MAC0317/MAC5920

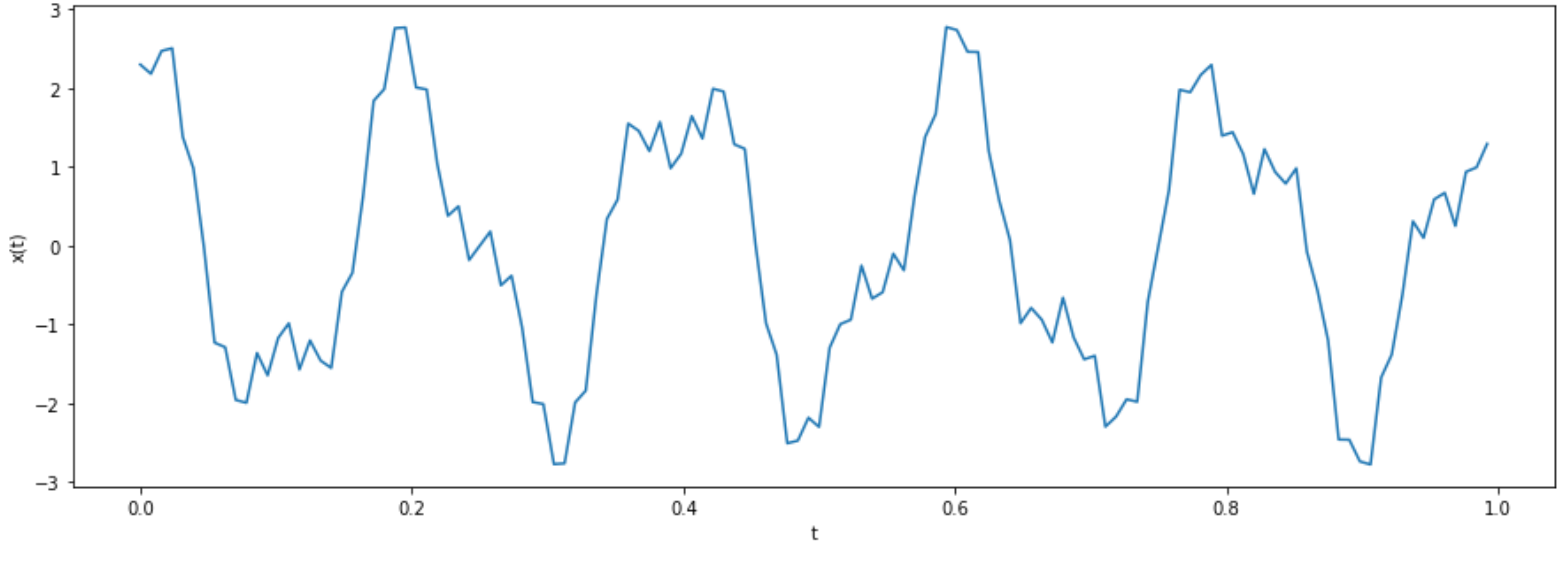
Introdução ao Processamento de Sinais Digitais

Seção 2.3: Um exemplo motivacional

Figura 2.2 - Um sinal simples no domínio do tempo

Considere o sinal $x(t) = 2\cos(2\pi 5t) + 0.8\sin(2\pi 12t) + 0.3\cos(2\pi 47t)$ amostrado no intervalo $t \in [0, 1)$ usando $N = 128$ amostras $\left(t = 0, \frac{1}{128}, \frac{2}{128}, \ldots, \frac{127}{128}\right)$.

In [2]:
```
T = 1; N = 128; t = np.arange(0, T, 1/N); plt.figure(figsize=(15,5))
x = 2*np.cos(2*m.pi*5*t) + 0.8*np.sin(2*m.pi*12*t) + 0.3*np.cos(2*m.pi*47*t)
plt.plot(t,x); plt.xlabel("t"); plt.ylabel("x(t)");plt.show()
print("Primeiras 10 amostras do sinal: x[0:10]={}...".format(np.round(x[0:10],3
)))
```




```
Primeiras 10 amostras do sinal: x[0:10]=[ 2.3    2.183  2.474  2.507  1.383
0.984 -0.023 -1.23   -1.288 -1.958]...
```

Decomposição do sinal em formas básicas E_k

$$x = \sum_{k=0}^{N-1} c_k E_k \implies c_k = \frac{(x, E_k)}{N} = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}$$

In [5]:

```
X = np.fft.fft(x) # a FFT calcula X[k] = (x,E_k)
c = X / N
print(np.round(c,2))
```

```
[-0.  +0.j  -0.  -0.j  -0.  +0.j  -0.  +0.j  -0.  -0.j   1.  -0.j
  0.  +0.j   0.  +0.j  -0.  +0.j   0.  +0.j  -0.  +0.j  -0.  +0.j
 -0.  -0.4j  0.  -0.j  -0.  -0.j   0.  -0.j   0.  -0.j   0.  -0.j
  0.  -0.j   0.  +0.j   0.  +0.j  -0.  -0.j   0.  -0.j   0.  -0.j
  0.  +0.j  -0.  -0.j   0.  +0.j   0.  +0.j   0.  +0.j  -0.  -0.j
  0.  -0.j  -0.  +0.j  -0.  -0.j   0.  -0.j   0.  -0.j   0.  -0.j
  0.  -0.j  -0.  +0.j   0.  -0.j   0.  +0.j   0.  +0.j   0.  -0.j
  0.  +0.j   0.  +0.j   0.  +0.j   0.  +0.j   0.  +0.j   0.15+0.j
 -0.  +0.j  -0.  +0.j  -0.  +0.j  -0.  +0.j  -0.  +0.j  -0.  -0.j
 -0.  +0.j  -0.  -0.j  -0.  -0.j  -0.  +0.j   0.  -0.j  -0.  -0.j
  0.  -0.j   0.  -0.j   0.  -0.j   0.  +0.j   0.  +0.j   0.  -0.j
  0.  +0.j   0.  -0.j   0.  +0.j  -0.  +0.j   0.  +0.j  -0.  -0.j
 -0.  +0.j  -0.  +0.j  -0.  -0.j  -0.  +0.j  -0.  -0.j  -0.  -0.j
 -0.  -0.j  -0.  +0.j  -0.  -0.j   0.15-0.j   0.  -0.j   0.  -0.j
  0.  -0.j   0.  -0.j   0.  -0.j   0.  +0.j   0.  -0.j   0.  -0.j
  0.  +0.j  -0.  +0.j   0.  +0.j   0.  +0.j   0.  +0.j   0.  +0.j
 -0.  +0.j  -0.  -0.j   0.  +0.j  -0.  -0.j   0.  -0.j   0.  -0.j
  0.  -0.j  -0.  +0.j   0.  -0.j   0.  +0.j   0.  +0.j  -0.  +0.j
  0.  -0.j   0.  +0.j   0.  +0.j   0.  +0.j   0.  +0.j   0.  +0.j
 -0.  +0.j   0.  +0.j  -0.  +0.4j  0.  -0.j  -0.  -0.j   0.  -0.j
 -0.  -0.j   0.  +0.j   0.  -0.j   1.  +0.j  -0.  +0.j  -0.  -0.j
 -0.  -0.j  -0.  +0.j ]
```

Coeficientes não-nulos de c e reconstrução de x

In [6]:
```
eps = 1e-8
form = r"x = "
for (k,), ck in np.ndenumerate(c):
    if abs(ck) > eps:
        print ("c({0}) = {1:.2f}".format(k,ck))
        form += r"+c_{{{0}}}E_{{{0}}}".format(k)
display(Math(form))
```

```
c(5) = 1.00-0.00j
c(12) = -0.00-0.40j
c(47) = 0.15+0.00j
c(81) = 0.15-0.00j
c(116) = -0.00+0.40j
c(123) = 1.00+0.00j
```

$$x = +c_{5}E_{5} + c_{12}E_{12} + c_{47}E_{47} + c_{81}E_{81} + c_{116}E_{116} + c_{123}E_{123}$$

Observe que

$$\begin{aligned} x &= E_5 - 0.4iE_{12} + 0.15E_{47} + 0.15E_{81} + 0.4iE_{116} + E_{123} \\ &= E_5 - 0.4iE_{12} + 0.15E_{47} + 0.15E_{-47} + 0.4iE_{-12} + E_{-5} \\ &= (E_5 + E_{-5}) - 0.4i(E_{12} - E_{-12}) + 0.15(E_{47} + E_{-47}) \end{aligned}$$

ou equivalentemente

$$\begin{aligned} x_n &= (e^{i2\pi 5\frac{n}{N}} + e^{-i2\pi 5\frac{n}{N}}) \\ &\quad -0.4i(e^{i2\pi 12\frac{n}{N}} - e^{-i2\pi 12\frac{n}{N}}) \\ &\quad +0.15(e^{i2\pi 47\frac{n}{N}} + e^{-i2\pi 47\frac{n}{N}}) \\ &= 2\cos(2\pi 5\tfrac{n}{N}) - 0.4i(2i\,\sin(2\pi 12\tfrac{n}{N})) + 0.15(2\cos(2\pi 47\tfrac{n}{N})) \\ &= 2\cos(2\pi 5\tfrac{n}{N}) + 0.8\,\sin(2\pi 12\tfrac{n}{N}) + 0.3\cos(2\pi 47\tfrac{n}{N}) \end{aligned}$$

Energia em cada componente

$$
\begin{aligned}
\|x\|^2 &= (x, x) \\
&= \left(\sum_{k=0}^{N-1} c_k E_k , \sum_{l=0}^{N-1} c_l E_l \right) \\
&= \sum_{k,l} c_k \overline{c_l} (E_k , E_l) \\
&= \sum_{k=0}^{N-1} |c_k|^2 (E_k , E_k) = \sum_{k=0}^{N-1} N |c_k|^2
\end{aligned}
$$

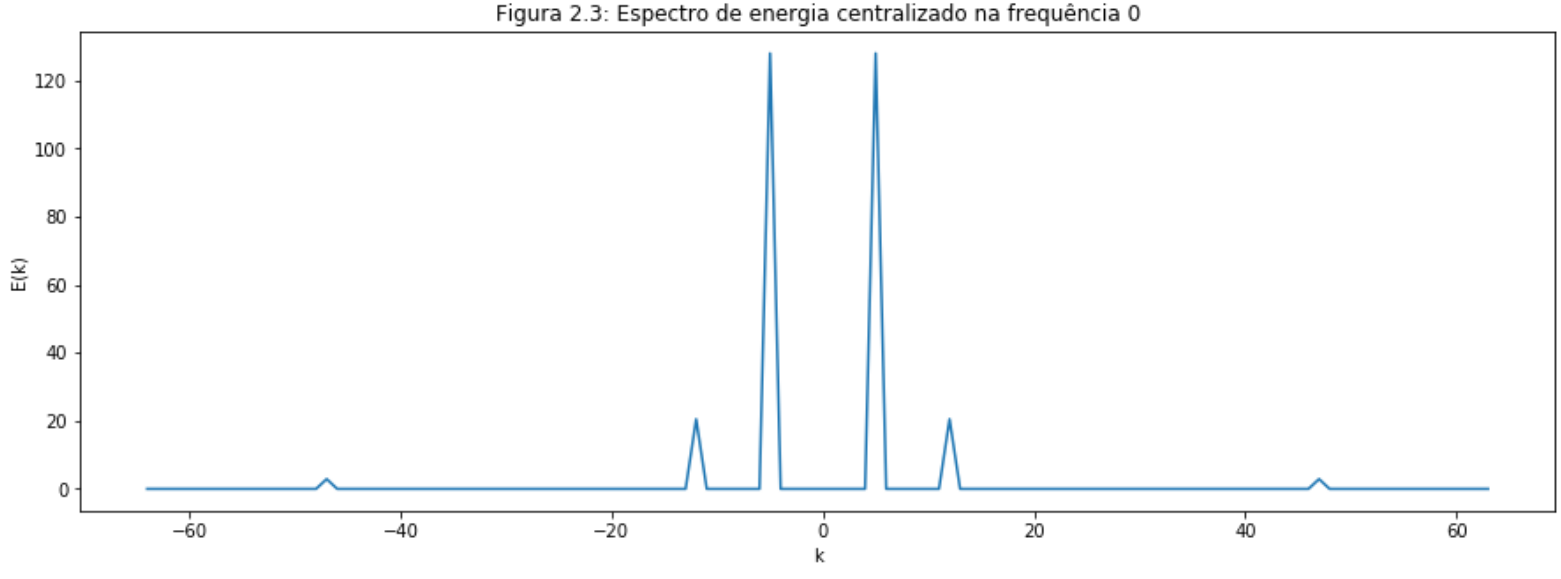
In [12]:

```
print("E(x)\t=\t{0:.2f}".format(np.linalg.norm(x)**2))
print("--------------------------")
E = N * abs(c)**2
for (k,), Ek in np.ndenumerate(E):
    if abs(Ek) > eps: print("E({0})\t=\t{1:.2f}".format(k,Ek))
print("--------------------------")
print("E(c)\t=\t{0:.2f}".format(sum(E)))
```

```
E(x)    =       302.72
--------------------------
E(5)    =       128.00
E(12)   =       20.48
E(47)   =       2.88
E(81)   =       2.88
E(116)  =       20.48
E(123)  =       128.00
--------------------------
E(c)    =       302.72
```

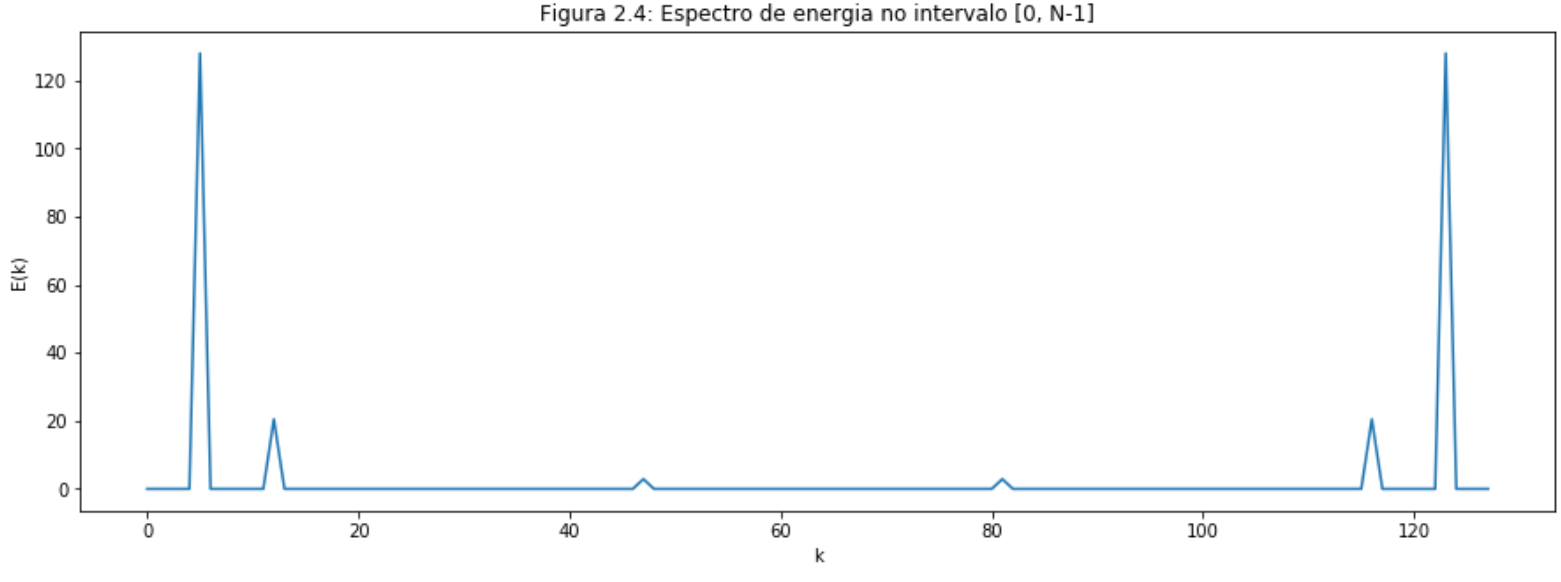
In [13]:

```
plt.figure(figsize=(15,5))
plt.plot(np.arange(N) - N/2, np.roll(E, int(N/2)))
plt.xlabel("k"); plt.ylabel("E(k)")
plt.title("Figura 2.3: Espectro de energia centralizado na frequência 0")
plt.show()
```



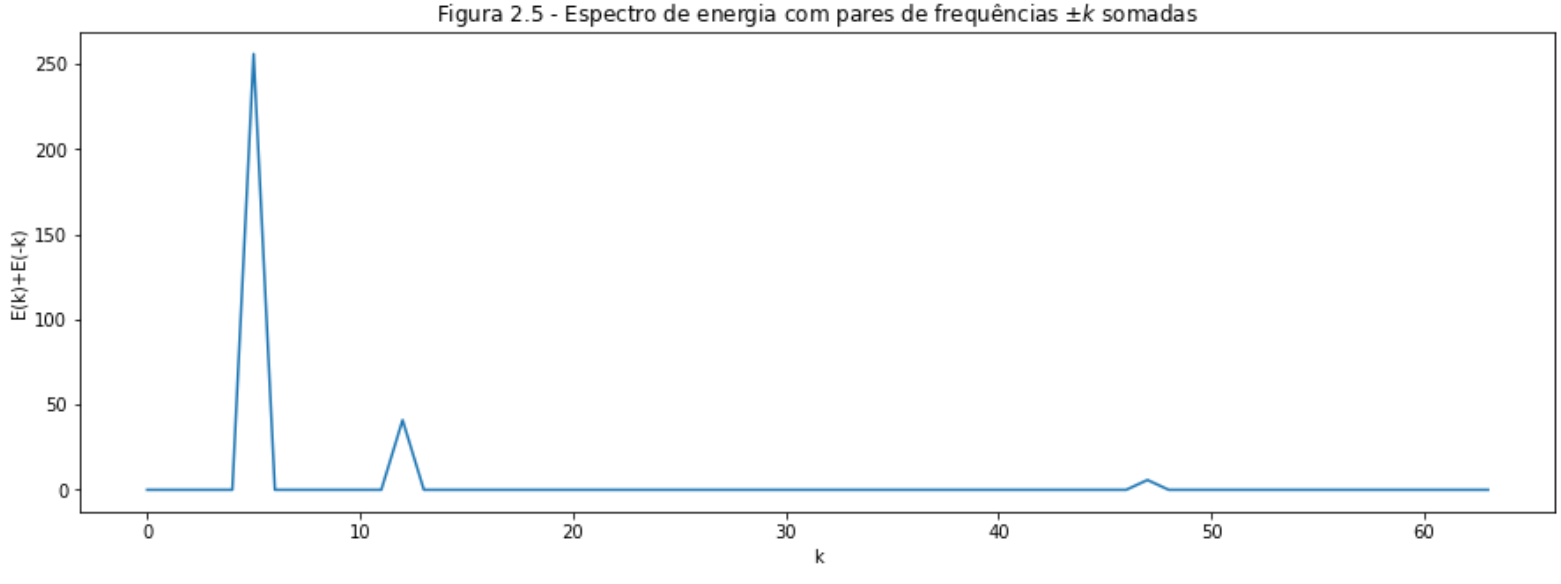
In [14]:

```
plt.figure(figsize=(15,5))
plt.plot(E); plt.xlabel("k"); plt.ylabel("E(k)")
plt.title("Figura 2.4: Espectro de energia no intervalo [0, N-1]")
plt.show()
```



In [15]:

```
plt.figure(figsize=(15,5))
E[1:int(N/2)-1:1] *= 2
plt.plot(E[0:int(N/2)]); plt.xlabel("k"); plt.ylabel("E(k)+E(-k)")
plt.title(r"Figura 2.5 - Espectro de energia com pares de frequências $\pm k$ so
madas")
plt.show()
```



Removendo ruído associado a altas frequências

Uma técnica simples de *suavização* ou *remoção de ruído* de sinais consiste em manipular o espectro do sinal, removendo componentes senoidais/exponenciais de alta frequência (índices acima de um certo k_0):

$$x \longrightarrow X = DFT(x) \longrightarrow \quad \tilde{X}_k \leftarrow \begin{cases} X_k & |k| < k_0 \\ 0 & |k| \geq k_0 \end{cases} \quad \longrightarrow \tilde{x} = IDFT(\tilde{X})$$

Observe que usamos $|k|$ para separar as altas das baixas frequências. Isso pressupõe que consideraremos o intervalo de frequências $\left(-\frac{N}{2}, +\frac{N}{2}\right]$ (ou $k = -63, \ldots, 64$ no exemplo anterior). Se o intervalo de representação de frequências adotado fosse $[0, N)$ ($k = 0, \ldots, 127$ no exemplo anterior), então as "altas frequências" estariam na porção central do vetor, ou seja, no intervalo $k_0, \ldots, N - k_0$

In [18]:

```
# Elimina "altas frequências", acima de k0=40 nesse exemplo
# (observe que k=40 ciclos por N amostras equivale a 40 Hz nesse exemplo,
# porque N amostras aqui corresponde a 1 segundo do sinal original)
Y = X; k0 = 40
Y[k0:N-k0:1] *= 0
y = np.real(np.fft.ifft(Y))
f, ax = plt.subplots(1,2,figsize=(15,5))
ax[0].set_title("Sinal original");ax[0].plot(t,x)
ax[1].set_title("Sinal filtrado");ax[1].plot(t,y)
f.suptitle("Figura 2.6 - Sinal ruidoso e sinal filtrado (frequência de corte=40Hz)")
plt.show()
```