar to the values found in the hippocampus. The strength of the synapses is scaled with the number of neurons, so the mean synaptic drive of our postsynaptic cell is kept constant.

We first simulate the net presynaptic spike train.

```
source('../Biophys/IF_sim.R', chdir=T)
```

```
p.release <- 0.2 # set the release probability
n.cells <- 2^seq(6,14) # number of cells is changed between 64 and 16000

n <- n.cells[4]
rate.cell <- 2 # Hz - mean firing rate of a single neuron
rate.population <- rate.cell * n # population firing rate
dt <- 1/1000 # s, simulation time step
Tmax <- 1 # s, duration of simulation
L <- Tmax / dt
t <- seq(dt, Tmax, by=dt)
# plot(t, sin(t*(2*pi) * 10)+1, t='l')
# the true pre pattern, sinusoidally modulated Poisson spike train
spikes <- rpois(L, rate.population*dt * (sin(t*(2*pi) * 10)+1))
```

Next, set up the synaptic filter - a simple exponential kernel to model the synaptic potentials

```
## we filter the spikes to get the EPSPs - this is not a conductance based synapse!
tau <- 5 # ms, synaptic time constant
Nfilt <- 50 # length of the synaptif filter
t.filt <- seq(1, Nfilt)
filt.syn <- exp(-t.filt/tau) # simple exponential synaptic filters - rise is infinitely fast
filt.syn <- filt.syn * 300000 / sum(filt.syn) / n ## filter integralis proportional to 1/n
## 1/300000: scale factor to convert input to nano-Amper
```

Finally we simulate the stochastic synaptic transmission (using the R function *rbinom()*), and generate the postsynaptic response for 10 different trials using the same presynaptic spike train. The postsynaptic cell's firing threshold is set to suppress spiking so that we can focus on the variance of the membrane potential.

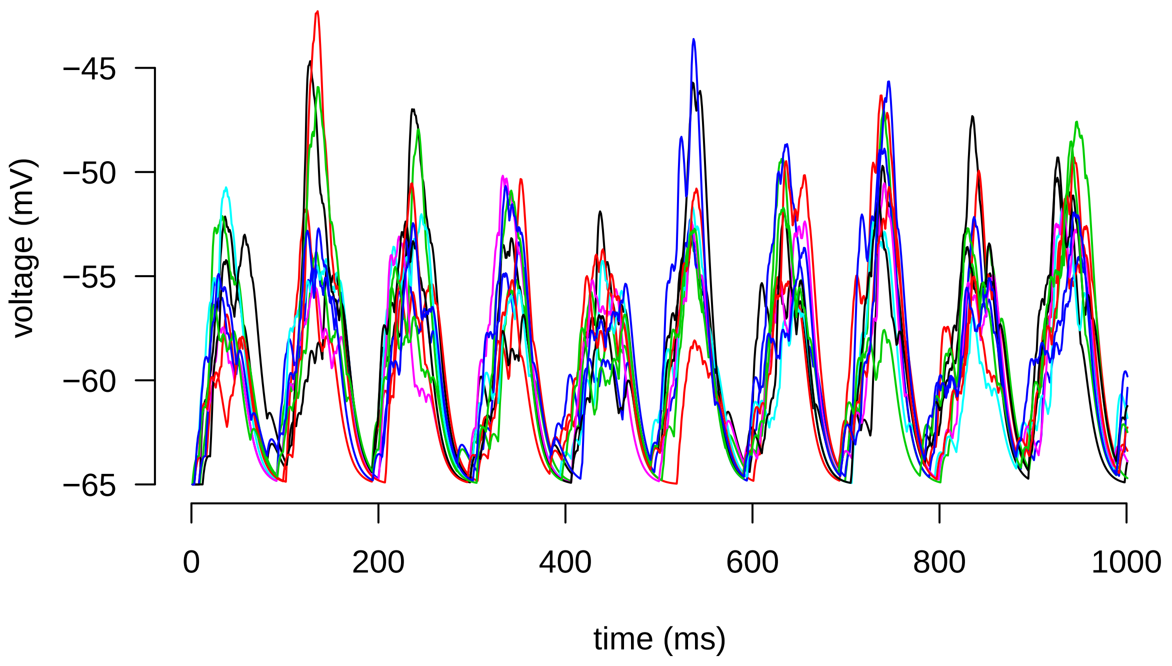
```
Nrep <- 10 # we will simulate 10 repetitions
vpost <- matrix(NA, L+1, Nrep) # 10 postsynaptic response

for (irep in 1:Nrep){
  release <- rbinom(L, spikes, p.release) # release events

  release <- c(rep(0, Nfilt), release)
  input <- filter(release, filt.syn, method='convolution')
  input <- input[(Nfilt/2):(L+Nfilt/2)]

  v.IF <- sim.IF(I=input, v.rest=-65, Rm=0.045, tau=10, v.threshold=-10, v.reset=-55)
  vpost[,irep] <- v.IF
}

sds <- apply(vpost, 1, sd)
matplot(vpost, t="l", lty=1, axes=F, xlab="time (ms)", ylab="voltage (mV)"); axis(1); axis(2, las=2)
```



## Homework

The task is to calculate how the variance of the membrane potential (across trials) depends on the number of presynaptic neurons. In particular, we are interested in the question whether increasing the release probability would lead to more accurate computations in neuronal circuits.

- Calculate the variance of the postsynaptic membrane potential accross trials! Average it along the 1s long trace. [3p]
- Change the number of presynaptic neurons (between 64 and 16000), and calculate the variance again! [2p]
- Plot the variance in the function ofthe number of presynaptic neurons. If you can, derive a functional relationship between the two quantities. [4p]
- What is the number of presynaptic neurons for a typical hippocampal CA3 pyramidal cell? What is the variability there? [2p]
- How the variance would change if we increased the release probability with $\epsilon << 1$? [3p]
- From computational and coding purposes, is it worth increasing the reliability of the synapses? Why? [3p]
- How the results change if you also consider inhibition? As a first approximation assume that inhibitory neurons also have low release probability. [3p]