Sheet

1.1

$X(f) = 0, \forall f \in [-W/2, W/2]$.
This means $X[f - kf_s] = 0 \forall f \notin [kf_s - f_s/2, +f_s/2]$. So if $k \neq l$, $X[f - kf_s]$ and $X[f - lf_s]$ shows the signal will not overlap.

This is because any signal over the bandwidth of $-W/2$ to $W/2$ causes the signals to overlap, which creates amplification that makes the signal less distinct.

To meet the frequency threshold $-W/2$ to $W/2$, we do a low pass filter on the Direc train representation.

$\tilde{x} = x_\delta * [f_s sinc(\pi f_s t)]$.
Forcing the bandlimit will prevent loss of information.

1.2

We first transform x into a signal x_f with $X_{f_s} = F(x_{f_s})$ where $X_{f_s}(f) = X(f) \sqcap f_s(f)$.
Like in 1.1 this becomes

$x_{f_s} = x * [f_s sinc(\pi f_s t)]$.

The frequency components between $-f_s/2$ and $f_s/2$ are retained because

1.3

We can approximate \sigma(t) with narrow pulses p(t).
Minimizing signal distortion requires you to use the actual modulated train pulse using $\sigma(t)$ and approximated modulated train pulse using $p(t)$,

$X_\delta(f) = \Sigma_{k=-\infty}^{\infty} X(f - kf_s)$

$X_p(f) = P(f)X_\delta(f) = P(f)\Sigma_{k=-\infty}^{\infty} X(f - kf_s)$.

$P(t)$ is the spectrum of $p(t)$
The reconstructed signal using

$\tilde{X}_\delta(f) = \sqcap_{f_s}(f)\Sigma_{k=-\infty}^{\infty} X(f - kf_s) = \sqcap_{f_s}(f)X(f)$, $\tilde{X}_p(f) = \sqcap_{f_s}(f)P(f)\Sigma_{k=-\infty}^{\infty} X(f - kf_s) = P(f) \sqcap_{f_s} (f)X(f)$,

The low pass filter eliminates all frequencies outside of $[-f_s/2, f_s/2]$ With $\tilde{X}_\delta(f)$ coincides with $\tilde{X}_p(f)$ there is no distortion results because the spectrum of $p(t)$ satisfies
$P(f) = 1, \forall f \in [-f_s/2, f_s/2]$ and also satisfies the property $p(t) = f_s sinc(\pi f_s t)$ with $P(f) = \sqcap_{f_s}(f)$.

2.1

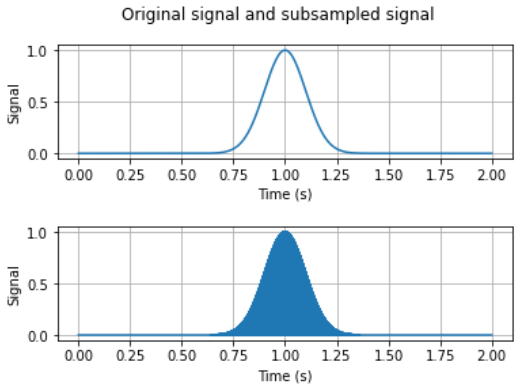
We can start with equation 11.
$x_\delta(n) = \sum_{m=-\infty}^{\infty} x_d(m\frac{\tau}{T_s})\delta(n - m\frac{\tau}{T_s})$
We can set the sampling time to be τ and and make the train of discrete deltas into $\sum_{m=-\infty}^{\infty} \delta(n - m)$
We then get:
$\tau \sum_{n=-\infty}^{\infty} e^{-j2\pi f n\tau} = \sum_{k=-\infty}^{\infty} \delta(f - kv)$
We can then compute the DTFT when the sampling time is T_s.
$X_c(f) = T_s \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \delta(n - m\tau/T_s)e^{-j2\pi f n\tau}$
We can then define b to be nT_s/τ.
Subbing in, we get:
$X_c(f) = T_s \sum_{b=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \delta(b - m\tau/T_s)e^{-j2\pi f b\tau}$
$T_s \sum_{b=-\infty}^{\infty} e^{-j2\pi f b\tau}$
$= \frac{T_s}{\tau} \sum_{k=-\infty}^{\infty} \delta(f - kv)$
Since we multiplied in the time domain, we can write this as:
$X_\delta(f) = \frac{T_s}{\tau} \sum_{k=0}^{r/T_s - 1} X(f - \frac{k}{t})$

```
import numpy as np
import cmath
import matplotlib.pyplot as plt
```

```
class dft():
    def __init__(self, x, fs, K=None):

        if (type(fs) != int) or (fs<=0):
            raise NameError('The frequency fs should be a positive integer.')
        if not isinstance(x, np. ndarray):
            raise NameError('The input signal x must be a numpy array.')
        if isinstance(x, np. ndarray):
            if x.ndim!=1:
                raise NameError('The input signal x must be a numpy vector array.')
        self.x=x
        self.fs=fs
        self.N=len(x)
        if K == None:
            K = len(self.x)

        if (type(K) != int) or (K <= 0) or (K < 0):
            raise NameError('K should be a positive integer.')
        self.K=K
        self.f=np.arange(self.K)*self.fs/self.K # (0:K-1) just creates a vector from 0 to K by steps of 1.
        self.f_c=np.arange(-np.ceil(K/2)+1,np.floor(self.K/2)+1)*self.fs/self.K
    def solve(self):
        X=np.fft.fft(self.x,self.K)/np.sqrt(self.N)
        X_c=np.roll(X,np.int(np.ceil(self.K/2-1)))
        return [self.f,X,self.f_c,X_c]

class idft():
    def __init__(self, X, fs, N, K=None):

        self.X=X
        self.fs=fs
        self.N=N
        self.K=K
        if self.K==None:
            self.K=int(len(X)/2)-1

    def solve_ifft(self):
        x=np.fft.ifft(self.X,self.N)*np.sqrt(self.N)

        Ts= 1/self.fs
        Treal= np.arange(self.N)*Ts

        return x, Treal

class gaussian_pulse():
    def __init__(self, mu, sigma, T, fs):
        self.N = np.int(np.floor( T * fs))
        self.t = np.arange(0, T, 1 / fs)
        self.sig = np.exp(-(self.t-mu)**2 / (2 * sigma**2))
        self.t = np.arange(0, T, 1 / fs)
```

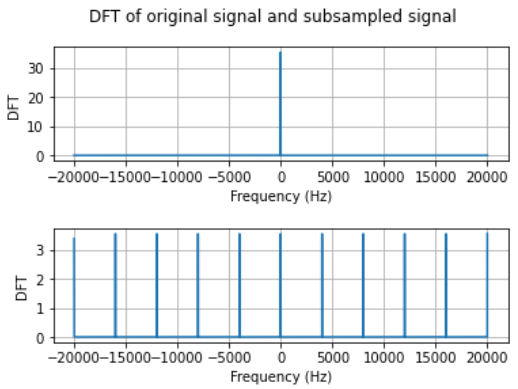
```
class subsample(object):

    def __init__(self, x, Ts, tau):

        self.x = x
        self.Ts = Ts
        self.fs = np.int(1/Ts)
        self.tau = tau
        self.fss = np.int(1/tau)
        self.N = len(x)

    def solve(self):

        step = np.int(self.tau/self.Ts)
        x_s = self.x[0::step]
        x_delta = np.zeros(self.N)
        x_delta[0::step] = x_s

        return x_s, x_delta

    def solve_filter(self):

        # Low-pass
        fmax = self.fss/2
        DFT = dft(self.x, self.fs)
        [_, _, f_c, X_c] = DFT.solve()
        index_min  = np.min(np.where(f_c >= -fmax)[0])
        index_max = np.max(np.where(f_c <= fmax)[0])
        X_band = np.zeros(self.N)
        X_band[index_min:index_max] = X_c[index_min:index_max]
        X_band = np.roll(X_band, np.int(np.floor(self.N/2+1)))
        iDFT = idft(X_band, self.fs, self.N)
        x_band, t = iDFT.solve_ifft()

        step = np.int(self.tau/self.Ts)
        x_s = x_band[0::step]
        x_delta = np.zeros(self.N)
        x_delta[0::step] = x_s

        return x_s, x_delta
```

```
def q_22(mu, sigma, fs, fss, T):

    gaussian_p = gaussian_pulse(mu, sigma, T, fs)
    x = gaussian_p.sig
    t = gaussian_p.t

    subsample_obj = subsample(x, 1/fs, 1/fss)
    x_s, x_delta = subsample_obj.solve()

    fig, axs = plt.subplots(2)
    axs[0].grid()
    axs[1].grid()
    fig.suptitle('Original signal and subsampled signal' )
    fig.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.6)
    axs[0].plot(t, x)
    axs[0].set_xlabel('Time (s)')
    axs[0].set_ylabel('Signal')
    axs[1].plot(t, x_delta)
    axs[1].set_xlabel('Time (s)')
    axs[1].set_ylabel('Signal')
    plt.savefig('signal_and_subsampled_time_'+ str(sigma) + '.png')

q_22(1, 0.1, 40000, 4000, 2)
```



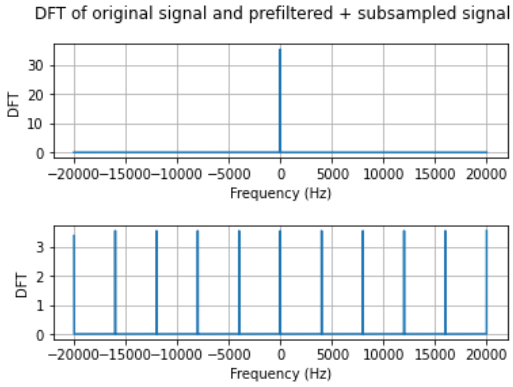

```
<ipython-input-8-a1b52b921199>:47: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.N = np.int(np.floor( T * fs))
<ipython-input-19-018ca49aab2d>:7: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fs = np.int(1/Ts)
<ipython-input-19-018ca49aab2d>:9: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fss = np.int(1/tau)
<ipython-input-19-018ca49aab2d>:14: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  step = np.int(self.tau/self.Ts)
```

```
def q_23(mu, sigma, fs, fss, T):
    gaussian_p = gaussian_pulse(mu, sigma, T, fs)
    x = gaussian_p.sig
    DFT_x = dft(x, fs)
    [_, _, freqs_c, X_c] = DFT_x.solve()

    subsample_obj = subsample(x, 1/fs, 1/fss)
    x_s, x_delta = subsample_obj.solve()
    DFT_x_delta = dft(x_delta, fs)
    [_, _, _, X_delta_c] = DFT_x_delta.solve()

    # Plot
    fig, axs = plt.subplots(2)
    axs[0].grid()
    axs[1].grid()
    fig.suptitle('DFT of original signal and subsampled signal' )
    fig.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.6)
    axs[0].plot(freqs_c, abs(X_c))
    axs[0].set_xlabel('Frequency (Hz)')
    axs[0].set_ylabel('DFT')
    axs[1].plot(freqs_c, abs(X_delta_c))
    axs[1].set_xlabel('Frequency (Hz)')
    axs[1].set_ylabel('DFT')
    plt.savefig('signal_and_subsampled_freq_'+ str(sigma) + '.png')

q_23(1, 0.1, 40000, 4000, 2)
```



```
<ipython-input-8-a1b52b921199>:47: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.N = np.int(np.floor( T * fs))
<ipython-input-8-a1b52b921199>:24: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  X_c=np.roll(X,np.int(np.ceil(self.K/2-1)))
<ipython-input-12-e84c7416aa06>:7: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fs = np.int(1/Ts)
<ipython-input-12-e84c7416aa06>:9: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fss = np.int(1/tau)
<ipython-input-12-e84c7416aa06>:14: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  step = np.int(self.tau/self.Ts)
```

```
def q_prefilter(mu, sigma, fs, fss, T):
    gaussian_p = gaussian_pulse(mu, sigma, T, fs)
    x = gaussian_p.sig
    subsample_obj = subsample(x, 1/fs, 1/fss)
    x_s, x_delta = subsample_obj.solve_filter()

    DFT_x = dft(x, fs)
    [_, _, freqs_c, X_c] = DFT_x.solve()

    DFT_x_delta = dft(x_delta, fs)
    [_, _, _, X_delta_c] = DFT_x_delta.solve()

    # Plot
    fig, axs = plt.subplots(2)
    axs[0].grid()
    axs[1].grid()
    fig.suptitle('DFT of original signal and prefiltered + subsampled signal' )
    fig.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.6)
    axs[0].plot(freqs_c, abs(X_c))
    axs[0].set_xlabel('Frequency (Hz)')
    axs[0].set_ylabel('DFT')
    axs[1].plot(freqs_c, abs(X_delta_c))
    axs[1].set_xlabel('Frequency (Hz)')
    axs[1].set_ylabel('DFT')
    plt.savefig('signal_and_prefiltered_freq_'+ str(sigma) + '.png')

q_prefilter(1, 0.1, 40000, 4000, 2)
```

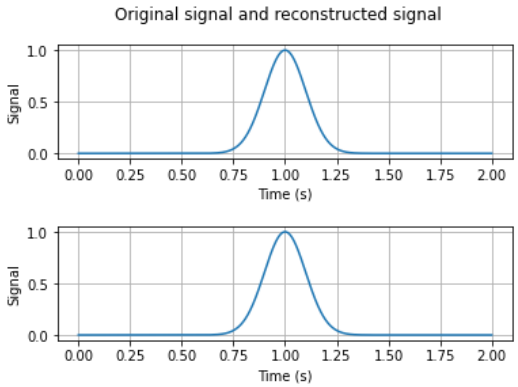



```
<ipython-input-8-a1b52b921199>:47: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.N = np.int(np.floor( T * fs))
<ipython-input-19-018ca49aab2d>:7: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fs = np.int(1/Ts)
<ipython-input-19-018ca49aab2d>:9: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fss = np.int(1/tau)
<ipython-input-8-a1b52b921199>:24: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  X_c=np.roll(X,np.int(np.ceil(self.K/2-1)))
<ipython-input-19-018ca49aab2d>:30: ComplexWarning: Casting complex values to real discards the imaginary part
  X_band[index_min:index_max] = X_c[index_min:index_max]
<ipython-input-19-018ca49aab2d>:31: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
```

```
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  X_band = np.roll(X_band, np.int(np.floor(self.N/2+1)))
<ipython-input-19-018ca49aab2d>:35: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  step = np.int(self.tau/self.Ts)
<ipython-input-19-018ca49aab2d>:38: ComplexWarning: Casting complex values to real discards the imaginary part
  x_delta[0::step] = x_s
```

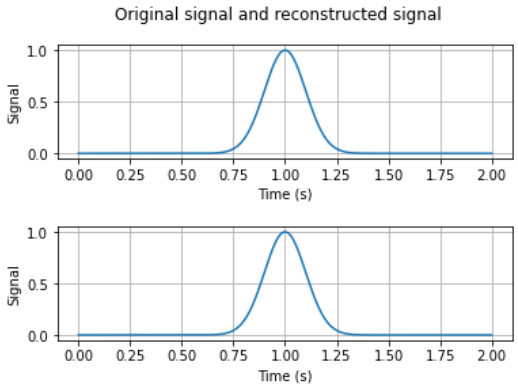
```
class reconstruct():

    def __init__(self, x_s, T_s, tau):

        self.x_s = x_s
        self.T_s = T_s
        self.f_s = np.int(1/T_s)
        self.tau = tau
        self.nu = np.int(1/tau)
        self.N = len(x_s)*np.int(tau/T_s)

    def solve(self):

        x = np.zeros(self.N)
        step = np.int(self.tau/self.T_s)
        x[0::step] = self.x_s
        DFT_obj = dft(x,self.f_s)
        [_,_,f_c,X_c] = DFT_obj.solve()
        fmax = self.nu/2
        index_min  = np.min(np.where(f_c >= -fmax)[0])
        index_max = np.max(np.where(f_c <= fmax)[0])
        X_band = np.zeros(self.N)
        X_band[index_min:index_max] = step*X_c[index_min:index_max]
        X_band = np.roll(X_band, np.int(np.floor(self.N/2+1)))
        iDFT = idft(X_band, self.f_s, self.N)
        x_band, t = iDFT.solve_ifft()

        return x_band
```

```
def q_26_first(mu, sigma, fs, fss, T):
    gaussian_p = gaussian_pulse(mu, sigma, T, fs)
    x = gaussian_p.sig
    t = gaussian_p.t

    subsample_obj = subsample(x, 1/fs, 1/fss)
    x_s, x_delta = subsample_obj.solve()
    reconstruct_obj = reconstruct(x_s, 1/fs, 1/fss)
    x_r = reconstruct_obj.solve()

    # Plot
    fig, axs = plt.subplots(2)
    axs[0].grid()
    axs[1].grid()
    fig.suptitle('Original signal and reconstructed signal' )
    fig.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.6)
    axs[0].plot(t, x)
    axs[0].set_xlabel('Time (s)')
    axs[0].set_ylabel('Signal')
    axs[1].plot(t, x_r)
    axs[1].set_xlabel('Time (s)')
    axs[1].set_ylabel('Signal')
    plt.savefig('signal_and_reconstructed_no_loss_time_'+ str(sigma) + '.png')

q_26_first(1, 0.1, 40000, 4000, 2)
```




```
<ipython-input-8-a1b52b921199>:47: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.N = np.int(np.floor( T * fs))
<ipython-input-19-018ca49aab2d>:7: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fs = np.int(1/Ts)
<ipython-input-19-018ca49aab2d>:9: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fss = np.int(1/tau)
<ipython-input-19-018ca49aab2d>:14: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  step = np.int(self.tau/self.Ts)
<ipython-input-22-3140944e793a>:10: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.N = len(x_s)*np.int(tau/T_s)
<ipython-input-22-3140944e793a>:15: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  step = np.int(self.tau/self.T_s)
<ipython-input-8-a1b52b921199>:24: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  X_c=np.roll(X,np.int(np.ceil(self.K/2-1)))
<ipython-input-22-3140944e793a>:23: ComplexWarning: Casting complex values to real discards the imaginary part
  X_band[index_min:index_max] = step*X_c[index_min:index_max]
/opt/python/envs/default/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1298: ComplexWarning: Casting complex values to real
  return np.asarray(x, float)
```

```
def q_26_sec(mu, sigma, fs, fss, T):
    gaussian_p = gaussian_pulse(mu, sigma, T, fs)
    x = gaussian_p.sig
    t = gaussian_p.t
    subsample_obj = subsample(x, 1/fs, 1/fss)
    x_s, x_delta = subsample_obj.solve_filter()
    reconstruct_obj = reconstruct(x_s, 1/fs, 1/fss)
    x_r = reconstruct_obj.solve()

    # Plot
    fig, axs = plt.subplots(2)
    axs[0].grid()
    axs[1].grid()
    fig.suptitle('Original signal and reconstructed signal' )
    fig.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.6)
    axs[0].plot(t, x)
    axs[0].set_xlabel('Time (s)')
    axs[0].set_ylabel('Signal')
    axs[1].plot(t, x_r)
    axs[1].set_xlabel('Time (s)')
    axs[1].set_ylabel('Signal')
    plt.savefig('signal_and_reconstructed_no_loss_time_'+ str(sigma) + '.png')

q_26_sec(1, 0.1, 40000, 4000, 2)
```




```
<ipython-input-8-a1b52b921199>:47: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.N = np.int(np.floor( T * fs))
<ipython-input-19-018ca49aab2d>:7: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fs = np.int(1/Ts)
<ipython-input-19-018ca49aab2d>:9: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.fss = np.int(1/tau)
<ipython-input-8-a1b52b921199>:24: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, 
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  X_c=np.roll(X,np.int(np.ceil(self.K/2-1)))
<ipython-input-19-018ca49aab2d>:30: ComplexWarning: Casting complex values to real discards the imaginary part
  X_band[index_min:index_max] = X_c[index_min:index_max]
<ipython-input-19-018ca49aab2d>:31: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  X_band = np.roll(X_band, np.int(np.floor(self.N/2+1)))
<ipython-input-19-018ca49aab2d>:35: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  step = np.int(self.tau/self.Ts)
<ipython-input-19-018ca49aab2d>:38: ComplexWarning: Casting complex values to real discards the imaginary part
  x_delta[0::step] = x_s
<ipython-input-22-3140944e793a>:10: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self.N = len(x_s)*np.int(tau/T_s)
<ipython-input-22-3140944e793a>:15: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning,
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  step = np.int(self.tau/self.T_s)
<ipython-input-22-3140944e793a>:16: ComplexWarning: Casting complex values to real discards the imaginary part
  x[0::step] = self.x_s
<ipython-input-22-3140944e793a>:23: ComplexWarning: Casting complex values to real discards the imaginary part
  X_band[index_min:index_max] = step*X_c[index_min:index_max]
/opt/python/envs/default/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1298: ComplexWarning: Casting complex values to real
  return np.asarray(x, float)
```