# Sinusoidal plus Residual Modeling

***Xavier Serra***
Music Technology Group
Universitat Pompeu Fabra, Barcelona
*http://mtg.upf.edu*

# Index

- Sinusoidal plus residual model
- Sinusoidal subtraction
- Stochastic model
- Stochastic approximation of residual
- Sinusoidal plus stochastic model
- Implementation

# Sinusoidal plus residual model

$$y[n]=\sum_{r=1}^{R} A_r[n]\cos(2\pi f_r[n]n)+yr[n]=ys[n]+yr[n]$$

$R$: number of sinusoidal components

$A_r[n]$: instantaneous amplitude

$f_r[n]$: instantaneous frequency

$yr[n]$: residual component

$ys[n]$: sinusoidal component

when yr[n] is an stochastic signal, it can be modeled as filtered white noise:

$$yr_l[n]=yst_l[n]=\sum_{k=0}^{N-1} u[n]h_l[n-k]$$

$u[n]$: white noise

$h[n]$: impulse response of filter approximating residual component

$l$: frame number

otherwise: $$yr[n]=x[n]-\sum_{r=1}^{R} A_r[n]\cos(2\pi f_r[n]n)=x[n]-ys[n]$$

# Spectral view of SpR model

$$Y_l[k]=\sum_{r=1}^{R_l} A_{(r,l)} W[k-\hat{f}_{(r,l)}]+Yr_l[k]=Ys_l[k]+Yr_l[k]$$

$W$ : spectrum of analysis window

$R_l$ : number of sinusoidal components

$A_{(r,l)}$ : Amplitude of sinusoid

$\hat{f}_{(r,l)}$ : Normalized frequency of sinusoid

$Yr_l$ : residual component spectrum

$Ys_l$ : sinusoidal component spectrum

$l$ : frame number

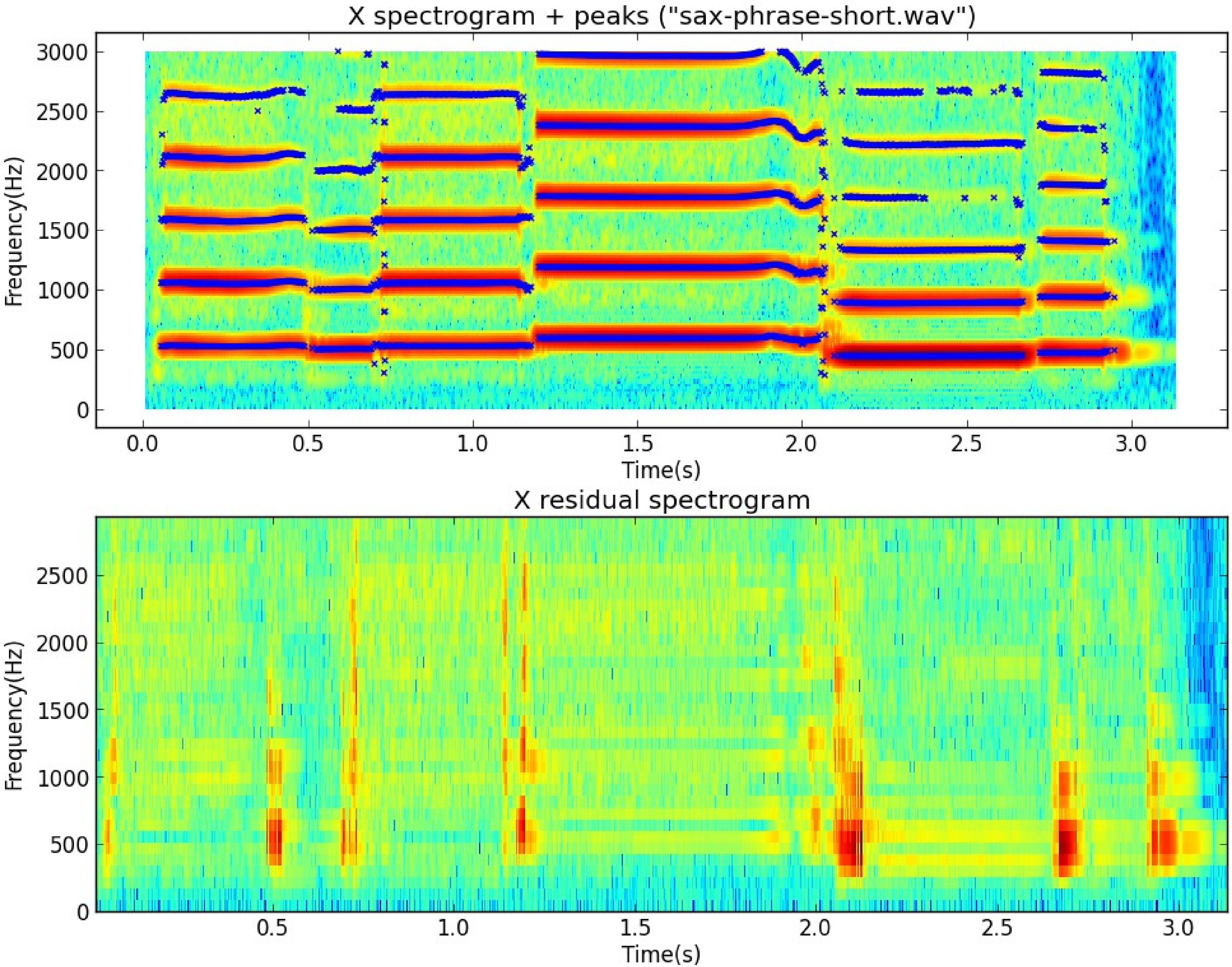
when e[n] is an stochastic signal, it can be modeled as filtered white noise:

