filteringAssignment_blank

June 1, 2019

1 Assignment 1: filtering of deterministic signals

This tutorial investigates deterministic signals and the behavior of filters.

Important: *Please provide clear and details explanations for you answers (not only code but text/equations showing your reasoning to answer the questions). Details response can either be typed or handwritten on a document that you could scan.*

To do this assignement we need a Python 3 distribution with the following libraries:

```
In [74]: # import numpy for array manipulation and fft
         import numpy as np
         #import classical constant pi form numpy to avoid namespace
         from numpy import pi
         # import filtering functionalities from scipy.signal
         from scipy.signal import lfilter, filtfilt, butter, freqs, freqz
         # import solver for differential equations
         from scipy.integrate import odeint, ode
         # plotting functionalities
         import matplotlib.pyplot as plt
```

If you want to give back the assignement as a jupyter notebook, the following command triggers inline plot.

```
In [10]: %matplotlib inline
```

1.0.1 Basic representation of signals

This paragraph is just a refresher of conventions that were already used during the tutorial.

We will use a numerical representation of signals, we will assume the following setting:

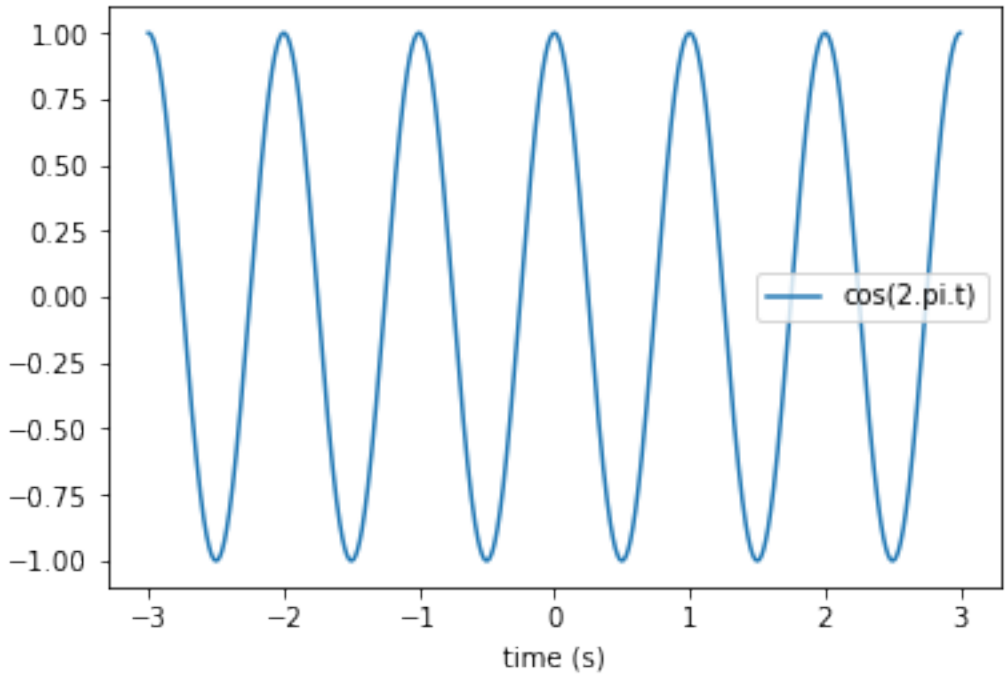
- signals are sampled at sampling period dt seconds
- as a consequence, the sampling frequency is F is $1/dt$ Hz

Here is a basic example of how to plot the representation of a continuous time signal ($\cos(2\pi t)$) by its sampled version in numpy:

```
In [11]: from numpy import pi
         # define sampling frequency
         dt=.001;
```

```
# define time axis
t = np.arange(-3,3,dt)
# sample a continuous function, e.g. 1-periodic cosine at these time point
x = np.cos(2*pi*t)
plt.plot(t,x)
plt.xlabel('time (s)')
plt.legend(['cos(2.pi.t)'])
```

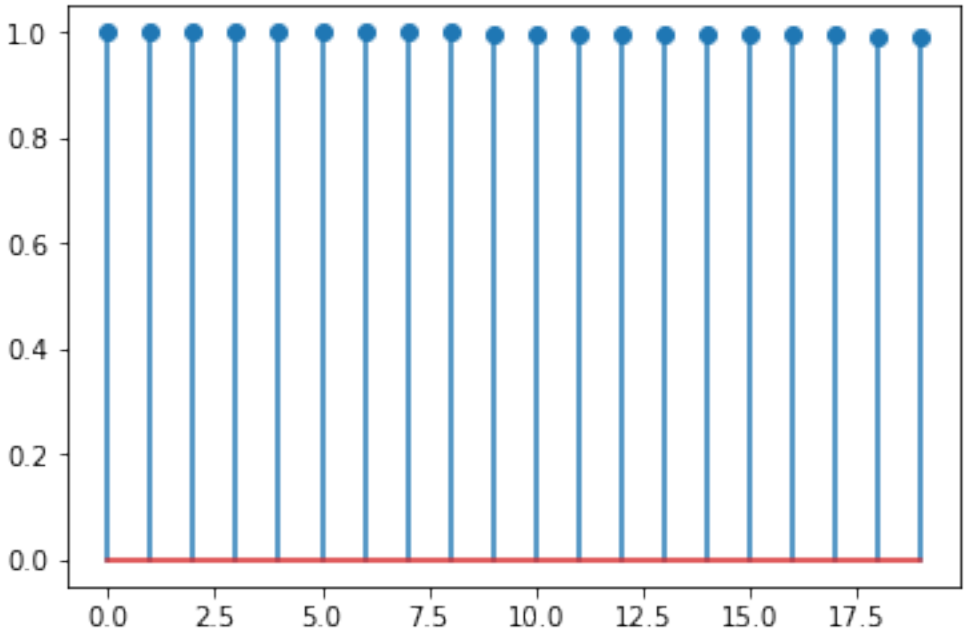
```
Out[11]: <matplotlib.legend.Legend at 0x26ddb9d4320>
```



If we want to plot the discretized signal for what it is, i.e. a finite sequence of numbers, we can use stem, however this is easy to visualize only for few samples

```
In [13]: plt.stem(x[0:20])
```

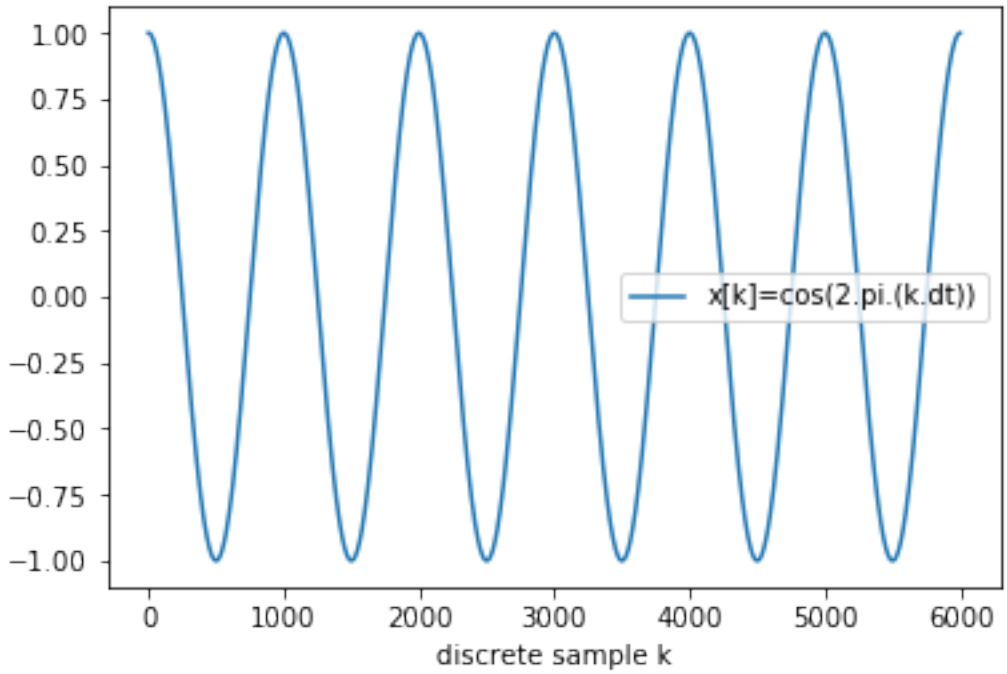
```
Out[13]: <Container object of 3 artists>
```



Alternatively we can just use the plot function, and only the x-axis changes

```
In [14]: plt.plot(x)
         plt.xlabel('discrete sample k')
         plt.legend(['x[k]=cos(2.pi.(k.dt))'])

Out[14]: <matplotlib.legend.Legend at 0x26de8ce8b38>
```



1.1 Modeling of post-synaptic potentials (PSP)

We will study a linear model of the post-synaptic potential of a neuronal membrane. The pre-synaptic input to this neuron will be denoted $x(t)$, and the output membrane potential $y(t)$.

We use a first order differential equation to model the input output relationship. This relies neglecting many non-linearities, and assuming $x(t)$ directly reflects the post-synaptic current flowing through the membrane with capacitance C (due to charge accumulation on both sides of the membrane) and resistance R (modeling the leak currents flowing through various ion channels).

This results in the differential equation:

$$C\frac{dy}{dt}(t) + \frac{1}{R}y(t) = x(t)$$

To simplify notation we will assume $R = 1$ such that the above equation is only controlled by the membrane time constant $\tau = RC$ and can be rewritten

$$\tau\frac{dy}{dt}(t) + y(t) = x(t)$$

2 Part 1: analog filter

2.1 Simulation with ODE solver

Such ordinary differential equation (ODE) can be simulated on a computer using scipy.integrate.ode, which is a wrapper for solving initial value problems of dynamical systems. To apply such routine, we put the equation in the form

$$\frac{dy}{dt}(t) = \frac{1}{\tau}\left(x(t) - y(t)\right) = f_x(y, t)$$

such that we can estimate the response to different inputs $x(t)$ by redefining the function f_x. We define this function with $\tau = 20ms$, such that we can pass an arbitrary function of time for x as third argument:

```
In [110]: tau = .02
          def f_x(t,y,x):
              return 1/tau*(x(t)-y)
```

In addition, to the differential equation, solving for y requires providing an initial value y_0 such that $y(t_0) = y_0$ for a given initial time point t_0. In order to unambiguously define the output of the system without further consideration for this initial value, we will always assume $y_0 = 0$ and choose t_0 such that $x(t) = 0$ for all $t < t_0$. Under such assumption, the output of the system is a signal $y(t)$ such that $y(t) = 0$ for all $t < t_0$, and for $t \geq t_0$, $y(t)$ is provided by the solution of the initial value problem defined above.

We will take for granted that the output y can be computed for time axis t using the *filterOutput* function defined below:

```
In [175]: def filterOutput(t,x):
              '''
              computes output values for the synapse model
              inputs:
                  t - a 1 dimensional numpy array of time points at which values sh
                  x - a function take one float input argument (the time)
              outputs:
                  y - a 1 diensional numpy array of values of y(t) for each input t
              '''
              r = ode(f_x,None)
              r.set_integrator('dopri5')
              r.set_initial_value(0,t[0])
              r.set_f_params(x)
              y = []
              for k in range(t.shape[0]):
                  y.append(r.integrate(r.t+dt))
              return np.array(y)
```

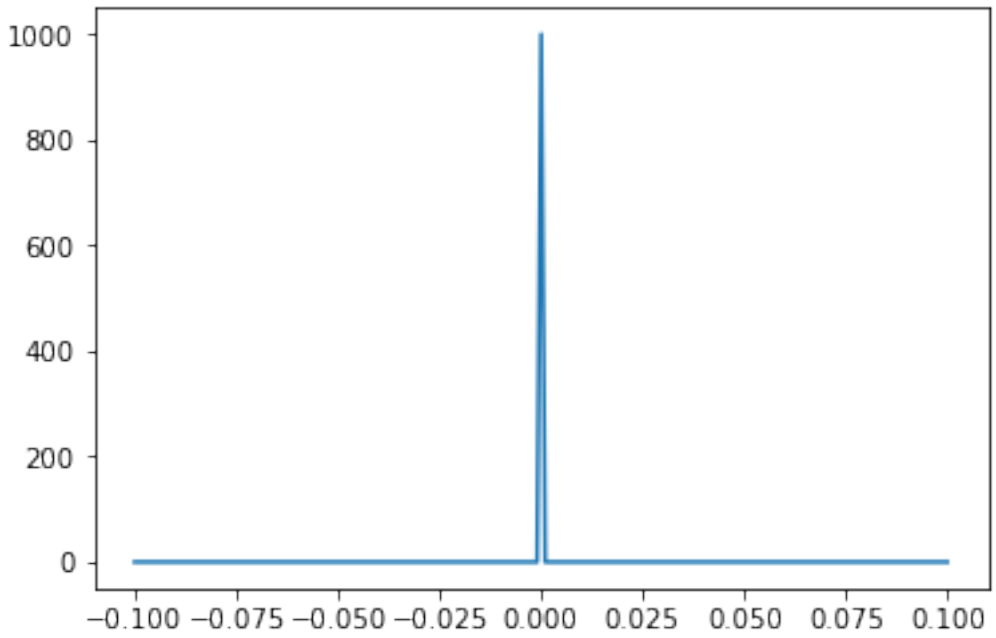
The post-synaptic response in real neurons is triggered afferent (pre-synaptic) action potentials. To model this, we define a triangular spike waveform parameterized by a spike width T (in seconds).

```
In [176]: def spikeWaveform(t,T):
              spkWave = (np.abs(t)<T).astype(np.float)*(1-np.abs(t)/T)/T
              return spkWave
```

We will assume a spike duration of $1ms$ and fix the spike(t) function accordingly

```
In [177]: T = .001
          dt=.0001
          spike = lambda t: spikeWaveform(t,T)
          t=np.arange(-.1,.1,dt)
          plt.plot(t,spike(t))

Out[177]: [<matplotlib.lines.Line2D at 0x26de42d1a90>]
```



Question 1:

