MAC0317/MAC5920

Introdução ao Processamento de Sinais Digitais

Seção 6.4: Bancos de filtros multiestágios

Aplicação reiterada de bancos de filtros

A transformada definida por

$$x \mapsto (X_l, X_h) = (D(l_a * x), D(h_a * x))$$

pode ser iterada a fim de ganhar novas perspectivas sobre o sinal e novas possibilidades de processamento.

A aplicação *reiterada* ou *encaixada* do banco de filtros pode ser feita recodificando os sinais X_l e/ou X_h através do mesmo processo: filtragem e subamostragem

Aplicação reiterada em todos os canais

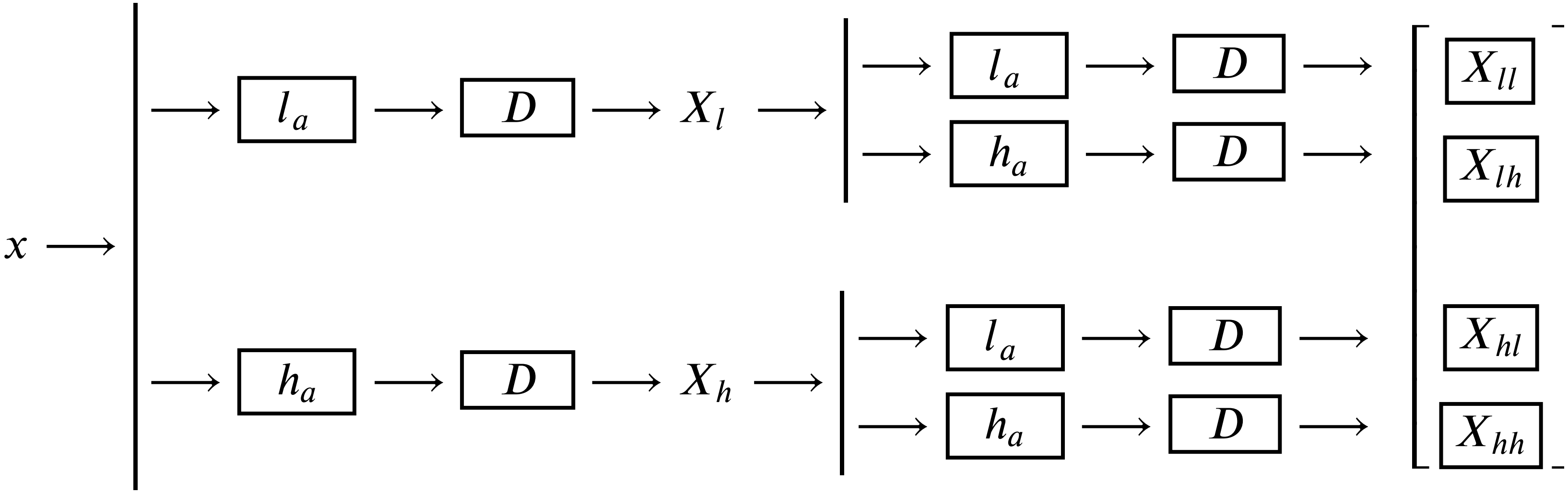
Se decidirmos recodificar tanto X_l quanto X_h, através das transformações

$$\begin{aligned} X_l &\mapsto (X_{ll}, X_{lh}) = (D(l_a * X_l), D(h_a * X_l)) \\ X_h &\mapsto (X_{hl}, X_{hh}) = (D(l_a * X_h), D(h_a * X_h)) \end{aligned}$$

teremos a teoria dos *pacotes wavelet* (*wavelet packets*). A transformada completa em dois estágios será

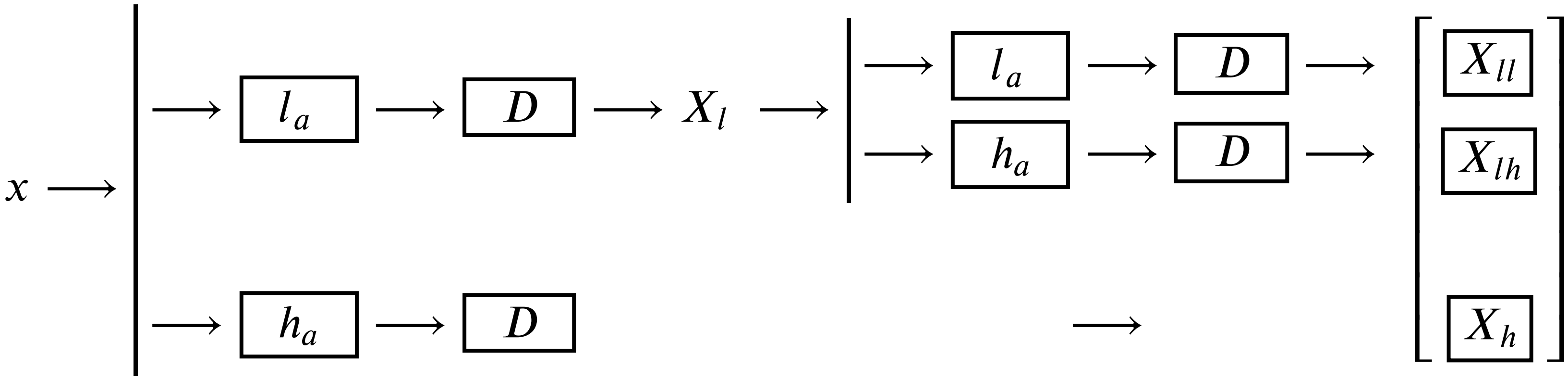
$$x \mapsto (X_{ll}, X_{lh}, X_{hl}, X_{hh}).$$

Graficamente, podemos representar o banco de filtros completo em dois estágios como



Aplicação reiterada apenas nos coeficientes de aproximação (X_l)

A teoria de **wavelets** prevê a aplicação reiterada ou encaixada do banco de filtros apenas no canal de baixas frequências em cada estágio:



Isso equivale à transformada

$$x \mapsto (X_{ll}, X_{lh}, X_h) = \left(\underbrace{D(l_a * \overbrace{D(l_a * x)}^{X_l})}_{X_{ll}}, \underbrace{D(h_a * \overbrace{D(l_a * x)}^{X_l})}_{X_{lh}}, D(h_a * x) \right),$$

cuja inversa depende da aplicação também em dois estágios das operações de *superamostragem* e *filtragem*, ou seja, a transformada inversa $(X_{ll}, X_{lh}, X_h) \mapsto x$ é dada por

$$x = l_s * U \left(\overbrace{l_s * U(X_{ll}) + h_s * U(X_{lh})}^{X_l} \right) + h_s * U(X_h).$$

Transformada em mais de dois estágios

O processo acima pode ser generalizado para M estágios, sempre recodificando o canal de baixas frequências obtidos no estágio anterior:

$$x \mapsto \begin{bmatrix} \boxed{X_l} \\ X_h \end{bmatrix} \mapsto \begin{bmatrix} \boxed{X_{ll}} \\ X_{lh} \\ X_h \end{bmatrix} \mapsto \begin{bmatrix} \boxed{X_{lll}} \\ X_{llh} \\ X_{lh} \\ X_h \end{bmatrix} \mapsto \begin{bmatrix} \boxed{X_{llll}} \\ X_{lllh} \\ X_{llh} \\ X_{lh} \\ X_h \end{bmatrix} \mapsto \cdots$$

onde após M aplicações teremos uma componente $X_{\underbrace{ll\ldots l}_{M}}$ e M componentes da forma $X_{\underbrace{ll\ldots l}_{L}h}$ para $L = M - 1, M - 2, \ldots, 0$.

A decodificação deve ser realizada também em M etapas, seguindo a sequência acima na direção oposta.

Exemplo 6.6

Considere outra vez o sinal do exemplo 6.2:

$$x(t) = \begin{cases} \sin(2\pi 12t), & 0 \leq t < t_1 \\ 0.8, & t_1 \leq t < t_2 \\ 0.3, & t_2 \leq t < t_3 \\ 0, & t_3 \leq t < 1 \end{cases}$$

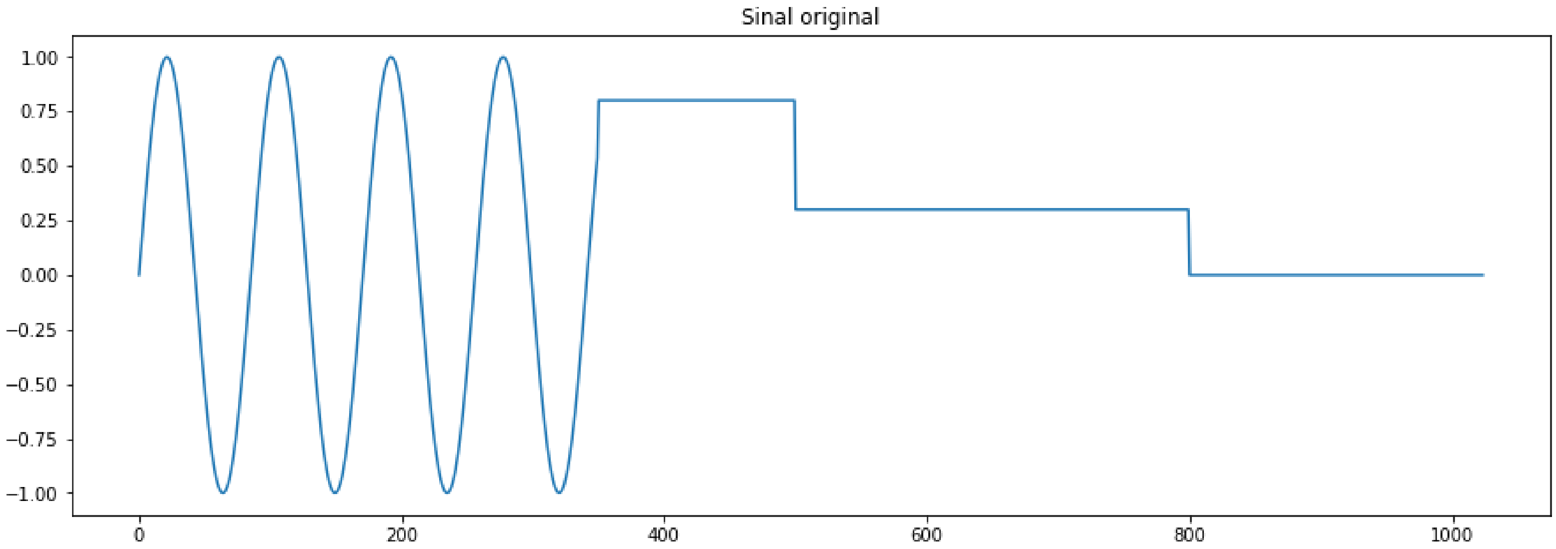
onde $0 < t_1 < t_2 < t_3 < 1$.

Consideraremos sua codificação em até 4 estágios pelo banco de filtros ortogonal da Haar, ou seja, usando

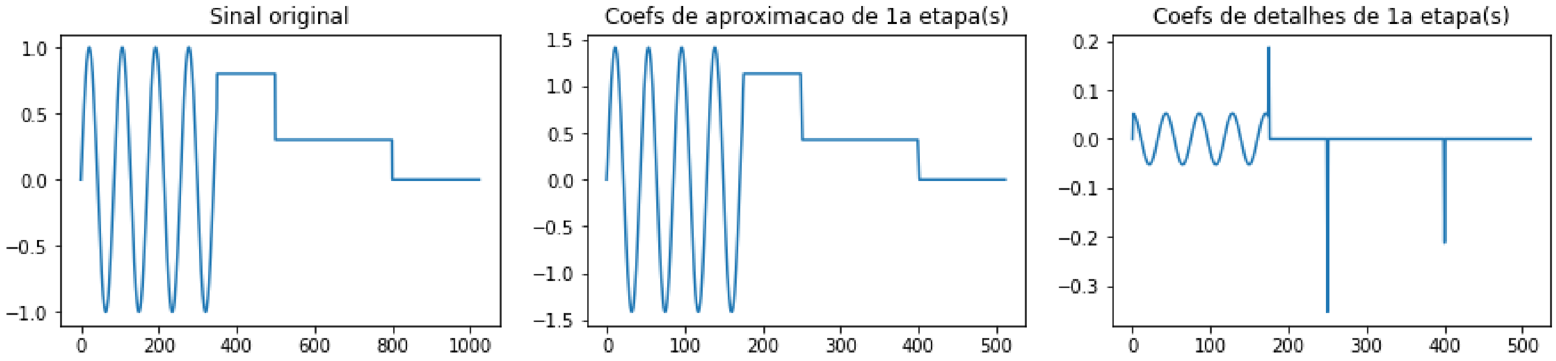
$$\begin{aligned} l &= (\ldots, 0, \quad 0, \quad \overbrace{\frac{\sqrt{2}}{2}}^{n=0}, \quad \frac{\sqrt{2}}{2}, \quad 0, \ldots) \\ h &= (\ldots, 0, \quad 0, \quad \frac{\sqrt{2}}{2}, \quad -\frac{\sqrt{2}}{2}, \quad 0, \ldots) \end{aligned}$$

In [2]:

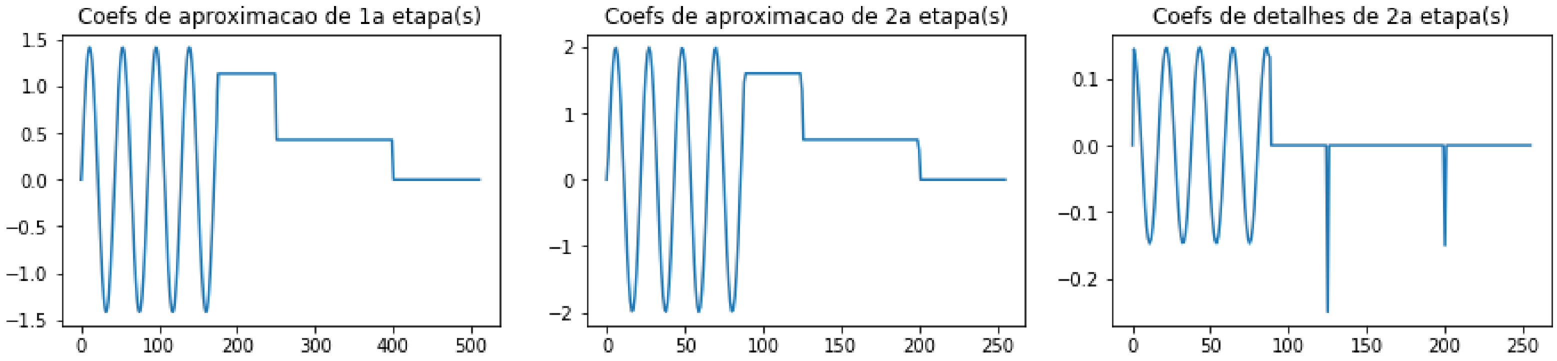
```
N = 1024;x = np.zeros(1024);x[0:350] = np.sin(2 * m.pi * 12 *  np.arange(0, 1,
1/N)[:350])
x[350:500] = 0.8 * np.ones(150);x[500:800] = 0.3 * np.ones(300)
fig, ax = plt.subplots(1,1, figsize=(15,5))
ax.plot(x);ax.set_title("Sinal original");plt.show()
```



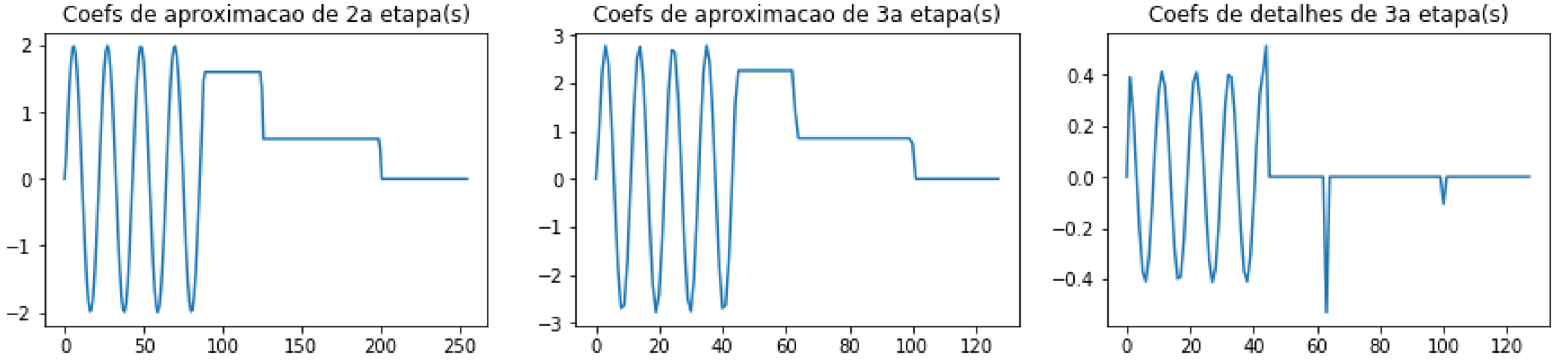
In [49]:
```
coeficientes_etapa(1)
```



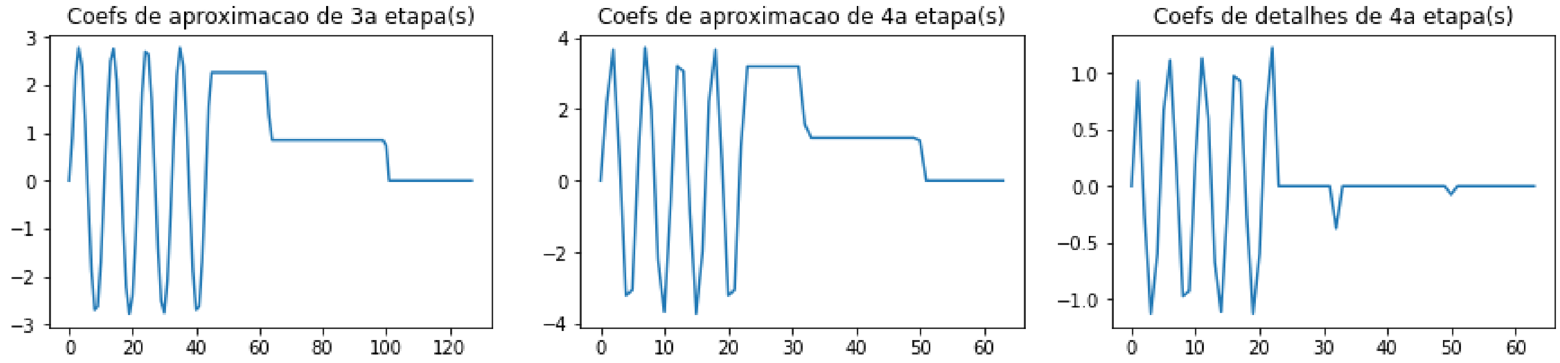
In [50]:
```
coeficientes_etapa(2)
```



In [51]:
```
coeficientes_etapa(3)
```



In [52]:
```
coeficientes_etapa(4)
```



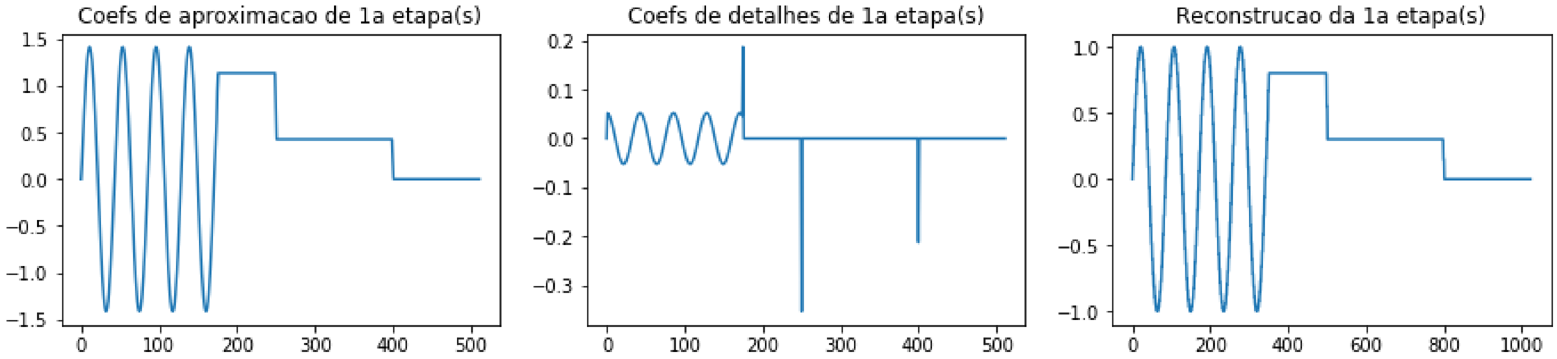
Observaremos agora o efeito de comprimir as representações em M estágios da forma

$$(X_{ll\cdots ll}, X_{ll\cdots lh}, \cdots, X_h)$$

mantendo *apenas* os coeficientes de aproximação da última etapa, ou seja, resintetizando o sinal a partir de

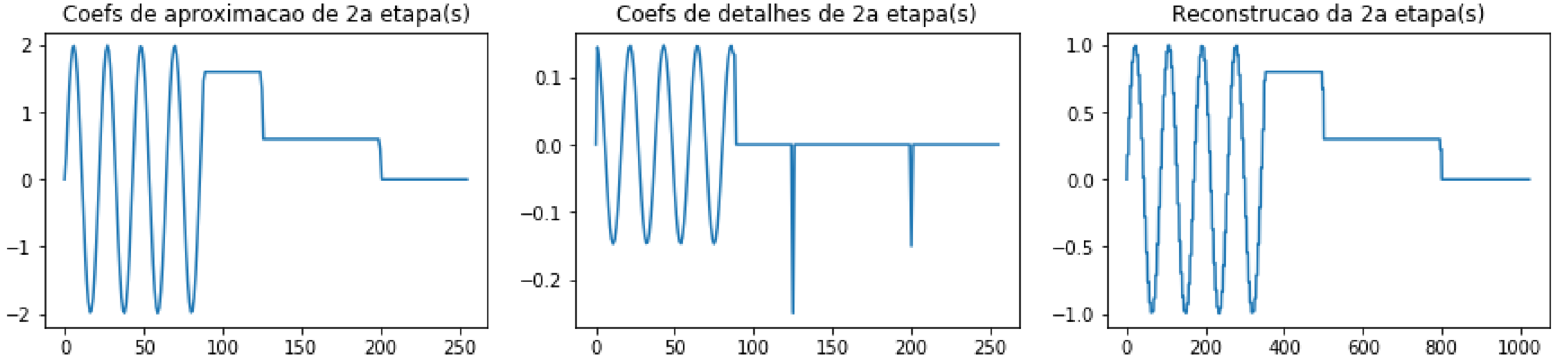
$$\left(X_{ll\cdots ll}, \overbrace{X_{ll\cdots lh}}^{=0}, \cdots, \overbrace{X_h}^{=0}\right).$$

In [56]:
```
reconstrução_etapa(1)
```



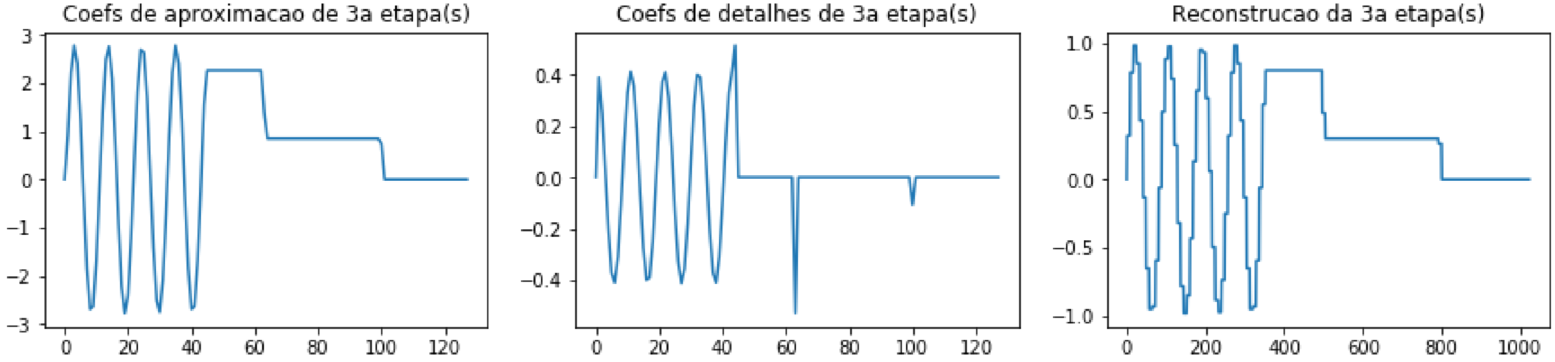

```
Tamanho (relativo) do vetor comprimido = 50.00%
Erro (relativo) da reconstrução = 0.15%
```

```
In [57]: reconstrução_etapa(2)
```




```
Tamanho (relativo) do vetor comprimido = 25.00%
Erro (relativo) da reconstrução = 0.51%
```

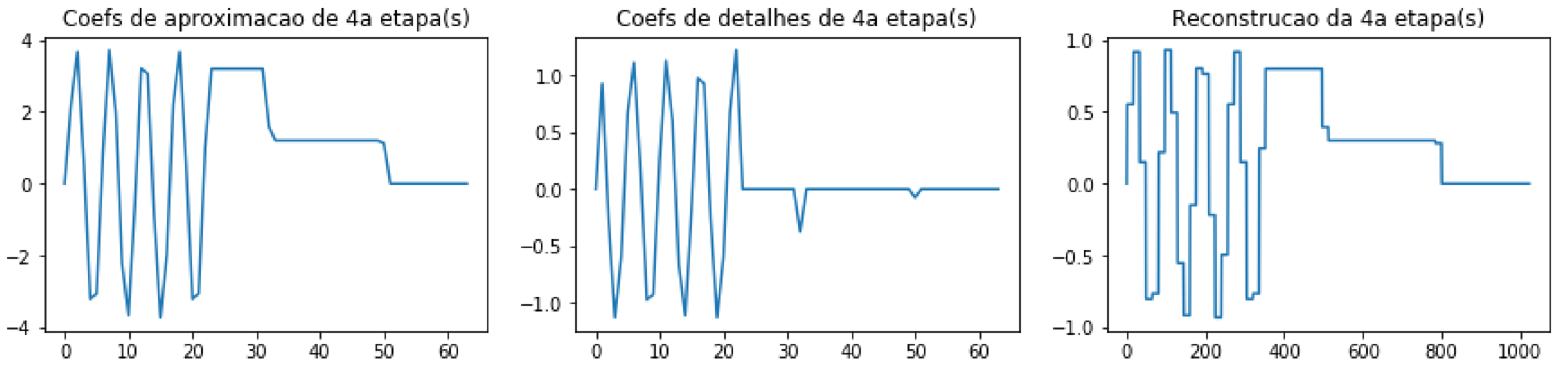
In [58]:
```
reconstrução_etapa(3)
```




```
Tamanho (relativo) do vetor comprimido = 12.50%
Erro (relativo) da reconstrução = 1.96%
```

In [59]:

```
reconstrução_etapa(4)
```




```
Tamanho (relativo) do vetor comprimido = 6.25%
Erro (relativo) da reconstrução = 7.10%
```

Pywavelets

O módulo Pywavelets fornece diversas funções para lidar com Wavelets discretas e contínuas e implementa nativamente diversas familias de funções bases. O banco de filtros de Harr é implementado ligeiramente diferente do utilizado no livro, e portanto os coeficientes de aproximação e detalhe tem pequenas diferenças em relação as figuras do capítulo 6, porém a transformada inversa é realizada de maneira que a reconstrução continua perfeita.

Exemplos de uso:

```
y = pywt.wavedec(x, 'haar', mode='zero', level=j)
x = pywt.waverec(y, 'haar', mode='zero')
```

A documentação do módulo pode ser encontrada aqui:
https://pywavelets.readthedocs.io/en/latest/ref/index.html
(https://pywavelets.readthedocs.io/en/latest/ref/index.html)