$$Yr_l[k]=Yst_l[k]=U[k]H_l[k]$$

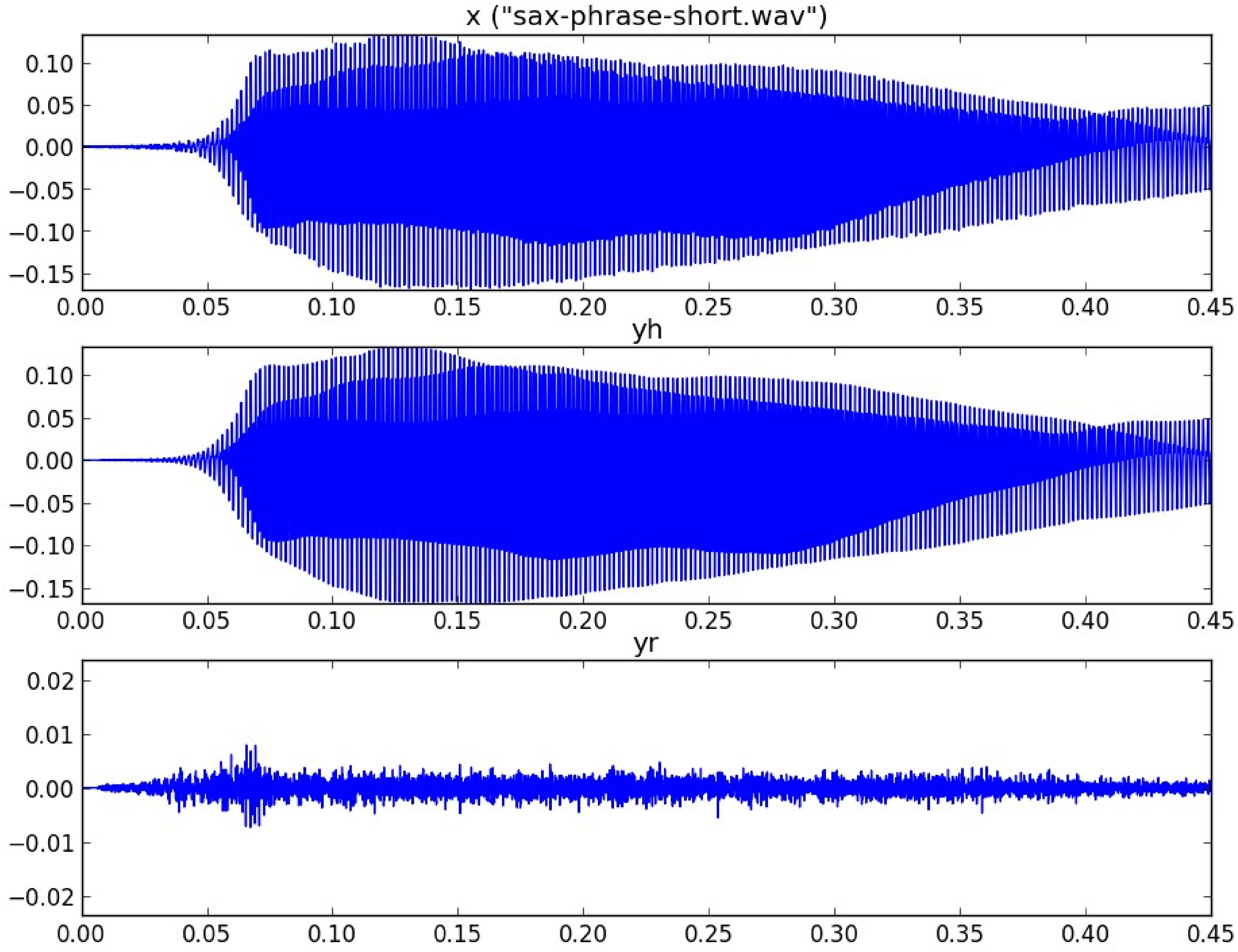
$U$ : white noise spectrum

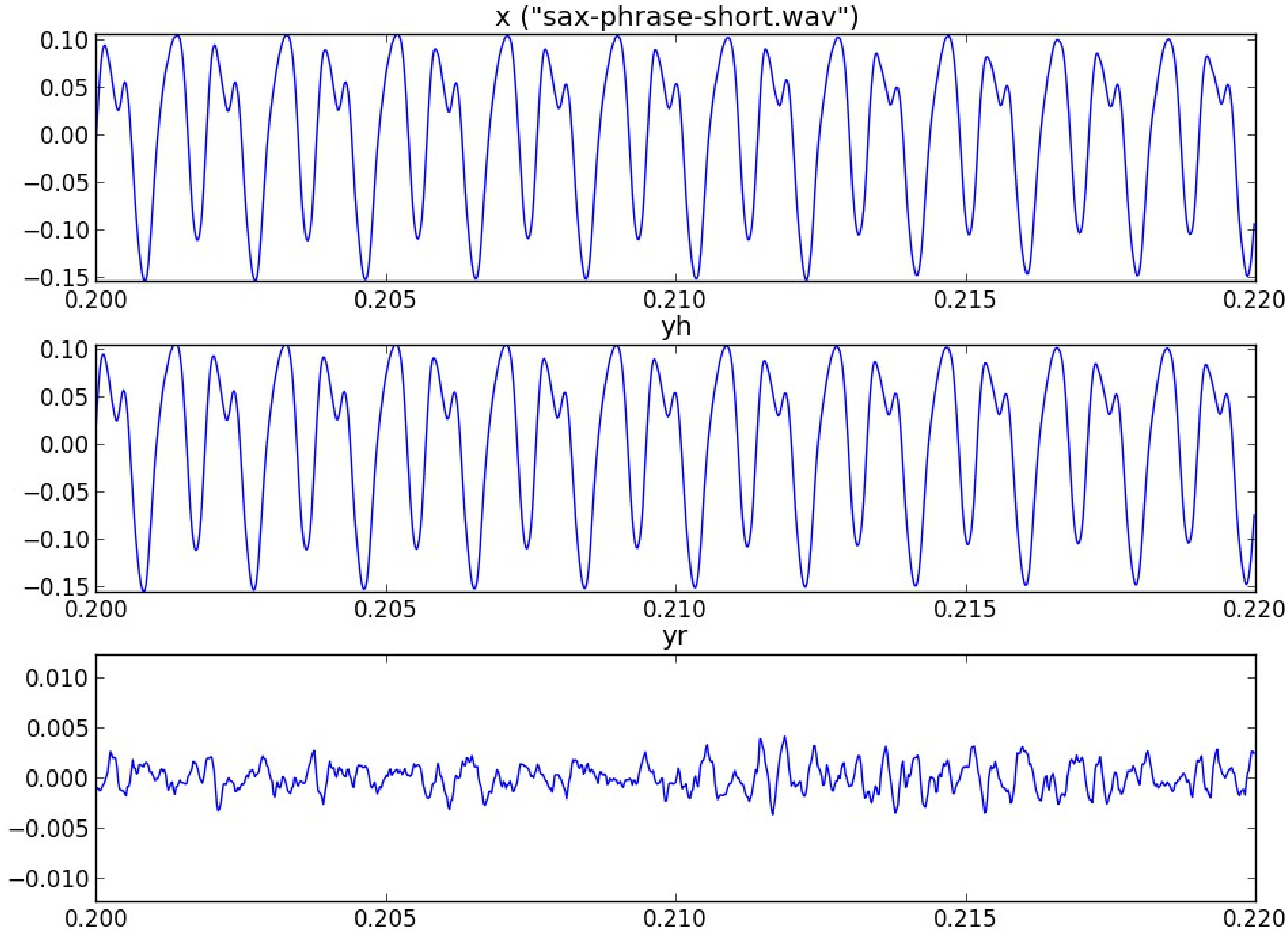
$H_l$ : frequency response of filter apporximating residual component

otherwise: $Yr_l[k]=X_l[k]-Ys_l[k]$

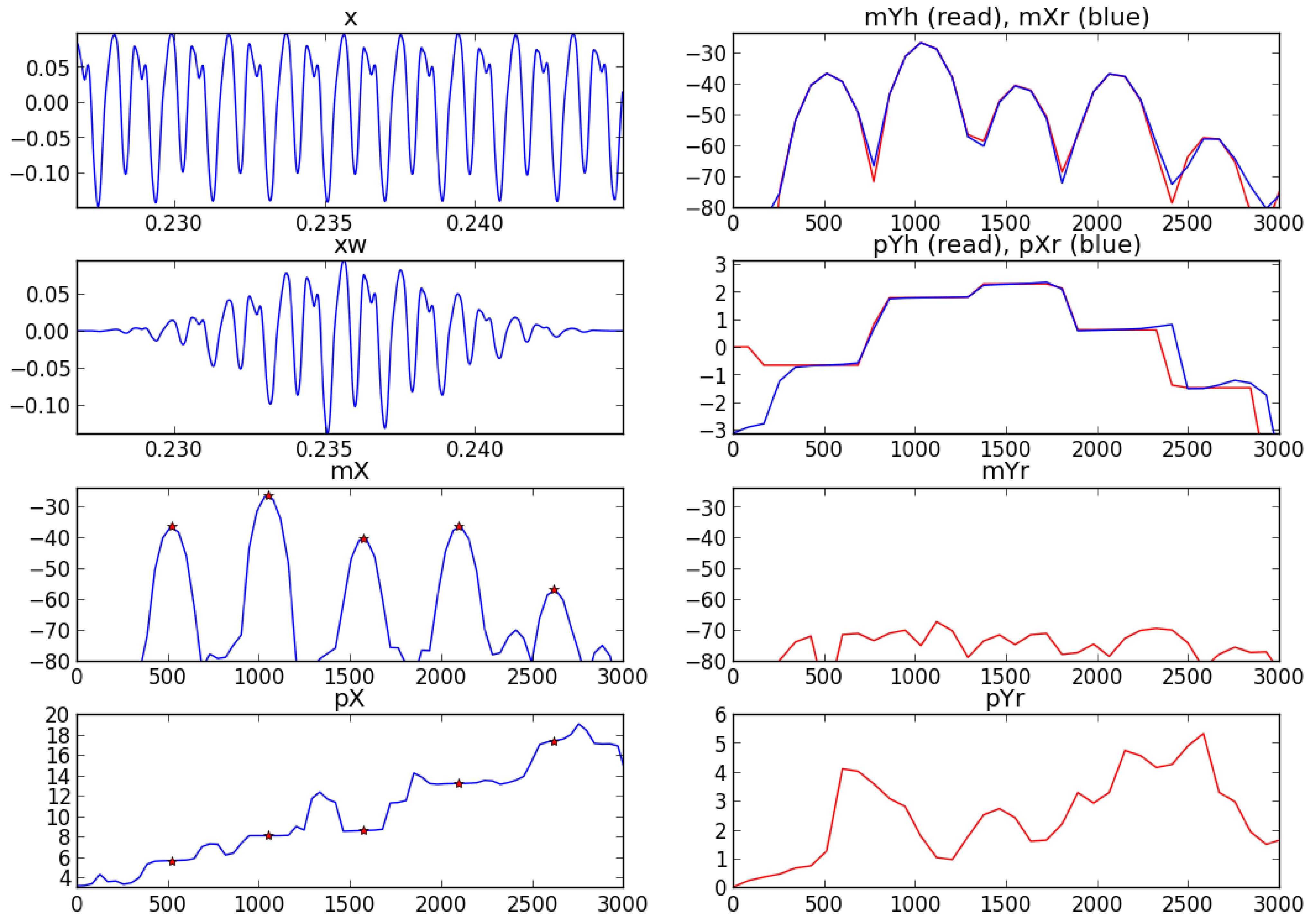


X spectrogram + peaks ("sax-phrase-short.wav")
Frequency(Hz)
0
500
1000
1500
2000
2500
3000
Time(s)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
X residual spectrogram
Frequency(Hz)
0
500
1000
1500
2000
2500
Time(s)
0.5
1.0
1.5
2.0
2.5
3.0


x ("sax-phrase-short.wav")
yh
yr
0.10
0.05
0.00
−0.05
−0.10
−0.15
0.02
0.01
−0.01
−0.02
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45



x ("sax-phrase-short.wav")
yh
yr
0.10
0.05
0.00
−0.05
−0.10
−0.15
0.010
0.005
0.000
−0.005
−0.010
0.200
0.205
0.210
0.215
0.220



x
0.05
0.00
−0.05
−0.10
0.230
0.235
0.240
mYh (read), mXr (blue)
−30
−40
−50
−60
−70
−80
0
500
1000
1500
2000
2500
3000
xw
pYh (read), pXr (blue)
3
2
1
0
−1
−2
−3
mX
mYr
pX
20
18
16
14
12
10
8
6
4
pYr
6
5
4
3
2
1
0

# Stochastic signals

- Stochastic processes
  - described by the laws of probability, mean, variance, probability distributions

- Autocorrelation

$$Z_{xx}[k] = \sum_{n=0}^{n=N-1} x[n]x[n+k] \qquad k = -N+1, \ldots, N-1$$

- Power spectral density

$$Xp[k] = \lim_{N \to \infty} |X[k]|^2$$

$$\text{where } X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \qquad k = 0, \ldots, N-1$$

# Stochastic model

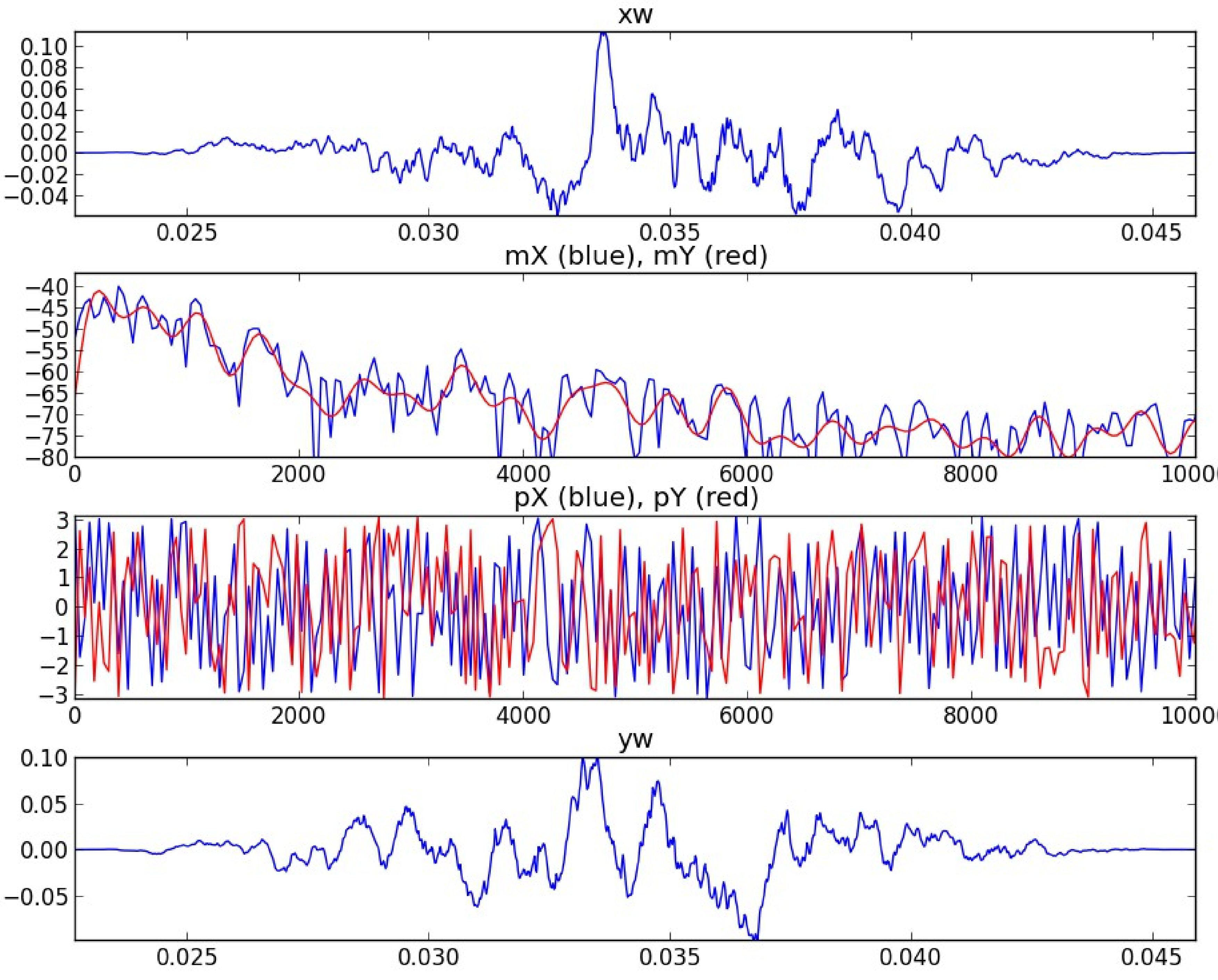
$$yst[n] = \sum_{k=0}^{N-1} u[n] h[n-k]$$

$u[n]$: white noise
$h[n]$: impulse response of filter approximating input signal $x[n]$

Spectral view:

$$Yst_l[k] = |H_l[k]||U[k]| e^{(\sphericalangle H[k] + \sphericalangle U[k])} = |\tilde{X}_l[k]| e^{\sphericalangle U[k]}$$

$|\tilde{X}[k]|$: approximation of magnitude spectrum of input signal $x[n]$
$\sphericalangle U[k]$: spectral phases of noise signal
$l$: frame number



xw
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
0.025
0.030
0.035
0.040
0.045
mX (blue), mY (red)
−40
−45
−50
−55
−60
−65
−70
−75
−80
0
2000
4000
6000
8000
10000
pX (blue), pY (red)
3
2
1
0
−1
−2
−3
0
2000
4000
6000
8000
10000
yw
0.10
0.05
0.00
−0.05
0.025
0.030
0.035
0.040
0.045

# Filter approximation

LPC model: $y[n]=\sum_{k=1}^{K} a_k x[n-k]+Au[n]$

$$\hat{x}[n]=-\sum_{k=1}^{K} a_k x[n-k]$$

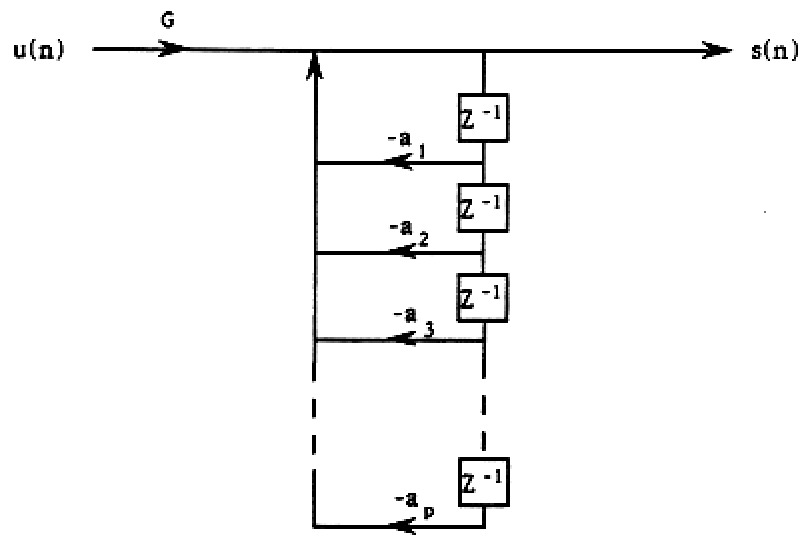
$$e[n]=x[n]-\hat{x}[n]=x[n]+\sum_{k=1}^{K} a_k x[n-k]$$

$$E=\sum_{n=-\infty}^{\infty} e[n]^2=\sum_{n=-\infty}^{\infty} \left(x[n]+\sum_{k=1}^{K} a_k x[n-k]\right)^2$$

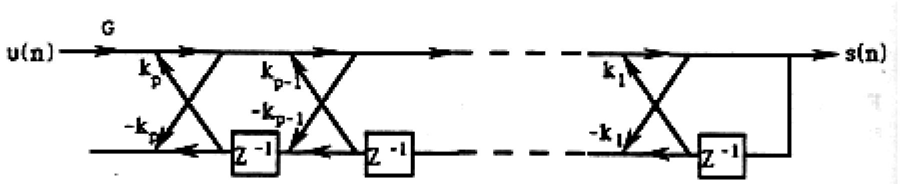
The error is minimized by minimizing the mean of the total squared error with respect to each of the parameters.

# Stochastic synthesis using filters

- Noise generation:
  - Algorithms for random number generation
  - White noise, other types of noises
  - Gaussian noise

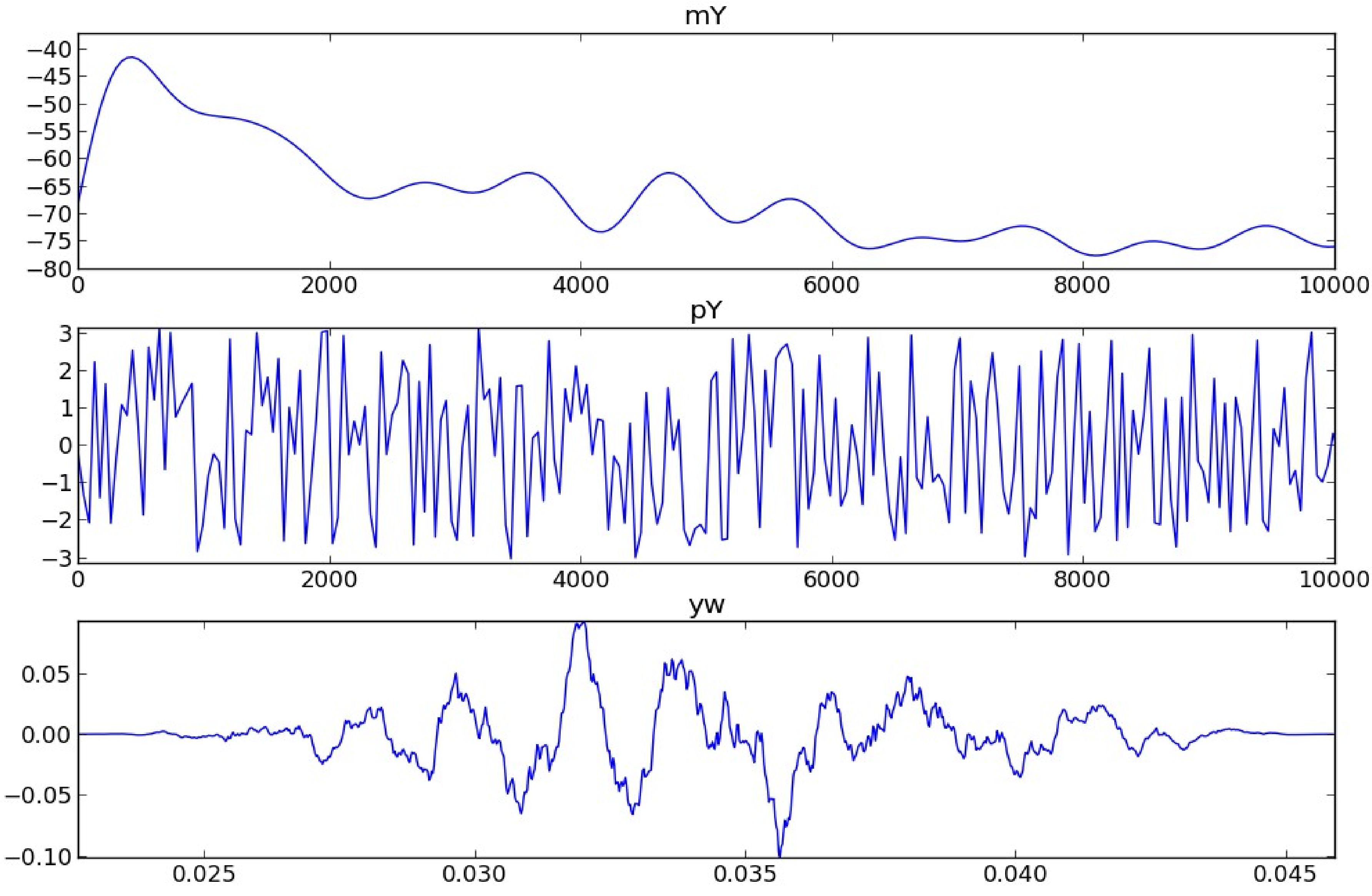



direct form structure




lattice structure

# Stochastic synthesis using IFFT



```
def stochasticModel(x, w, N, H, stocf) :
  hN = N/2
  hM = (w.size)/2
  pin = hM
  pend = x.size-hM
  yw = np.zeros(w.size)
  y = np.zeros(x.size)
  w = w / sum(w)
  ws = hanning(w.size)*2
  while pin<pend:
    xw = x[pin-hM:pin+hM] * w
    X = fft(xw)
    mX = 20 * np.log10( abs(X[:hN]) )
    mXenv = resample(np.maximum(-200, mX), mX.size*stocf)
    mY = resample(mXenv, hN)
    pY = 2*np.pi*np.random.rand(hN)
    Y[:hN] = 10**(mY/20) * np.exp(1j*pY)
    Y[hN+1:] = 10**(mY[:0:-1]/20) * np.exp(-1j*pY[:0:-1])
    fftbuffer = np.real(ifft(Y))
    y[pin-hM:pin+hM] += H*ws*fftbuffer
    pin += H
  return y
```

# Sinusoidal plus Stochastic model

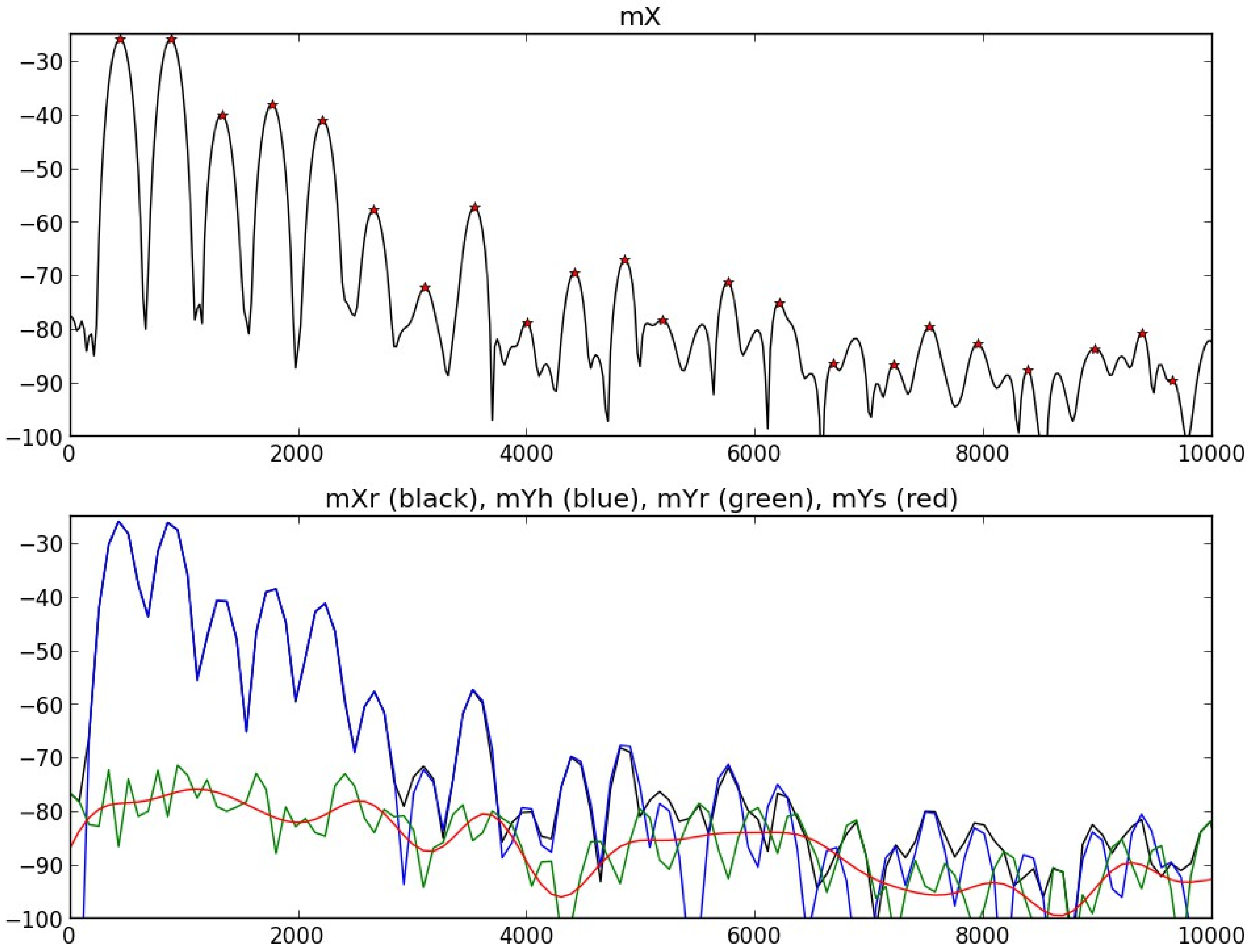
$$y[n] = \sum_{r=1}^{R} A_r[n]\cos(2\pi f_r[n]n) + yst[n]$$

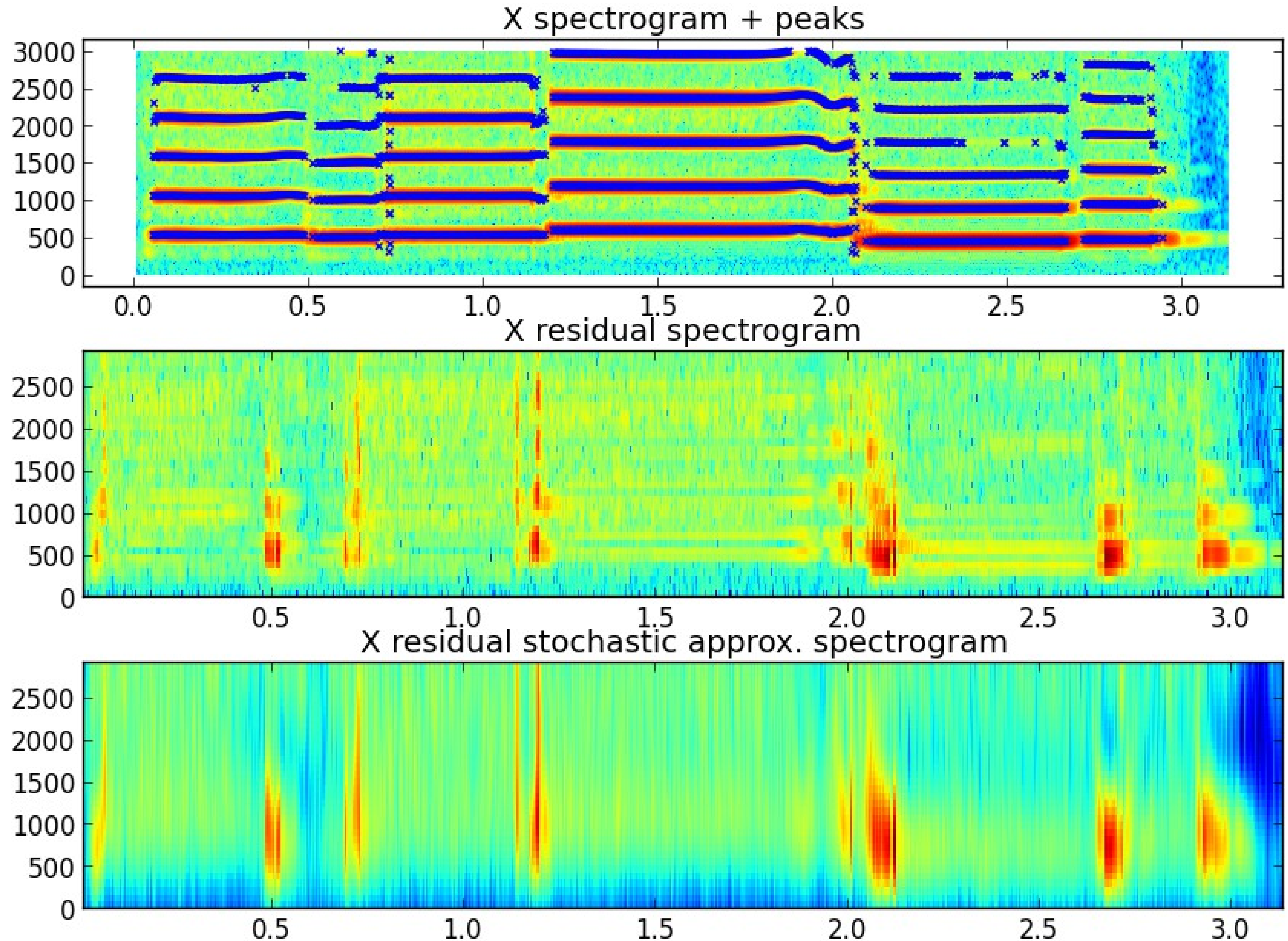
$R$: number of sinusoidal components
$A_r[n]$: instantaneous amplitude
$f_r[n]$: instantaneous frequency
$yst[n]$: stochastic signal

Spectral view:

$$Yst_l[k] = |\tilde{Y}r_l[k]| e^{\sphericalangle U[k]}$$

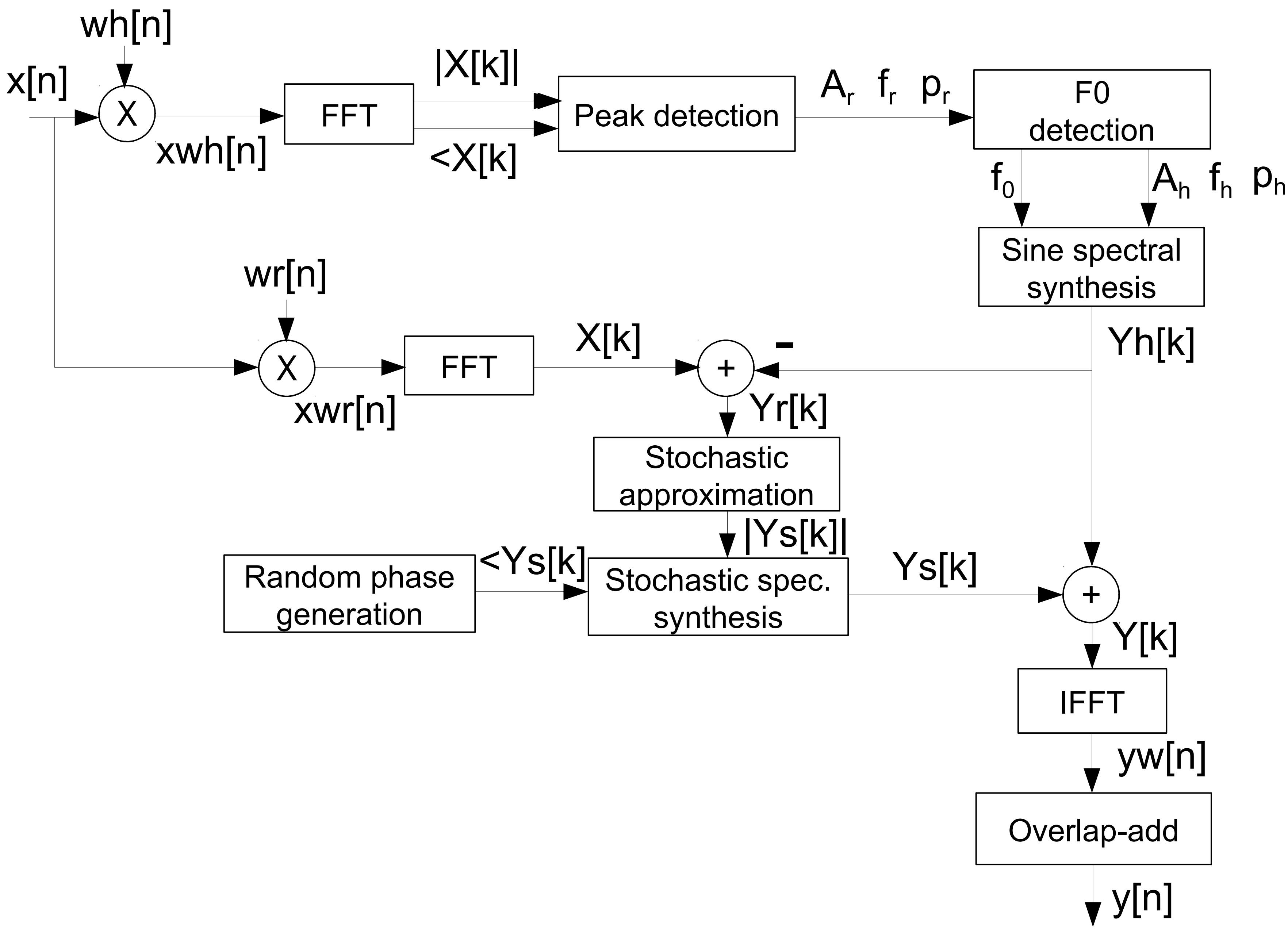
$|\tilde{Y}r_l[k]|$: approximation of magnitude spectrum of input signal
$\sphericalangle U[k]$: spectral phases of noise
$l$: frame number


mX
−30
−40
−50
−60
−70
−80
−90
−100
0
2000
4000
6000
8000
10000
mXr (black), mYh (blue), mYr (green), mYs (red)
−30
−40
−50
−60
−70
−80
−90
−100
0
2000
4000
6000
8000
10000



X spectrogram + peaks
3000
2500
2000
1500
1000
500
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
X residual spectrogram
2500
2000
1500
1000
500
0
0.5
1.0
1.5
2.0
2.5
3.0
X residual stochastic approx. spectrogram
2500
2000
1500
1000
500
0
0.5
1.0
1.5
2.0
2.5
3.0

# Implementation: HpS model



```
xw = x[pin-hM1:pin+hM2] * w
fftbuffer[:hM1] = xw[hM2:]
fftbuffer[N-hM2:] = xw[:hM2]
X = fft(fftbuffer)
mX = 20 * np.log10(abs(X[:hN]))
ploc = PP.peakDetection(mX, hN, t)
pX = np.unwrap(np.angle(X[:hN]))
iploc, ipmag, ipphase = PP.peakInterp(mX, pX, ploc)
iploc = (iploc!=0) * (iploc*Ns/N)
ri = pin-hNs-1
xr = x[ri:ri+Ns]*wr
fftbuffer[:hNs] = xr[hNs:]
fftbuffer[hNs:] = xr[:hNs]
Xr = fft(fftbuffer)
Ys = GS.genSpecSines(iploc, ipmag, ipphase, Ns)
Yr = Xr-Ys;
fftbuffer = np.real(ifft(Ys))
ysw[:hNs-1] = fftbuffer[hNs+1:]
ysw[hNs-1:] = fftbuffer[:hNs+1]
fftbuffer = np.real(ifft(Yr))
yrw[:hNs-1] = fftbuffer[hNs+1:]
yrw[hNs-1:] = fftbuffer[:hNs+1]
```

# References


- https://ccrma.stanford.edu/~jos/sasp/Spectrum_Analysis_Sinusoids.html
- http://en.wikipedia.org/wiki/Stochastic_process
- http://en.wikipedia.org/wiki/Linear_predictive_coding
- Sounds: http://www.freesound.org/people/xserra/packs/13038/

# Credits