- (a) Compute the response of the above defined synapse to spike(t), using the above function, for a time axis ranging from $-10ms$ to $100ms$, with a sampling period $dt = 0.1ms$ (unless otherwise stated, dt takes this value for all other questions).
- (b) From the lecture, find the analytical expression of the impulse response of the system with input x and output y, plot this response.
- (c) Compare plots in (a) and (b), and based on that argue which well know distribution $spike(t)$ approximates in the context of this simulation.
- (d) We define the *current clamp* experiments as setting $x(t) = 1$ (from an original value of 0 for $t < 0$) for the duration $0 \leq t \leq D$ with $D = 100ms$, before setting back $x(t) = 0$ for $t > D$. Compute and plot the output y(t) for such experiment for t ranging from $-10ms$ to $200ms$.

In [ ]:

In [ ]:

We now study the frequency properties of this system.
Question 2:

- (a) Compute the frequency response (Fourier transform of the impulse response) using the routine scipy.signal.freqs (check the documentation for the appropriate definition of the arguments), for a frequency axis ranging from 1Hz to 1000Hz. Note that the angular frequency ω corresponds to $2\pi f$, where f is the freqency in Hz.

- (b) From the lecture retrieve the analytical expression of the frequency response
- (c) Plot and compare the modulus of both frequency responses obtained in (a) and (b)
- (d) Compute the output of the system for input $\cos(2\pi f_0 t)$ for $f_0 = 1, 10, 100, 1000Hz$, for a time ranging from 0 to $3s$, estimate the amplitude of the response in each case (use $(max(y) - min(y))/2$ over e.g. the last period of the computed output). Which aspect of the frequency response this amplitude should reflect? Check consistency of the results with (c).

```
In [ ]:
```

```
In [ ]:
```

3 Part 2: Digital filters

We will now approximate the behavior of the above continuous time system based on the discretization of the impulse response.

3.1 Finite impulse response filter

We want to define an discrete time LTI system based on a finite length discretization of continuous time impulse response.

Question 3:

- (a) Use question 1 to create a sequence h_d of N values resulting from the sampling of the empirically estimated impulse response $h(t)$ (i.e. the response to a spike(t)) of the continuous time system at times $t = 0, dt, 2dt, ..., (N-1)dt$ with N =200 and dt=.0001
- (b) Consider the discrete time LTI system with impulse response $g[k] = \begin{cases} h_d[k] & \text{, if } \quad k \in [0 \dots N-1] \\ 0 & \text{, otherwise.} \end{cases}$. Check the lecture and the documentation of the function scipy.signal.lfilter to define the routine that computes the M values of the output of this system given M values of the input (all values taken by inputs and outputs before those are assumed to be zero).
- (c) The above discretized system approximates the contiuous time system of questions 1-2 by replacing continuous time convolution by discrete time convolution. Using approximation of integrals based on rectangles of width dt (as done in the lectures), explain how the output of the discretized system can be rescaled to approximately match the range of values achieved by the continuous time system.
- (d) Use the routine defined in (b) to compute a discretize response for the current clamp experiment of question 1(d). For that you will use the response to the discretized input sampled at dt=.001. Compare the discretized response to the response computed in question 1(d).
- (e) Use the routine *scipy.signal.freqz* to compute the frequency response of the discrete time LTI system defined above. Compare this response to the response of the continuous

time system computed in questionn 2, by converting normalized frequencies to physical frequencies (in Hz) by multiplying them by the sampling frequency $1/dt$. Note again that the frequency axis of freqz is given in pulsation (2π times normalized frequency)

- (f) Increase N to observe how the discretized system response gets closer to the original continuous time system (reproduce c-d plots with selected values of N.

In [ ]:

In [ ]:

3.2 Infinite impluse response filter

Based on lectures and the above question 1, we know that for our synpase system, if we discretize the analytic expression of the continuous time impulse response at multiples of sampling period dt, the resulting sequence takes the form

$$h_d[k] = \begin{cases} \alpha\rho^k\,, & k \geq 0 \\ 0\,, & \text{otherwise} \end{cases} \tag{1}$$

We use this property to define an discrete time approximation of the system with infinite length response. This implies that the output computation will rely on a difference equation with a feedback term involving the past output values.

Queston 4:

- (a) Give the expression of α and ρ as a function of τ (refer to the lecture/homeworks if needed, but this is immediate using question 1).
- (b) Based on developments shown in the lecture, show that such an h_d is the impulse response of the difference equation (with input x and output y)

$$y[n+1] = \rho y[n] + \alpha x[n]$$

- (c) Use the lfilter function to implement this filter for the numerical values of our problem.
- (d) Reproduce the questions 3.c-e with this new LTI sytem.
- (e) Check how the approximation deteroriates (current clamp response and frequency response) when we use a coarser sampling period (dt=5ms,10ms).

In [ ]:

In [ ]: