tp1

June 3, 2019

0.1

Procesamiento Digital de Señales MSE2019 18PDS

1 Trabajo Práctico 1

Pablo Slavkin

1.1 Recepción de una señal degradada

1) a) Filtrar la señal con un filtro digital para eliminar potencia de ruido. Pueden probar con filtros FIR o IIR, los que les parezcan convenientes. £Es óptimo el filtro elegido para detectar la información en forma confiable?

Respuesta: Se levantan las seniales y se analizan sus espectros:

```
In [1]: import  numpy   as np
        from    plotter import *
        from    filter  import *

        fs    = 20                      #frecuencia de sampleo de la senial
        pl    = plotter_class ( 2,2 )   #objeto para generar un grafico de 2x2
        f     = filter_class  ( )       #objero para funciones de filtrado

        signal = np.load('signalLowSNR.npy')        #lee la senial a decodidicar
        #signal = np.load('signal.npy')
        signal=signal[:2560*1]                      #tomo una pocrion para debug, luego coment
        pl.plot_signal  ( 1 ,pl.spaceX(signal) ,signal ,'signal' ,'time' ,'volt' ,trace='-' )

        pulse  = np.load('pulse.npy')               #cargo el pulso original para plotearlo
        pl.plot_signal  ( 2 ,pl.spaceX(pulse)  ,pulse  ,'pulse'  ,'time' ,'volt' ,trace='-' )

        pl.plot_dft_signal ( 3 ,fs ,signal ,'dft signal' ,'Hz' ,'Pnormal' ,trace='-' ,center=si
        pl.plot_dft_signal ( 4 ,fs ,pulse  ,'dft pulse'  ,'Hz' ,'Pnormal' ,trace='-')

        pl.plot_show()
```

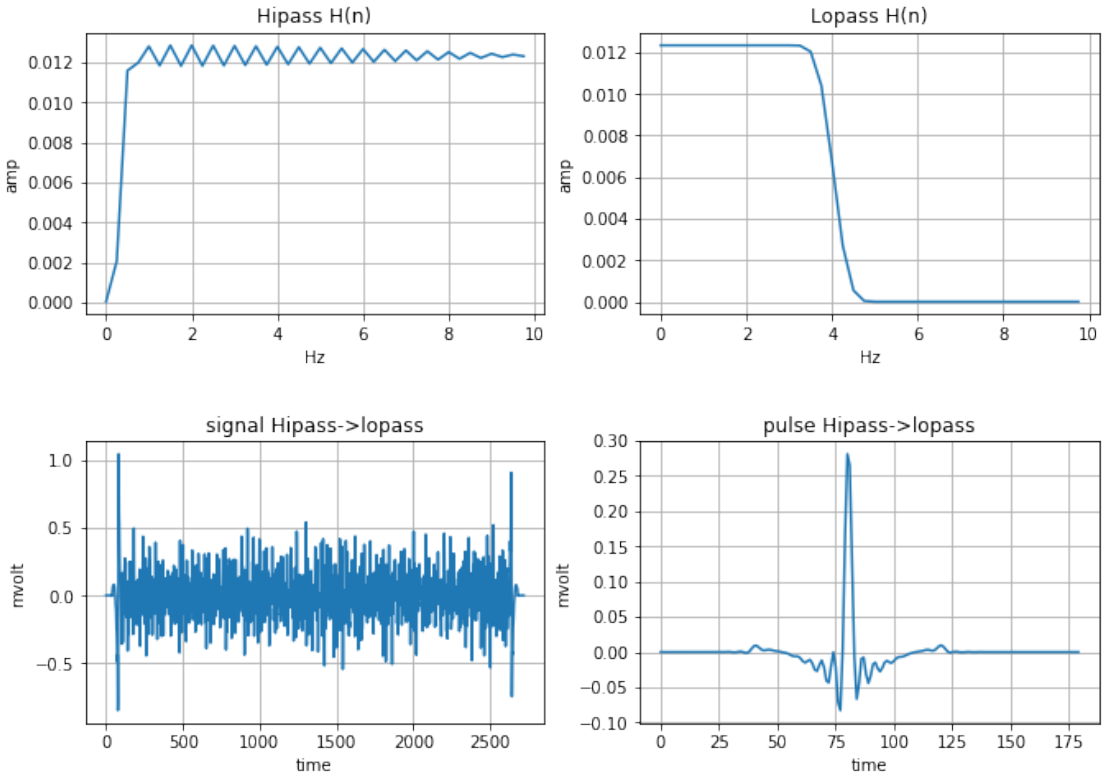
```
/opt/anaconda3/lib/python3.7/site-packages/matplotlib/figure.py:98: MatplotlibDeprecationWarnin
Adding an axes using the same arguments as a previous axes currently reuses the earlier instanc
  "Adding an axes using the same arguments as a previous axes "
```

```
<Figure size 1000x700 with 4 Axes>
```

#

Respuesta: Se puede ver que tanto el espectro del pulso como el espectro de la senial tienen potencia distribuida en todo el espectro, con lo cual cualquiera sea el filtro que se elija para limitar la potencia de ruido que se aprecia en el espectro de signal, recortara un poco de la energia de la senial de interes. Igualmente aun asi, se lograra aumentar un poco la SNR. Como se ve en el siguiente filtrado:

```
In [2]: pl    = plotter_class ( 2,2 )       #nuevo grafico

        hipass = np.load("hipass_fir.npz")['ba'][0]     #leo los parametros del pasabajo
        lopass = np.load("lopass_fir.npz")['ba'][0]     # y pasaaltos diseniados
        pl.plot_dft_signal ( 1 ,fs,hipass,'Hipass H(n)' ,'Hz' ,'amp' ,trace='-' )
        pl.plot_dft_signal ( 2 ,fs,lopass,'Lopass H(n)' ,'Hz' ,'amp' ,trace='-' )

        signal = f.fir ( signal ,hipass )               #aplico el filtrado a la senial y la m
        signal = f.fir ( signal ,lopass )
        pl.plot_signal ( 3 ,pl.spaceX(signal ),signal ,'signal Hipass->lopass' ,'time' ,'mvolt

        ans = f.fir ( pulse ,hipass )                   #aplico el filtrado al pulso para veri
        ans = f.fir ( ans   ,lopass )                   #recortando demasiado
        pl.plot_signal ( 4 ,pl.spaceX(ans    ),ans    ,'pulse Hipass->lopass'  ,'time' ,'mvolt

        pl.plot_show()
```



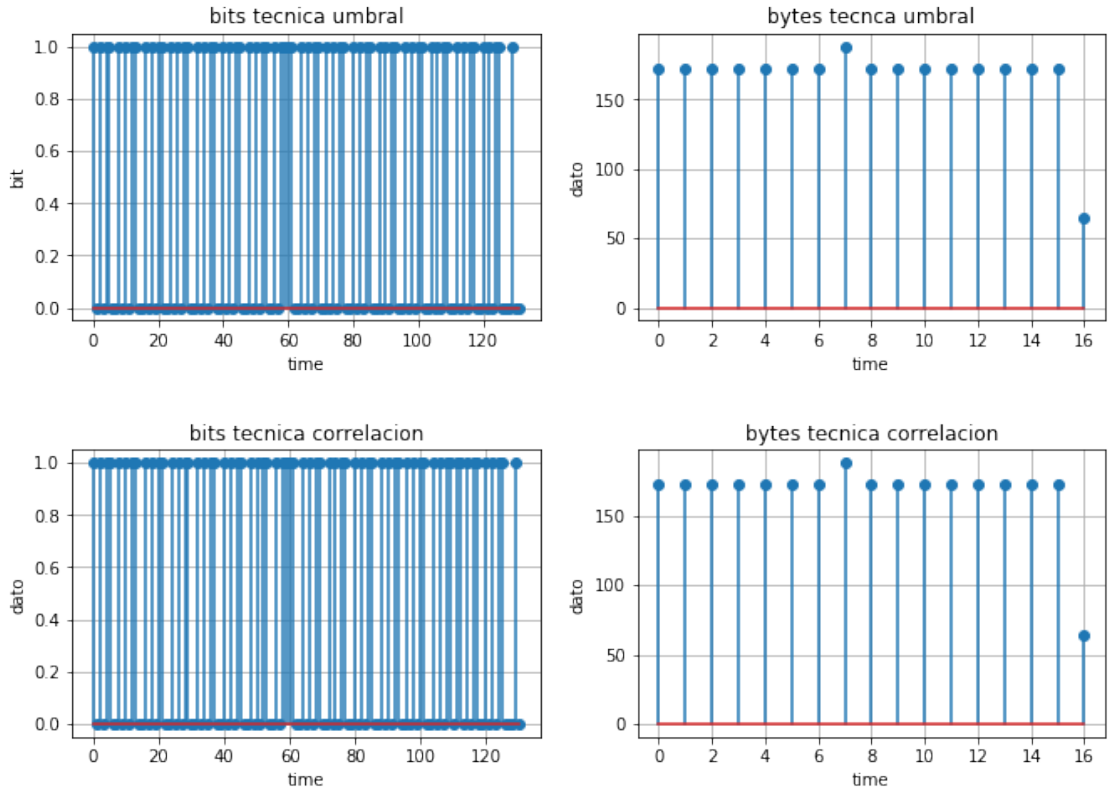
Se puede ver el resultado de la senial y el pulso filtrado y se aprecia como se ha recortado bastante la energia perdiendo parte de la senial

Con la senial mas limpia, se procede a decodificar. Se testean 2 tecnicas, una de umbral y otra de correlacion. En la primera se toma el primer sample de cada 20 (el largo del pulso) y se decide si es mayor o menor a cero para determinar si el pulso es un uno o un cero. Es decir de todos los 20 samples de energia del pulso solo se usa el primero. En la tecnica de correlacion en cambio se correlacionan los 20 samples que tiene la senial con el pulso original conocido y se toma el maximo para determinar si se 'parece' mas a un pulso positivo o negativo

```
In [3]: groupDelay  = hipass.size//2+lopass.size//2     #calculo el retado basado en el largo
        signal      = signal[groupDelay+1:]             #demoro la senial original dicaha cant
        #esto es lo importante, aca lo que hago es quedarme solo con el primer pulso de cada 2
        #sample que tiene la mayor energia y es el que usare para comparar contra un umbral pa
        #el pulso es 1 o -1
        pl    = plotter_class ( 2,2 )

        ans         = signal[::pulse.size]
        ans[ans>=0] = 1                                 #digitalizo la senial con un comparado
        ans[ans<0]  = 0
        ans         = np.uint8(ans)                     #lo necesito apra empaquetar luego a b
        pl.stem_signal  ( 1 ,pl.spaceX(ans) ,ans ,'bits tecnica umbral'  ,'time' ,'bit'  ,trac

        ans = np.packbits(ans)                          #empaqueto de 8 por comodidad
```

```
        pl.stem_signal  ( 2 ,pl.spaceX(ans) ,ans ,'bytes tecnca umbral' ,'time' ,'dato' ,trace=

        print(f"tecnica umbral=\n{ans}")                                    #muestra los res

        ans=np.zeros(0)
        ans2=np.zeros(0)
        for i in range(signal.size//pulse.size):
            ans=np.append(ans,np.correlate(signal[i*pulse.size:(i+1)*pulse.size],pulse))

        ans[ans>=0] = 1                                  #digitalizo la senial con un comparado
        ans[ans<0]  = 0
        ans         = np.uint8(ans)                      #lo necesito apra empaquetar luego a b
        pl.stem_signal  ( 3 ,pl.spaceX(ans),ans ,'bits tecnica correlacion' , 'time' ,'dato'
        ans = np.packbits(ans)                           #empaqueto de 8 por comodidad
        pl.stem_signal  ( 4 ,pl.spaceX(ans),ans ,'bytes tecnica correlacion' , 'time' ,'dato'
        print(f"tecnica correlacion=\n{ans}")                                  #muestra lo

        pl.plot_show()
```

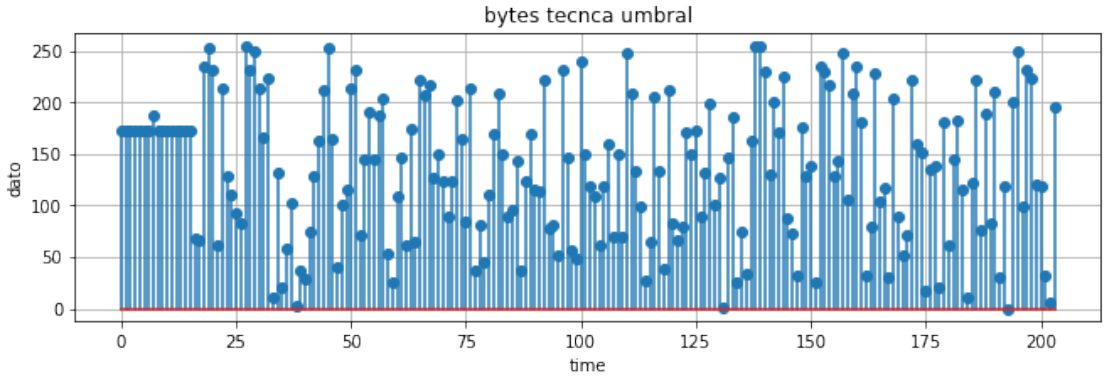
```
tecnica umbral=
[172 172 172 172 172 172 172 188 172 172 172 172 172 172 172 172  64]
tecnica correlacion=
[172 172 172 172 172 172 172 188 172 172 172 172 172 173 172 172  64]
```

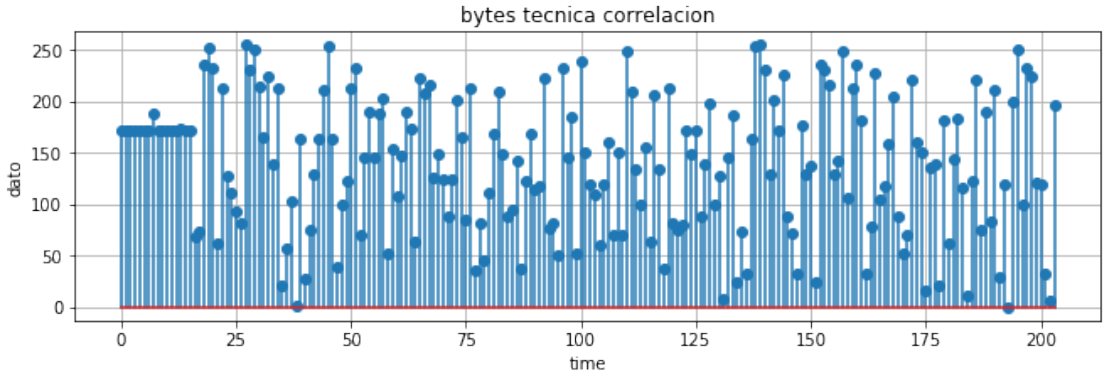


Conclusion: Se filtro la senial y el pulso para eliminar el ruido teniendo en cuenta de no deteriorar demasiado la senial. Se utilizaron las tecnicas de umbral usando 1 solo sample y la de correlacion con el pulso original utilizando los 20 samples de cada pulso para la toma de decision y se obtuvieron resultados similares. A continuacion una decodificacion completa

```
In [4]: import  numpy   as np
        from    plotter import *
        from    dft     import *
        from    filter  import *
        from    dft     import *

        fs    = 20                      #frecuencia de sampleo de la senial
        pl    = plotter_class ( 2,1 )   #objeto para generar un grafico de 2x2
        f     = filter_class  ( )       #objero para funciones de filtrado

        signal = np.load('signalLowSNR.npy')        #lee la senial a decodidicar
        pulse  = np.load('pulse.npy')               #cargo el pulso original para plotearlo
        hipass = np.load("hipass_fir.npz")['ba'][0]     #leo los parametros del pasabajo
        lopass = np.load("lopass_fir.npz")['ba'][0]     # y pasaaltos diseniados
        signal = f.fir ( signal ,hipass )               #aplico el filtrado a la senial y la m
        signal = f.fir ( signal ,lopass )

        groupDelay  = hipass.size//2+lopass.size//2     #calculo el retado basado en el largo
        signal      = signal[groupDelay+1:]             #demoro la senial original dicaha cant

        ans         = signal[::pulse.size]
        ans[ans>=0] = 1                                 #digitalizo la senial con un comparado
        ans[ans<0]  = 0
        ans         = np.uint8(ans)                     #lo necesito apra empaquetar luego a b
        ans = np.packbits(ans)                          #empaqueto de 8 por comodidad
        pl.stem_signal  ( 1 ,pl.spaceX(ans) ,ans ,'bytes tecnca umbral' ,'time' ,'dato' ,trace=

        print(f"tecnica umbral=\n{ans}")                                    #muestra los res

        ans=np.zeros(0)
        ans2=np.zeros(0)
        for i in range(signal.size//pulse.size):
            ans=np.append(ans,np.correlate(signal[i*pulse.size:(i+1)*pulse.size],pulse))

        ans[ans>=0] = 1                                 #digitalizo la senial con un comparado
        ans[ans<0]  = 0
        ans         = np.uint8(ans)                     #lo necesito apra empaquetar luego a b
        ans = np.packbits(ans)                          #empaqueto de 8 por comodidad
        pl.stem_signal  ( 2 ,pl.spaceX(ans),ans ,'bytes tecnica correlacion' , 'time' ,'dato'
        print(f"tecnica correlacion=\n{ans}")                                  #muestra lo
```

```
        pl.plot_show()

tecnica umbral=
[172 172 172 172 172 172 172 188 172 172 172 172 172 172 172 172  68  66
 235 252 232  62 213 128 111  93  82 255 231 250 214 166 224  11 132  21
  58 103   2  36  28  75 128 163 211 253 164  40 100 115 213 232  71 145
 190 145 188 203  53  26 108 147  61 174  64 222 207 216 126 149 124  89
 124 202 165  85 213  36  81  45 111 169 209 149  89  95 143  37 123 169
 115 113 222  77  81  51 232 146  57  49 239 150 119 109  61 119 160  70
 150  70 248 209 134  99  27  64 206 134  38 212  82  67  80 171 149 172
  89 131 198 100 127   1 146 186  25  74  33 163 254 255 230 130 201 171
 225  88  72  32 176 129 138  25 235 230 216 129 143 248 106 208 235 181
  32  79 228 104 117  30 204  89  52  71 221 160 151  17 135 139  21 181
  62 144 183 116  11 122 221  76 189  83 210  30 119   0 200 250  99 232
 224 121 119  32   6 196]
tecnica correlacion=
[172 172 172 172 172 172 172 188 172 172 172 172 172 173 172 172  68  74
 235 252 232  62 213 128 111  93  82 255 231 250 214 166 224 139 212  21
  58 103   2 164  28  75 129 163 211 253 164  40 100 123 213 232  71 145
 190 145 188 203  53 154 108 147 189 174  64 222 207 216 126 149 124  89
 124 202 165  85 213  36  81  45 111 169 209 149  89  95 143  37 123 169
 115 117 222  77  81  51 232 146 185  53 239 150 119 109  61 119 160  70
 150  70 248 209 134  99 155  64 206 134  38 212  82  75  80 171 149 172
  89 139 198 100 127   9 146 186  25  74  33 163 254 255 230 130 201 171
 225  88  72  32 176 129 138  25 235 230 216 129 143 248 106 212 235 181
  32  79 228 104 117 158 204  89  53  71 221 160 151  17 135 139  21 181
  62 144 183 116  11 122 221  76 189  83 211  30 119   0 200 250  99 232
 224 121 119  32   6 196]
```



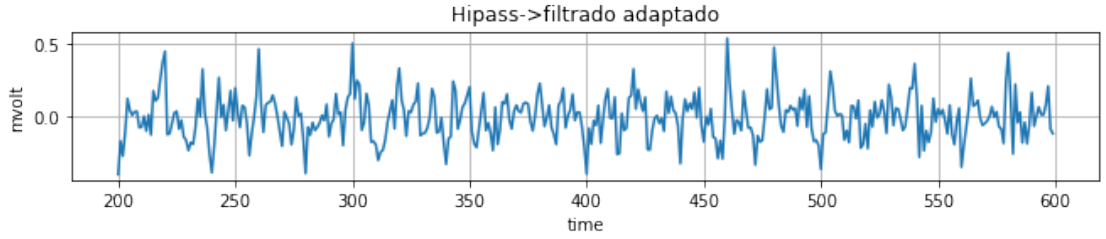


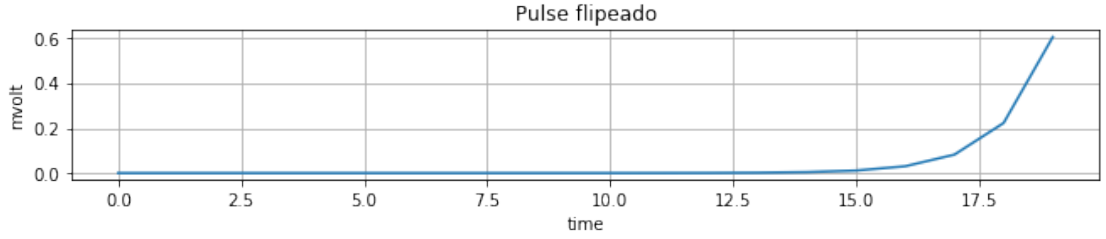
Parte 2:Repetir el procedimiento para la parte 1, pero utilizar el filtro y el umbral óptimo.

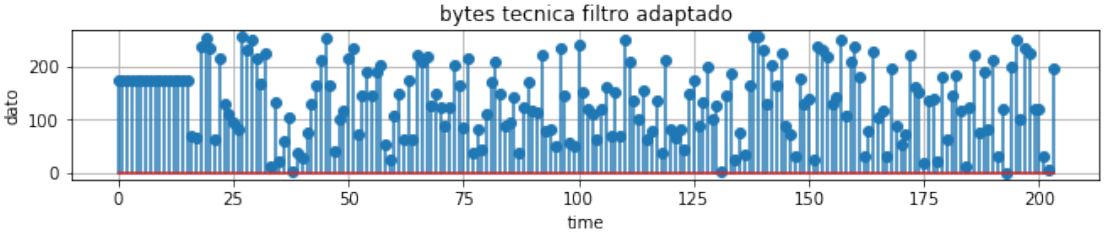
```
In [6]: import  numpy   as np
        from    plotter import *
        from    filter  import *

        fs    = 20                      #frecuencia de sampleo de la senial
        #pl    = plotter_class ( 2,2 )   #objeto para generar un grafico de 2x2
        f     = filter_class  ( )       #objero para funciones de filtrado

        signal = np.load('signalLowSNR.npy')        #lee la senial a decodidicar
        signalOriginal=signal
        pulse  = np.load('pulse.npy')

        pl    = plotter_class ( 3,1 )
        pulse=np.flip(pulse)
        signal=signalOriginal[:]
        hipass = np.load("hipass_fir.npz")['ba'][0]    #leo los parametros del pasabajo
        ans = f.fir ( signal ,hipass )                 #elimino continua y bajas frecs
        ans = f.fir ( ans    ,pulse )                  #Aca aplico el filtro adaptado
        pl.plot_signal ( 1 ,pl.spaceX(ans),ans     ,'Hipass->filtrado adaptado'  ,'time' ,'mvol
        pl.plot_signal ( 2 ,pl.spaceX(pulse),pulse ,'Pulse flipeado'  ,'time' ,'mvolt' ,trace=
```

```
    groupDelay  = hipass.size//2+pulse.size//2      #calculo el retado basado en el largo
    ans         = ans[groupDelay+pulse.size//2::pulse.size]       #demoro la senial origin
    ans[ans>=0] = 1                                #digitalizo la senial con un comparado
    ans[ans<0]  = 0
    ans         = np.uint8(ans)                    #lo necesito apra empaquetar luego a b
    #pl.stem_signal  ( 3 ,pl.spaceX(ans) ,ans ,'bits tecnica filtro adaptado'  ,'time' ,'b
    ans = np.packbits(ans)                         #empaqueto de 8 por comodidad
    pl.stem_signal  ( 3 ,pl.spaceX(ans) ,ans ,'bytes tecnica filtro adaptado' ,'time' ,'dat
    print(f"tecnica filtro adaptado=\n{ans}")
    pl.plot_show()
```

```
tecnica filtro adaptado=
[172 172 172 172 172 172 172 172 172 172 172 172 172 172 172 172  68  66
 235 252 232  62 213 128 111  93  82 255 231 250 214 166 224  11 132  21
  58 103   2  36  28  75 128 163 211 253 164  40 100 115 213 232  71 145
 190 145 188 203  53  26 108 147  61 174  64 222 207 216 126 149 124  89
 124 202 165  85 213  36  81  45 111 169 209 149  89  95 143  37 123 169
 115 113 222  77  81  51 232 146  57  49 239 150 119 109  61 119 160  70
 150  70 248 209 134  99 155  64  78 134  38 212  82  67  80  43 149 172
  89 131 198 100 127   1 146 186  25  74  33 163 254 255 230 130 201 163
 225  88  72  32 176 129 138  25 235 230 216 129 143 248 106 208 235 181
  32  79 228 104 117  30 196  89  52  71 221 160 151  17 135 139  21 181
  62 144 183 116  11 122 221  76 189  83 210  30 119   0 200 250  99 232
 224 121 119  32   6 196]
```







Se repite el procedimiento pero ahora utilizando un filtro adaptado que implica filtrar la senial con el pulso invertido (flip). Se pueden ver que los resultados son superiores a las otras dos tecnicas utilizadas previamente. Para la eleccion del umbral de decision entre uno y cero se utilizo en valor cero, dado que se esta haciendo pasar la senial por un filtro pasa altos, eliminandose la componente de continua y ondulaciones de muy baja frecuencia relativa a la posible desintonia entre los pulsos unos y ceros. Otra tecnica podria haber sido la media de la senial, que se ha probado con similares resultados

Conclusion: Se utilizaron 3 sistemas de filtrado para una senial modulada sumergida en ruido. El primero utilizando un simple umbral, con resultados pobres, el segundo utilizando una correlacion sincronizada con cada pulso y deciendo por el de mayor verosimilitud, y finalmente un filtro adaptado optimo utilizando la correlacion de la senial con el pulso invertido en tiempo y a la vista de los resultados se puede verificar el mayor rendimiento en este ultimo.