14/15 points

Exercise Sheet 3: Head Modeling & Non-linear dynamics

In [1]:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import signal as signal
def logistic(h, a=1):
    return 1/(1+np.exp(-a*h))
```

In [2]:

```
import bci_minitoolbox as bci
from mpl_toolkits.mplot3d import Axes3D
```

Task 1: BEM vs infinite solution (4 points)

5/5

The file 'cortex2dsm.npy', lets call it again L is similar to the leadfields of the last sheet but this time mapping to the cortex surface. Again, you have to select a dipole and the dipolar moments q. The voltage is then calculated by $v = L_i \cdot q$.

Tasks:

a) Chose an arbitrary dipole and a dipolar moment yourself and simulate the field using the BEM head model.

b) Then, calculate the anayltical solution for an infinite homogeneous current dipole at the cortex surface vertices (positions) for the same source location and dipolar moment. Source conductivity is $\sigma_1 = 0.201$. You can find the source position r_q for the dipole in 'gridpos.npy' which has the same indexing as the leadfield.

c) Plot the distribution of the potential on the brain surface using the *plot_trisurf* function similar as in task 7 sheet 2 in combination with indiviudally colored triangles.

c) Also, use the simple *plot* function to plot the voltage for both the analytic solution in infiinite space and the BEM soution on the cortex surface over the vertex index to get a closer look at individual values.

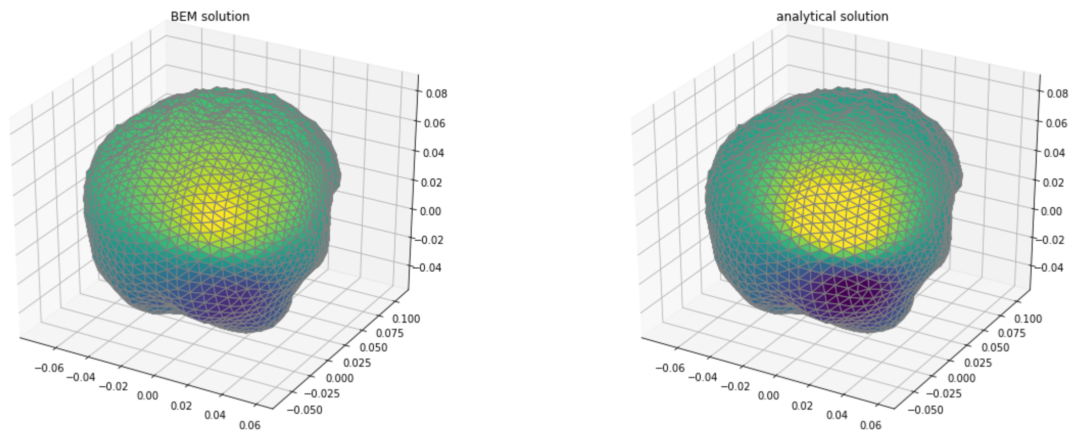
Are the solutions equal? Investigate and explain the difference.

In [3]:

```
def phi_dip(r, Q, r_0=None,sigma=0.201):
    r=np.array(r)
    Q=np.array(Q)
    if r_0 is not None:
        r_0=np.array(r_0)
        r=(r.T-r_0).T
    return 1/(4*np.pi*sigma)*(np.tensordot(Q,r,1))/np.power(np.linalg.norm(r,axis=0),3)

pos1=np.load('bnd4_pos1.npy')
tri1=np.load('bnd4_tri1.npy')

cortex2dsm=np.load('cortex2dsm.npy')
gridpos=np.load('gridpos.npy')
pos1=np.load('bnd4_pos1.npy')
tri1=np.load('bnd4_tri1.npy')
NoTri=tri1.shape[0]
NoPnt=pos1.shape[0]
```

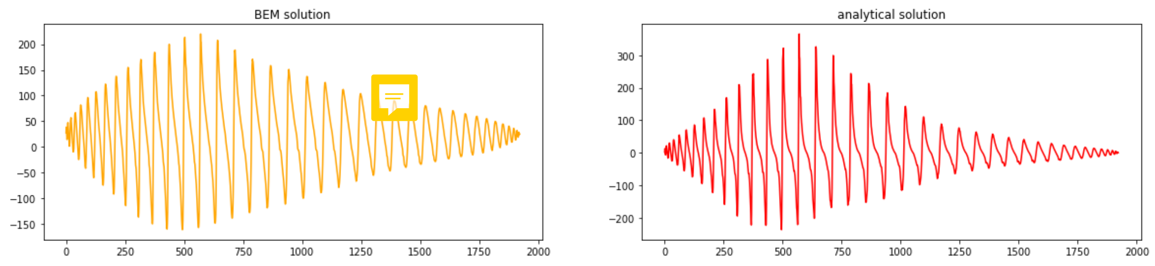
In [4]:

```
n=np.mean(tri1,1)
tri_average_n = n / n.max(axis=0)

x_axes1 = pos1[:,0]
y_axes1 = pos1[:,1]
z_axes1 = pos1[:,2]

source = 2460
p = np.array([0,0,1])

############## BEM ##############
phis = np.dot(cortex2dsm[:,source,:],p)

phis_avg = np.array([0]*tri1.shape[0])

for i,pos in enumerate(tri1):
    phis_avg[i] = (phis[pos[0]]+phis[pos[1]]+phis[pos[2]])/3

absmax=np.max(np.abs([phis_avg]))
mappy=plt.cm.ScalarMappable()
mappy.set_clim([-absmax,absmax])
mappy.set_array([])
#####
fig = plt.figure(figsize=(20,8))
ax = fig.add_subplot(121, projection='3d')
plt.title('BEM solution')
trisufplt=ax.plot_trisurf(x_axes1, y_axes1, z_axes1, triangles=tri1, color='yellow',edgecolor='g
ray')
trisufplt.set_facecolors(mappy.to_rgba(phis_avg))


############## analyltical ##############
phi_cortex = phi_dip(pos1.T,p,gridpos[source,:],sigma=0.201)

phi_cortex_avg = np.array([0]*tri1.shape[0])

for i,pos in enumerate(tri1):
    phi_cortex_avg[i] = (phi_cortex[pos[0]]+phi_cortex[pos[1]]+phi_cortex[pos[2]])/3

absmax=np.max(np.abs([phis_avg]))
mappy=plt.cm.ScalarMappable()
mappy.set_clim([-absmax,absmax])
mappy.set_array([])
#####
ax=fig.add_subplot(122, projection='3d')
plt.title('analytical solution')
trisufplt = ax.plot_trisurf(x_axes1, y_axes1, z_axes1, triangles=tri1, color='yellow',  edgecolo
r='gray')
trisufplt.set_facecolors(mappy.to_rgba(phi_cortex_avg))
```



In [5]:

```
fig = plt.figure(figsize=(20,4))
ax = fig.add_subplot(121)
ax.plot(phis,label='BEM',color='orange')
plt.title('BEM solution')
# ax.legend()
ax = fig.add_subplot(122)
ax.plot(phi_cortex,label='analytical',color='red')
plt.title('analytical solution')
# ax.legend()
plt.show()
```



d) Difference:

Quite similar in terms of shape, but we can find that the amplitude from BEM and Analytical solution is quite different, and the attenuation is different. When we compare the 3D plots, we can see that analytical solution present the dipole more strongly.

Task 2: The effect of the non-linear transfer function: simulation (4 points)

5/5

Tasks:

a) Write a function simulating a sine-wave $x(t) = a * sin(\omega t) + b = a * sin(2\pi f t) + b$. Simulate a time interval of 10s with a sampling rate of fs=200Hz and plot it over the first $0.5s$ with $a = 1$ for the amplitude, $b = 0$ for the offset and $f = 10Hz$ for the frequency.

b) Calculate the Power-Spectral-Density (PSD) of the signal and plot it over the frequency. For calculating the PSD, use the welch algorithm (scipy.signal.welch) and plot it using a logarithmic scale (10*log10(Pxx)). Clip the PSD at -80dB.

c) Use the logistic function $y(x) = \frac{1}{1+e^{-ax}}$ on the signal $x(t)$ as a non-linear time-invariant amplitude transformation. Do this for all possible combinations of the following amplitudes and offsets of the sine wave: a={0.5, 1, 10} and b={-2,-1,0, 1, 2}. Calculate and plot the PSD as in b).

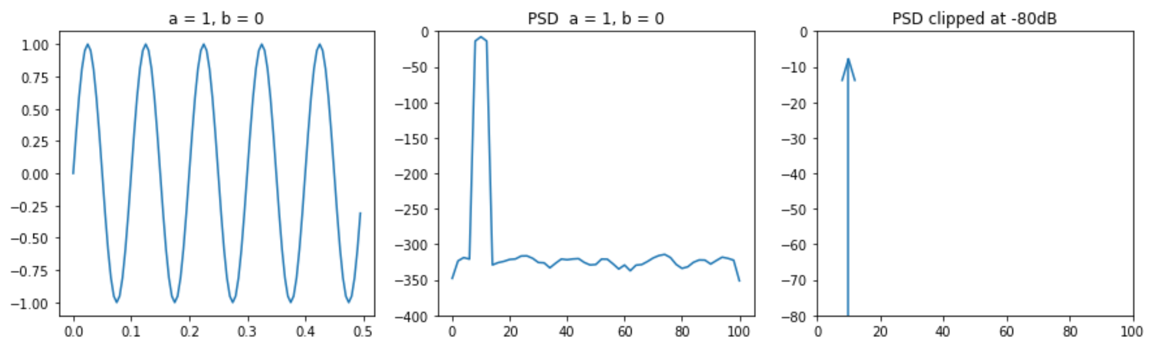
d) The signal is a pure sine wave, which as a fourier transform has a single peak in the frequency spectrum. Explain the differences to the PSD of the sine wave (b) and the effect of the non-linear transformation on the PSD (c).

In [6]:

```python
def sine_wave(t, amp = 1, freq = 1,  b = 0):
    return amp*np.sin(2*np.pi*freq*t)+b
```

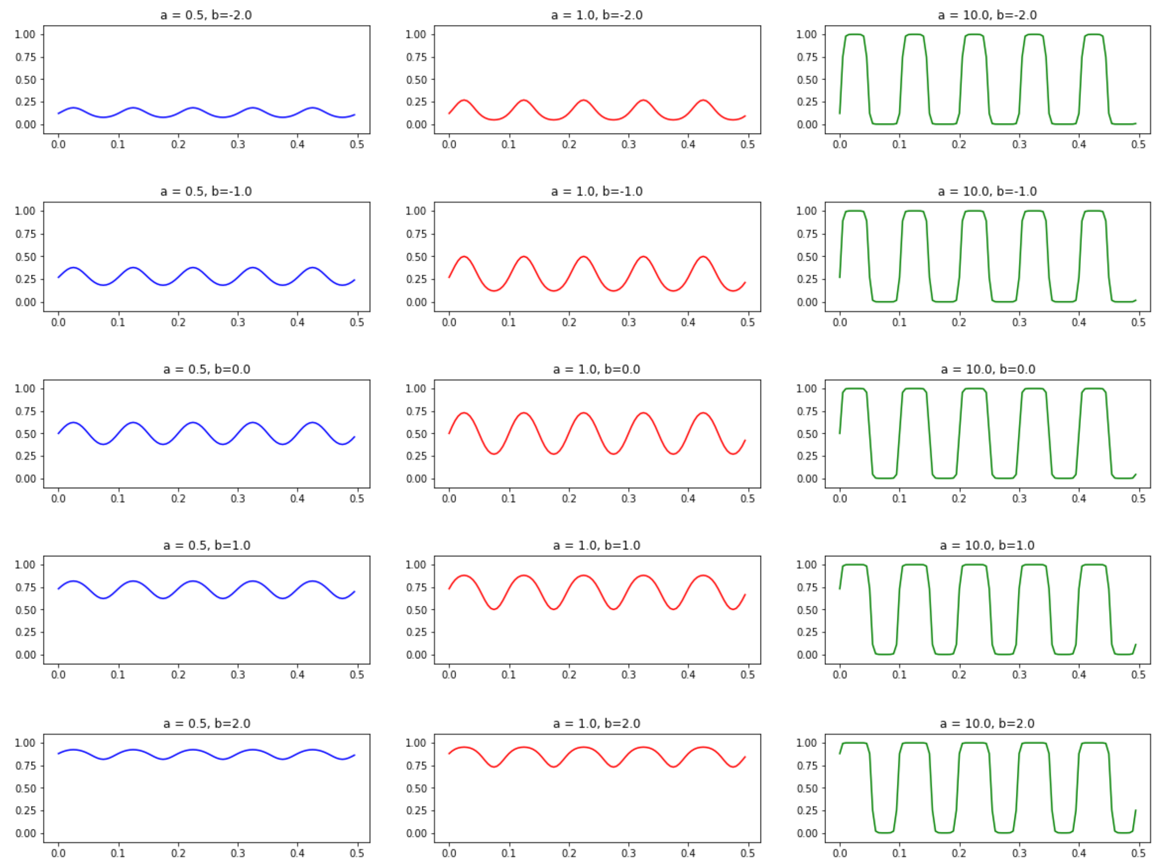
In [13]:

```
sampling_rate = 200
time = np.arange(0, 0.5, 1/sampling_rate)
sample = sine_wave(time, amp = 1, freq = 10, b = 0)

fig = plt.figure(figsize=(20,4))
ax = fig.add_subplot(141)
plt.title('a = 1, b = 0')
ax.plot(time,sample)

ax = fig.add_subplot(142)
plt.title('PSD  a = 1, b = 0')
f,pxx = signal.welch(sample, sampling_rate)
ax.set_ylim([-400,0])
ax.plot(f,10*np.log10(pxx))

ax = fig.add_subplot(143)
plt.title('PSD clipped at -80dB')
H = 10*np.log10(pxx)
H[H<-80] = float('NaN')
ax.set_xlim([f[0],f[-1]])
ax.plot(f,H)
ax.set_ylim([-80,0])
ax.plot([10,10],[-80, H[5]],color='C0')

# ax = fig.add_subplot(144)
# plt.title('sinus clipped at -80dB (inverse of ffp)')
# F = np.fft.ifft(pxx, n=len(time))
# ax.plot(time,F)
plt.show()
```



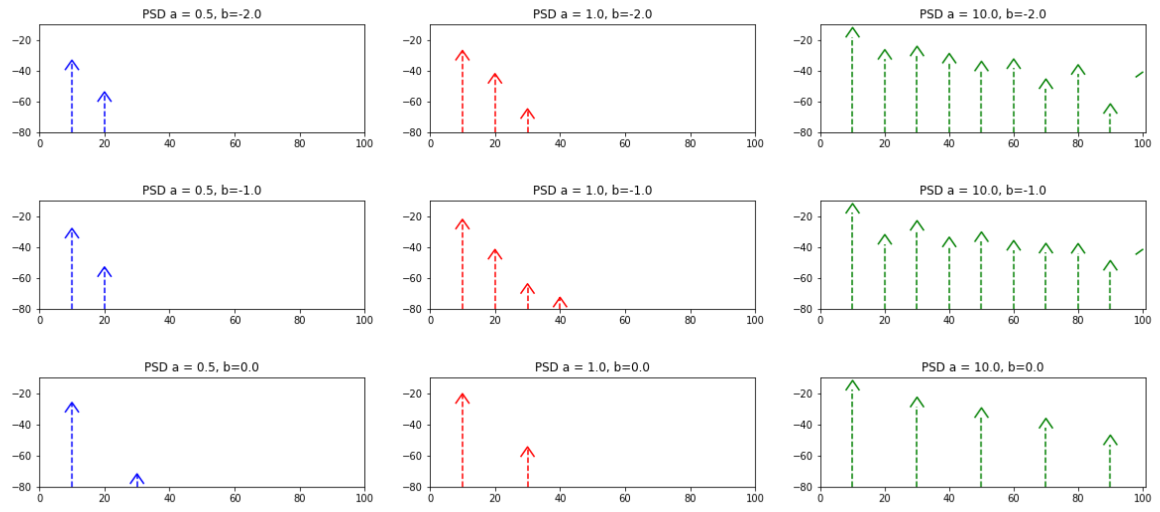
In [8]:

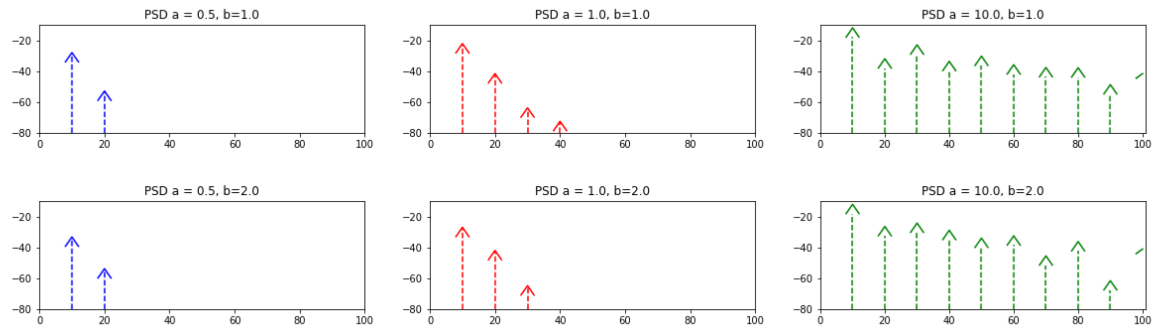
```
a_s = np.array([0.5, 1, 10])
b_s = np.array([-2, -1, 0, 1, 2])

for b in b_s:
    fig = plt.figure(figsize=(20,2))
    ####
    ax = fig.add_subplot(1,3,1)
    sample = sine_wave(time, amp = a_s[0], freq = 10, b = b)
    changed = logistic(sample)
    ax.set_ylim([-0.1,1.1])
    plt.title('a = %.1f, b=%.1f'%(a_s[0],b))
    ax.plot(time, changed, color='blue')
    ####
    ax = fig.add_subplot(1,3,2)
    sample = sine_wave(time, amp = a_s[1], freq = 10, b = b)
    changed = logistic(sample)
    ax.set_ylim([-0.1,1.1])
    plt.title('a = %.1f, b=%.1f'%(a_s[1],b))
    ax.plot(time,changed, color='red')
    ####
    ax = fig.add_subplot(1,3,3)
    sample = sine_wave(time, amp = a_s[2], freq = 10, b = b)
    changed = logistic(sample)
    ax.set_ylim([-0.1,1.1])
    plt.title('a = %.1f, b=%.1f'%(a_s[2],b))
    ax.plot(time,changed, color='green')

    plt.show()
```



In [9]:

```
a_s = np.array([0.5, 1, 10])
b_s = np.array([-2, -1, 0, 1, 2])

for b in b_s:
    fig = plt.figure(figsize=(20,2))
    ####
    sample = sine_wave(time, amp = a_s[0], freq = 10, b = b)
    changed = logistic(sample)
    f,pxx = signal.welch(changed, sampling_rate)
    ax = fig.add_subplot(1,3,1)
    plt.title('PSD a = %.1f, b=%.1f'%(a_s[0],b))
    pxx = 10*np.log10(pxx)
    pxx[pxx<-80] = float('NaN')
    ax.set_xlim([f[0],f[-1]])
    ax.set_ylim([-80,-10])
    ax.plot(f, pxx, color='blue', linestyle="-")
    for i in range(0,50,5):
        plt.plot([f[i],f[i]], [-80,pxx[i]], color='blue', linestyle="--")

    ####
    sample = sine_wave(time, amp = a_s[1], freq = 10, b = b)
    changed = logistic(sample)
    f,pxx = signal.welch(changed, sampling_rate)
    ax = fig.add_subplot(1,3,2)
    plt.title('PSD a = %.1f, b=%.1f'%(a_s[1],b))
    pxx = 10*np.log10(pxx)
    pxx[pxx<-80] = float('NaN')
    ax.set_xlim([f[0],f[-1]])
    ax.set_ylim([-80,-10])
    ax.plot(f, pxx, color='red', linestyle="-")
    for i in range(0,50,5):
        plt.plot([f[i],f[i]], [-80,pxx[i]], color='red', linestyle="--")

    ####
    sample = sine_wave(time, amp = a_s[2], freq = 10, b = b)
    changed = logistic(sample)
    f,pxx = signal.welch(changed, sampling_rate)
    ax = fig.add_subplot(1,3,3)
    plt.title('PSD a = %.1f, b=%.1f'%(a_s[2],b))
    pxx = 10*np.log10(pxx[1:])
    pxx[pxx<-80] = float('NaN')
    ax.set_xlim([f[0],101])
    ax.set_ylim([-80,-10])
    ax.plot(f[1:], pxx, color='green', linestyle="-")
    for i in range(0,50,5):
        plt.plot([f[i],f[i]], [-80,pxx[i]], color='green', linestyle="--")

    plt.show()
```





d)Explain the difference of PSD and PSD after non-linear transformation:

For pure sine-wave, the power spectral density is quite centralized and we can only observe one formant (peak). The energy is not spitting. When we implement non-linear transformation, the singal is not pure anymore, it is compound. We can observe more formants(peaks) in the diagram, meanwhile, the energy with in the wave is also spitting around. The amplitude a has strong influence on the PSD and how PSD is cliped and attenuated. b has influence on the number of peaks of each PSD.

Task 3: The effect of the non-linear transfer function: analytic & simulation (5 points)

4/5

With a little trick, the clipping of an oscillating signal like a cosine wave y(t) can be seen as a linear operation: the multiplication with a rectangular oscillation x(t) of same frequency and apropriate phase. We ignore the DC part in this exercise. This models the "on/off" effect of the clipping:

$z(t) = x(t)y(t)$

For the periodic rectangular time series defined within a period $T_0 = \frac{1}{f_0}$ (note that this is a little bit different than in the lecture):

$$x(t) = \begin{cases} 1 & \text{for } |t| \leq T_1 \\ 0 & \text{else} \end{cases}$$

the corresponding fourier transform (here one-sided) is based on the delta function:

$$\hat{x}(f) = \frac{T_1}{T_0}\delta(f) + \sum_{n=1}^{\infty} \frac{\sin^2(\pi n \frac{T_1}{T_0})}{\pi n}\delta(f - \frac{n}{T_0})$$

A cosine wave of frequency f_0 has the fourier transform:

$$\hat{y}(f) = 2\pi\delta(f - f_0)$$

In the theory of Fourier Transformation, a multiplicatiion in time domain corresponds to a convolution in the frequency:

$$z(t) = x(t)y(t) <=> z(f) = \hat{x}(f) * \hat{y}(f)$$

with the convolution being:

$$\hat{x}(f) * \hat{y}(f) = \int_{-\infty}^{\infty} \hat{x}(f - f')\hat{y}(f')df'$$

Task:

a) Calculate the Fourier Transform $\hat{z}(f)$ of the signal $z(t)$ analyitcally by convolving $\hat{x}(f)$ and $\hat{y}(f)$ in the frequency domain.

b) Implement the corresponding function for plotting and plot the amplitde spectrum for a clipped 10Hz. Set T_1 consequently to $\frac{1}{4}T_0$, $\frac{1}{2}T_0$ and $\frac{3}{4}T_0$. Plot the delta function using the matplot functiion stem. What are the corresponding clipping values?

c) Investigate the values at nf_0 and their dependency on the frequency. What is the effect of the clipping in the spectrum? What implications does the non-linear effect of clipping have on clipped sine waves (single frequency peaks) and white noise (spectrum is constant over frequency)?

*Hint: For the convolution, think about the special properties of the dirac delta function $\delta(x)$.

Solution to a):

$$\hat{z}(f) = \int_{-\infty}^{\infty} \hat{x}(f-f')\hat{y}(f')df'$$

$$= \int_{-\infty}^{\infty} (\frac{T_1}{T_0}\delta(f-f') + \sum_{n=1}^{\infty} \frac{\sin^2(\pi n \frac{T_1}{T_0})}{\pi n}\delta(f-f'-\frac{n}{T_0})) \cdot 2\pi\delta(f'+f_0)df'$$

$$= 2\pi(\int_{-\infty}^{\infty} \frac{T_1}{T_0}\delta(f-f')\delta(f'+f_0)df' + \int_{-\infty}^{\infty} \sum_{n=1}^{\infty} \frac{\sin^2(\pi n \frac{T_1}{T_0})}{\pi n}\delta(f-f'-\frac{n}{T_0})\delta(f'+f_0)df')$$

$$= 2\pi(\frac{T_1}{T_0}\delta(f+f_0) + \sum_{n=1}^{\infty} \frac{\sin^2(\pi n \frac{T_1}{T_0})}{\pi n}\delta(f+f_0-\frac{n}{T_0}))$$

$$= 2\pi\frac{T_1}{T_0}\delta(f+f_0) + \sum_{n=1}^{\infty} 2\frac{\sin^2(\pi n \frac{T_1}{T_0})}{n}\delta(f+f_0-\frac{n}{T_0})$$

because: $\int_{-\infty}^{\infty} f(t)\delta(t-T)dt = f(T)$

$\int_{-\infty}^{\infty} \delta(\xi - x)\delta(x-\eta)\mathrm{d}x = \delta(\xi-\eta)$

In [10]:

```
def z_fourier(fs,n,fr=1,fc=1):
    t0 = 1/fr
    t1 = n*t0
    freq = [fc]
    a = [2*np.pi*n+(2*(np.sin(np.pi*2*n))**2)/2]
    for m in range(3,12):
        freq.append(m/t0-fr)
        a.append((2*(np.sin(np.pi*m*n))**2)/m)
    return freq,a
```

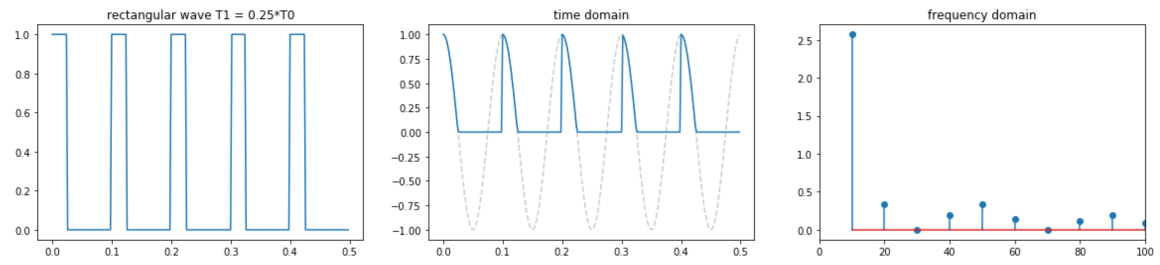
In [11]:

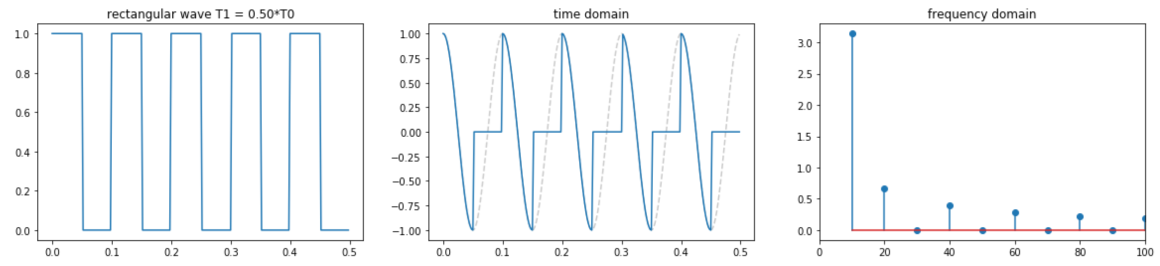
```
def rect_wave(t, f0, n):
    x = []
    t0 = 1/f0
    t1 = t0*n
    tx = t/t0 - t//t0
    for ty in tx:
        if ty <= n:
            x.append(1)
        else:
            x.append(0)
    return x
```

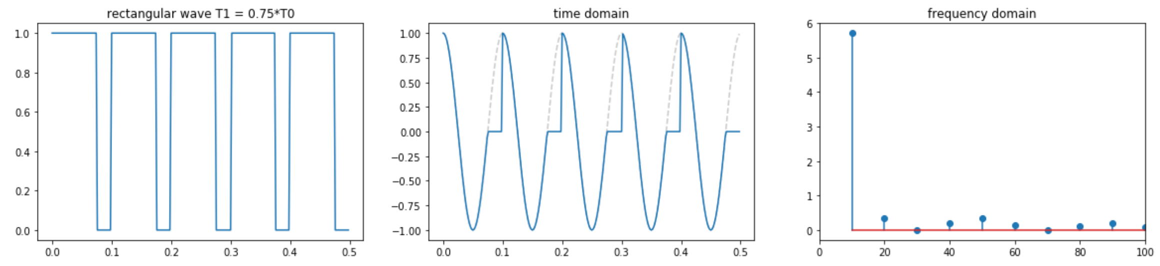
-0.5

In [12]:

```
sampling_rate = 500
time = np.arange(0,0.5, 1/sampling_rate) #T = 1/f0 = 0.1
fr = 10
fc = fr
cosine = np.cos(2*np.pi*fc*time) # y #Tc=1/fc

for n in [0.25, 0.5, 0.75]:
    rect = rect_wave(time,fr,n) # x

    fig = plt.figure(figsize=(20,4))

    ## rectangle wave
    ax = fig.add_subplot(131)
    plt.title('rectangular wave T1 = %.2f*T0'%n)
    ax.plot(time,rect)

    ## cos & clipped cos wave
    ax = fig.add_subplot(132)
    ax.plot(time,cosine,label='original cos', alpha = 0.4, color = 'C7', linestyle="--")
    ax.plot(time,rect*cosine,label='clipped', color = 'C0')
    plt.title('time domain')


    ## fourier
    ax = fig.add_subplot(133)
    freq = np.fft.fftfreq(len(time), d=1/sampling_rate) # freq = 0-100
    length = (len(freq)//2) # show only the positive side of freqence domain

    z2a,z2b = z_fourier(freq[:length],n,fr=fr,fc=fc)
    ax.stem(z2a,z2b)

    ax.set_xlim([0,100])
    plt.title('frequency domain')

    plt.show()
```







c) Investigate the values at $nf0$ and their dependency on the frequency. What is the effect of the clipping in the spectrum? What implications does the non-linear effect of clipping have on clipped sine waves (single frequency peaks) and white noise (spectrum is constant over frequency)?

-0.5

The original cosine wave has one peak at f0. The retangle wave, for example T1=0.5*T0, is a combination of multipule harmonic wave. So here we can see in the second frequency domain plot, when the cosine wave is clipped by the retangle wave, is a combination of them. And the energy at f0(original consine) is splited up into different frequency at k*f0.

In the thrid plot, we can see, when n from T1=n*T0 is bigger, the clipped harmonic wave can remain more its original shape, which means the freqency at f0 (the freqency from the original harmonic wave) is stronger than the rectangular wave. This means that the harmonic wave is not clipped too much. When n is smaller, the most part of harmonic wave is clipped, and the effect of f0 is smaller than a bigger n.

In [ ]: