Sinusoidal Modeling

Xavier Serra

Music Technology Group

Universitat Pompeu Fabra, Barcelona

http://mtg.upf.edu

Index

- Sinusoidal model
- Sinewave spectrum
- Sinusoidal detection
  - Peak detection
  - Peak interpolation
- Sinusoidal synthesis
- Implementation

Sinusoidal model

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

R: number of sinewave components
$A_r[n]$: instantaneous amplitude
$f_r[n]$: instantaneous frequency

Sinewave spectrum

$$x[n] = A\cos(2\pi k_0 n/N + \varphi)$$

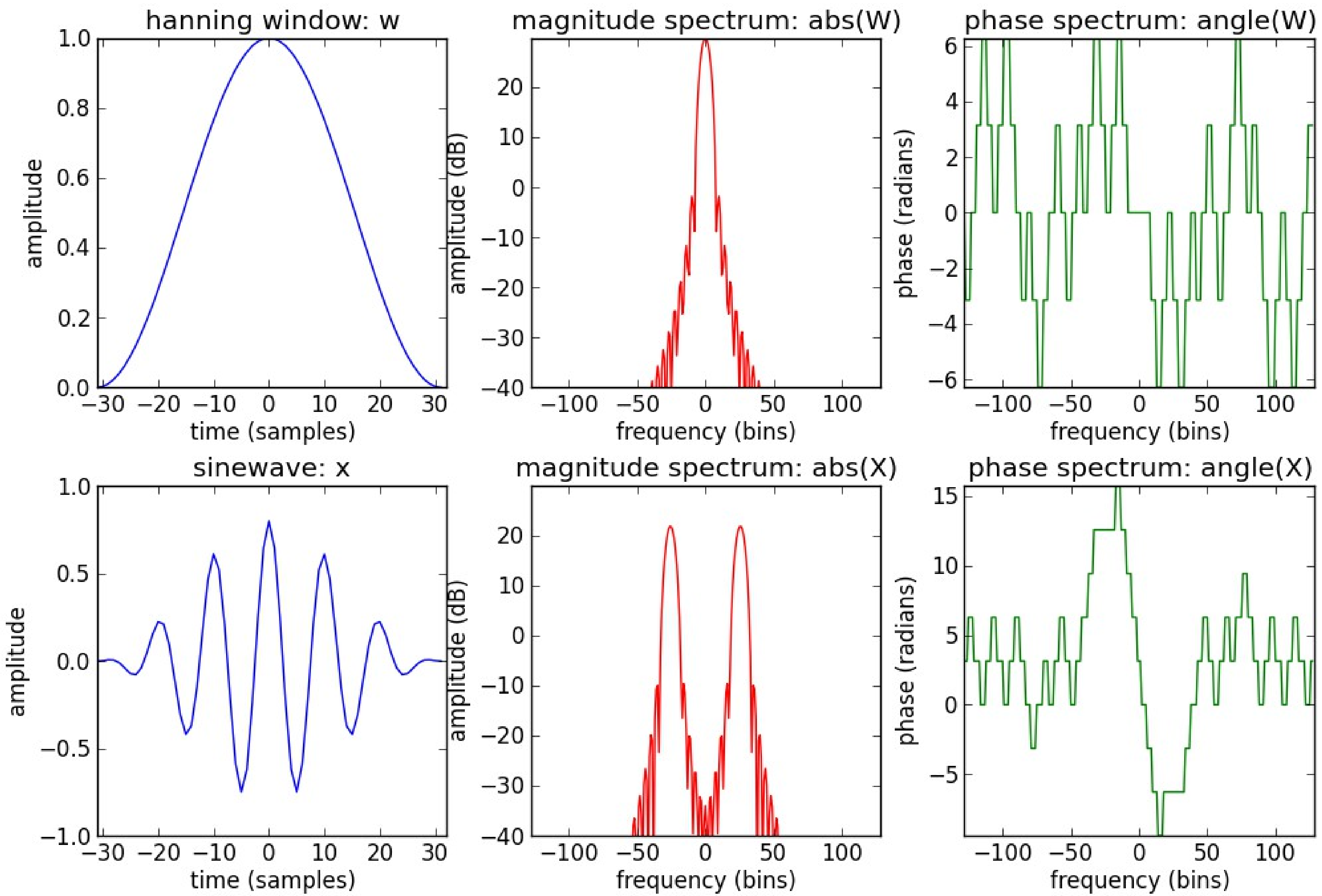
$$X[k] = A\sum_{n=0}^{N-1} w[n]\frac{1}{2}\left(e^{j2\pi k_0 n/N} + e^{-j2\pi k_0 n/N}\right)e^{-j2\pi kn/N}$$

$$= \frac{A}{2}\sum_{n=0}^{N-1} w[n]e^{j2\pi k_0 n/N}e^{-j2\pi kn/N} + \frac{A}{2}\sum_{n=0}^{N-1} w[n]e^{-j2\pi k_0 n/N}e^{-j2\pi kn/N}$$

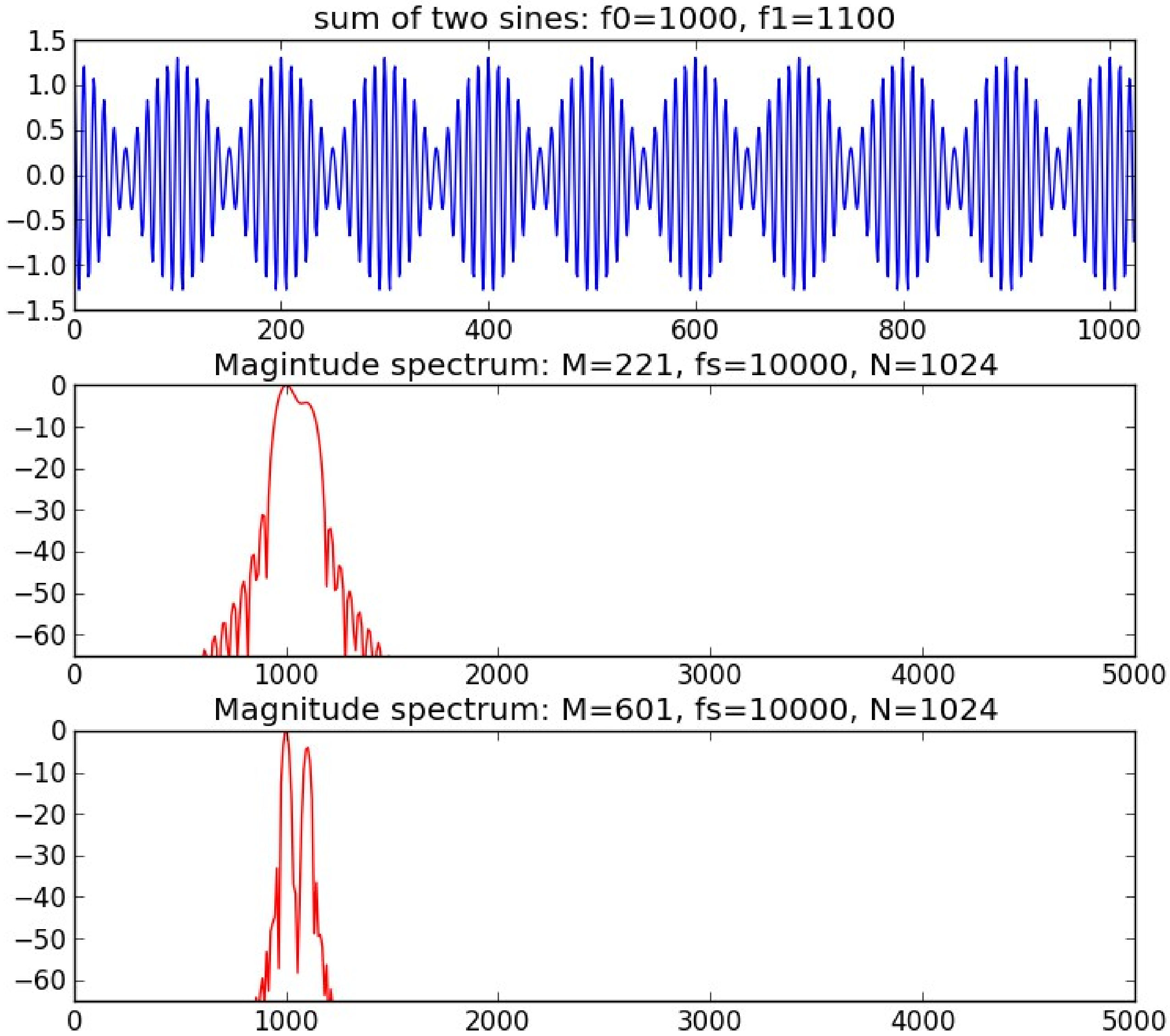
$$= \frac{A}{2}\sum_{n=0}^{N-1} w[n]e^{-j2\pi(-k_0+k)n/N} + \frac{A}{2}\sum_{n=0}^{N-1} w[n]e^{-j2\pi(k_0+k)n/N}$$

$$= \frac{A}{2}W[-k_0+k] + \frac{A}{2}W[k_0+k]$$

Sinewave spectrum



Sinusoidal detection – freq. resolution

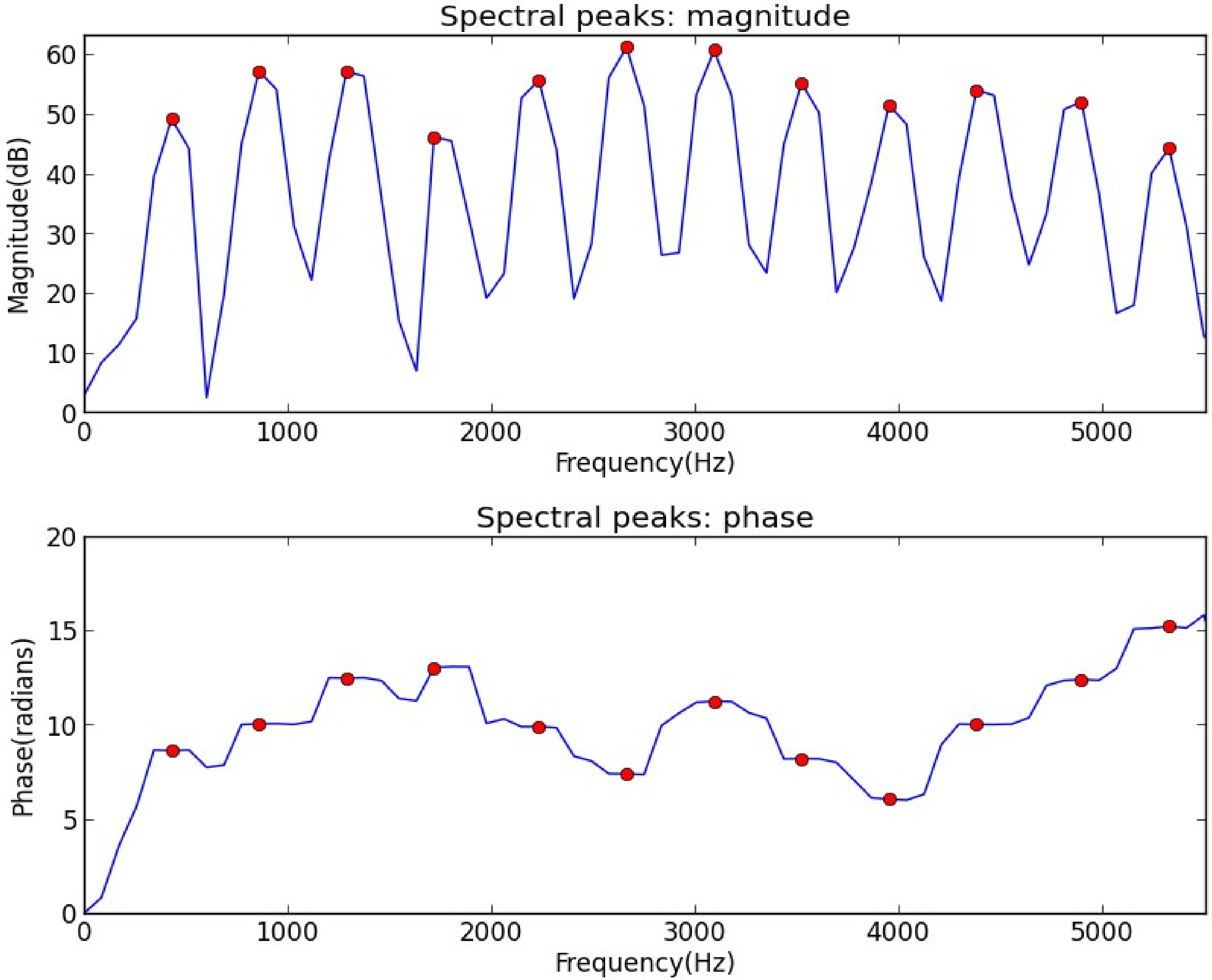


```
N = 1024
M1 = 221
M2 = 601
f0 = 1000
f1 = 1100
fs = 10000
A0 = .8
A1 = .5
hN = N/2
w1 = np.hanning(M1)
w2 = np.hanning(M2)
x = A0*np.cos(2*np.pi*f0/fs*np.arange(N))+
    A1*np.cos(2*np.pi*f1/fs*np.arange(N))

plt.subplot(3,1,1)
plt.plot(np.arange(N), x, 'b')

X = fft(x[0:M1]*w1, N)
mX = 20*np.log10(abs(X[0:hN]))
plt.subplot(3,1,2)
plt.plot((np.arange(hN)/float(N))*fs, mX-max(mX), 'r')

X = fft(x[0:M2]*w2, N)
mX = 20*np.log10(abs(X[0:hN]))
plt.subplot(3,1,3)
plt.plot((np.arange(hN)/float(N))*fs, mX-max(mX), 'r')
```

Peak detection

- A peak is defined as a local maximum in the magnitude spectrum.
- Each peak is accurate only to within half a sample.
- Zero-padding increases the number of DFT bins per Hz and thus increases the accuracy of peak detection.
- A better peak detection strategy is based on spectral interpolation.

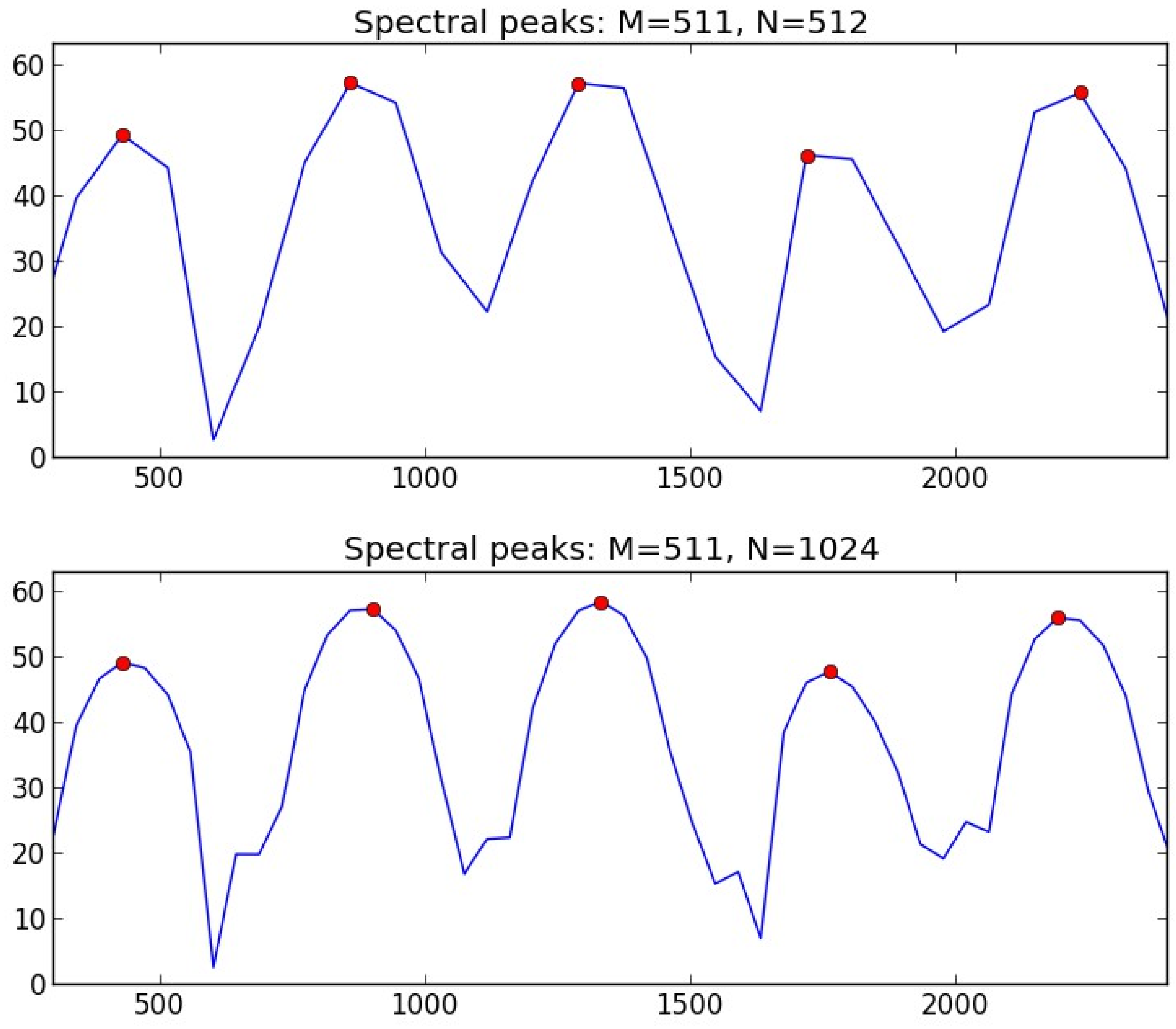
Peak detection



Peak detection

```
def peak_detection(mX, hN, t):
  # mX: magnitude spectrum, hN: size of positive spectrum,
  # t: threshold

  thresh = np.where(mX[1:hN-1]>t, mX[1:hN-1], 0)
  next_minor = np.where(mX[1:hN-1]>mX[2:], mX[1:hN-1], 0)
  prev_minor = np.where(mX[1:hN-1]>mX[:hN-2], mX[1:hN-1], 0)
  ploc = thresh * next_minor * prev_minor
  ploc = ploc.nonzero()[0] + 1

  return ploc
```

Peak detection with zero-padding



Peak detection and window-size

To resolve two sinusoids separated in frequency by ΔHz we require main-lobe bandwidth B_f <= Δ

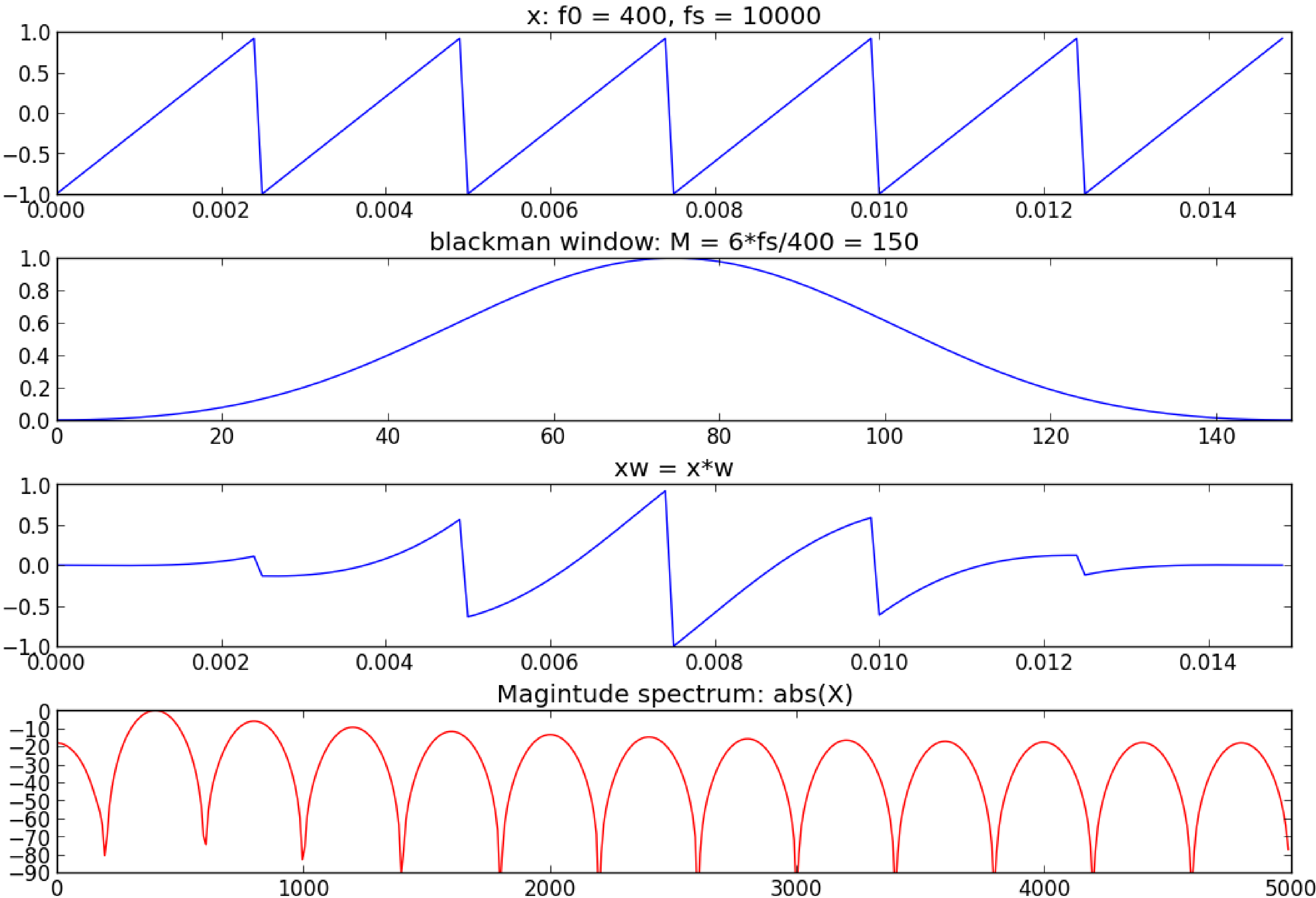
If $B_f = B_s f_s / M$ and $\Delta = \left| f_{k+1} - f_k \right|$

where B_s : main-lobe bandwidth in bins, f_s : sampling rate,
M: window length, f_k and f_{k+1} frequencies of the sinusoids.

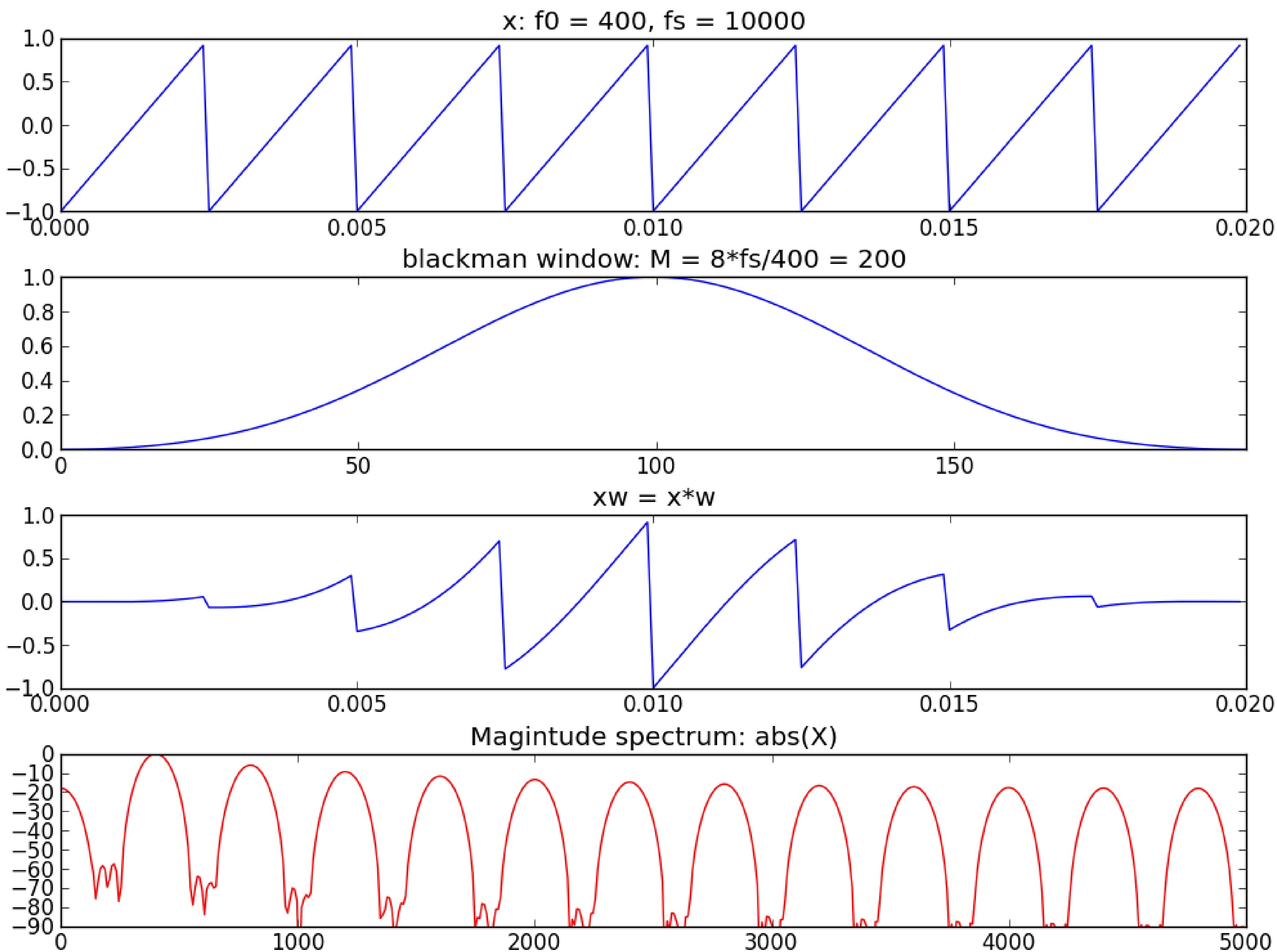
$$M \geq B_s \frac{f_s}{\Delta} = B_s \frac{f_s}{\left| f_{k+1} - f_k \right|}$$

If f_k and f_{k+1} are successive harmonics of a fundamental frequency f_1,
Then $f_1 = f_{k+1} - f_k = \Delta$. Harmonic resolution requires $B_f \leq f_1$ and
$M \geqslant B_s f_s / f_1$

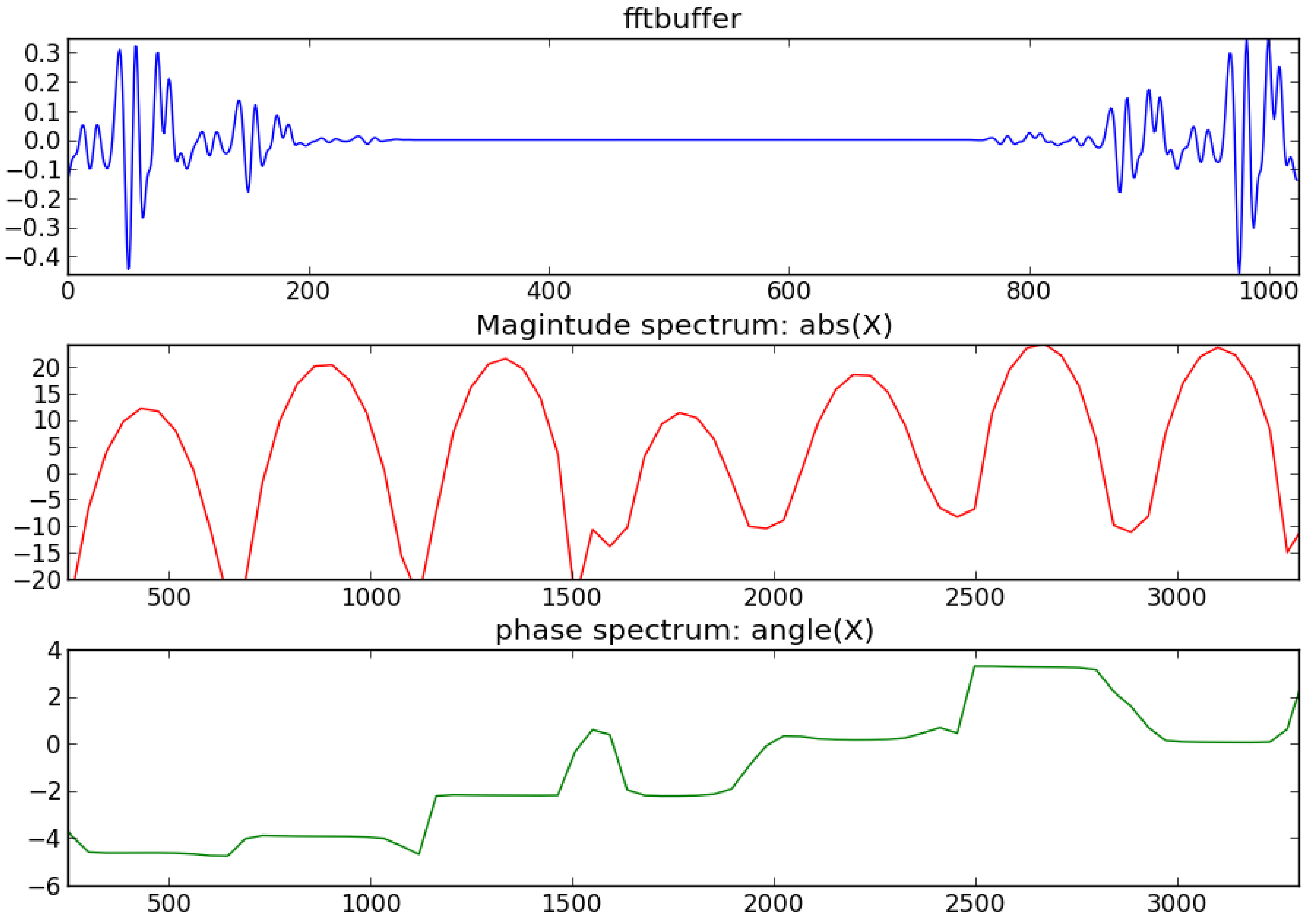
Since the period in samples is $P = f_s / f_1$, then $M \geqslant B_s P$


x: f0 = 400, fs = 10000
blackman window: M = 6*fs/400 = 150
xw = x*w
Magintude spectrum: abs(X)

```
N = 1024
hN = N/2
f0 = 400.0
fs = 10000.0
M = 6 * fs / f0
w = np.blackman(M)
x = signal.sawtooth(2 * np.pi * (f0/fs) * np.arange(M))

plt.figure(1)
plt.subplot(4,1,1)
plt.plot(np.arange(M)/fs, x, 'b')

plt.subplot(4,1,2)
plt.plot(np.arange(M), w, 'b')

xw = x * w
plt.subplot(4,1,3)
plt.plot(np.arange(M)/fs, xw, 'b')

X = fft(xw, N)
mX = 20*np.log10(abs(X[0:hN]))
plt.subplot(4,1,4)
plt.plot(fs*np.arange(hN)/N,mX-max(mX), 'r')
```


x: f0 = 400, fs = 10000
blackman window: M = 8*fs/400 = 200
xw = x*w
Magintude spectrum: abs(X)

Peak phase



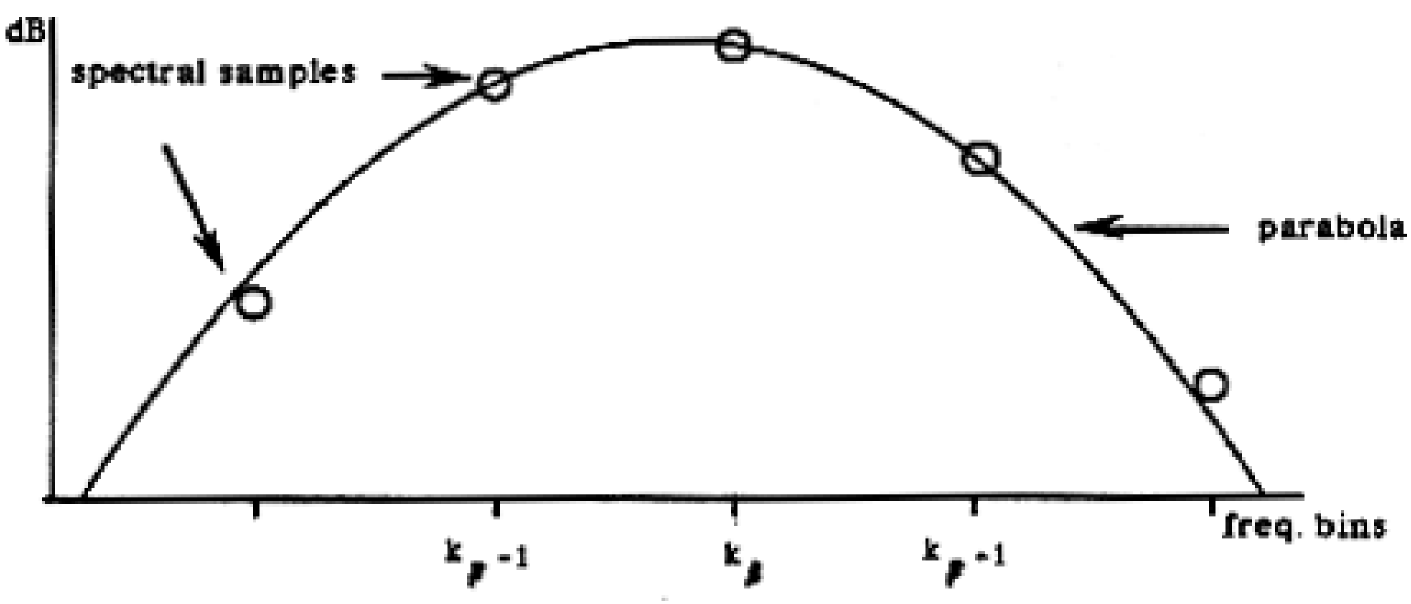
fftbuffer
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
200
400
600
800
1000
Magintude spectrum: abs(X)
20
15
10
5
0
−5
−10
−15
−20
500
1000
1500
2000
2500
3000
phase spectrum: angle(X)
4
2
0
−2
−4
−6
500
1000
1500
2000
2500
3000

```
N = 1024
hN = N/2
M = 601
hM = (M+1)/2
w = np.blackman(M)

(fs, x) = wp.wavread('oboe.wav')
xw = x[40000:40000+M] * w

fftbuffer = np.zeros(N)
fftbuffer[:hM] = xw[hM-1:]
fftbuffer[N-hM+1:] = xw[:hM-1]
plt.subplot(3,1,1)
plt.plot(np.arange(N), fftbuffer, 'b')

X = fft(fftbuffer)
mX = 20*np.log10(abs(X[:hN]))
plt.subplot(3,1,2)
plt.plot(fs*np.arange(hN)/float(N),mX, 'r')

pX = np.unwrap(np.angle(X[0:hN]))
plt.subplot(3,1,3)
plt.plot(fs*np.arange(hN)/float(N),pX, 'g')
```

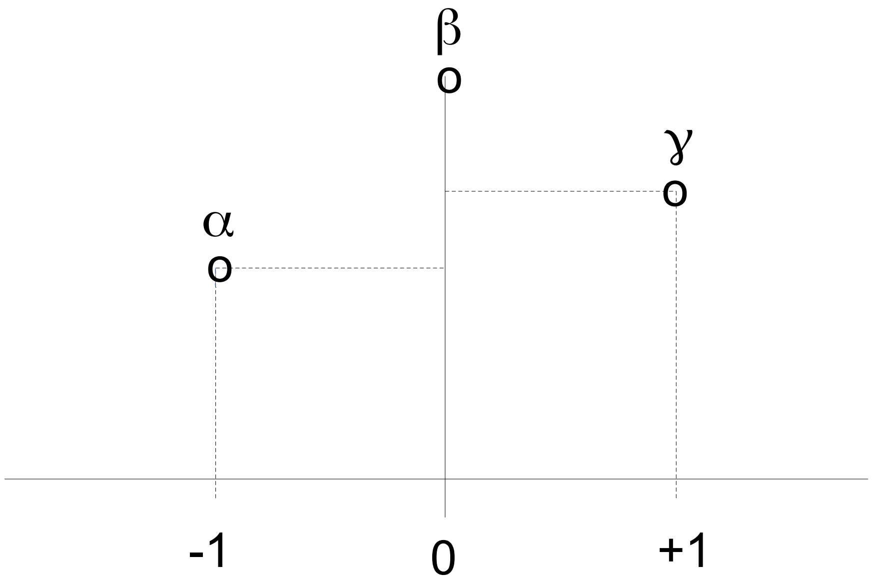
Parabola

$$y(x) = a(x - p)^2 + b$$

p: center of the parabola
a: measure of concavity
b: offset



Peak interpolation



$$y(-1) = \alpha = 20\log_{10}\left|X\left(k_\beta - 1\right)\right|,$$

$$y(0) = \beta = 20\log_{10}\left|X\left(k_\beta\right)\right|,$$

$$y(1) = \gamma = 20\log_{10}\left|X\left(k_\beta + 1\right)\right|,$$

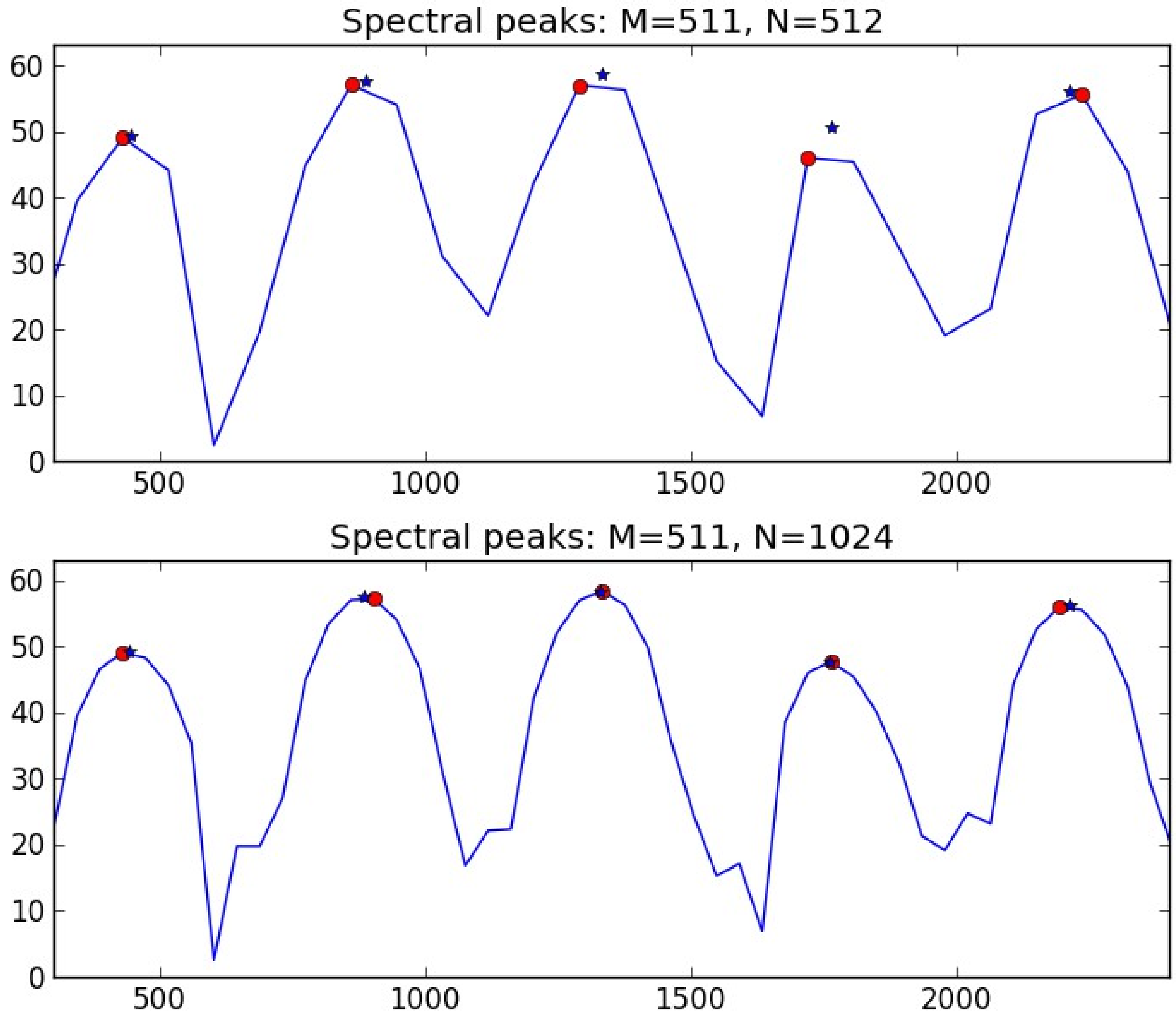
Center of the parabola: $\hat{k}_p = \hat{k} + \frac{\alpha - \gamma}{2}(\alpha - 2\beta + \gamma)$

Amplitude: $\hat{a} = \beta - \frac{\hat{k}_p}{4}(\alpha - \gamma)$

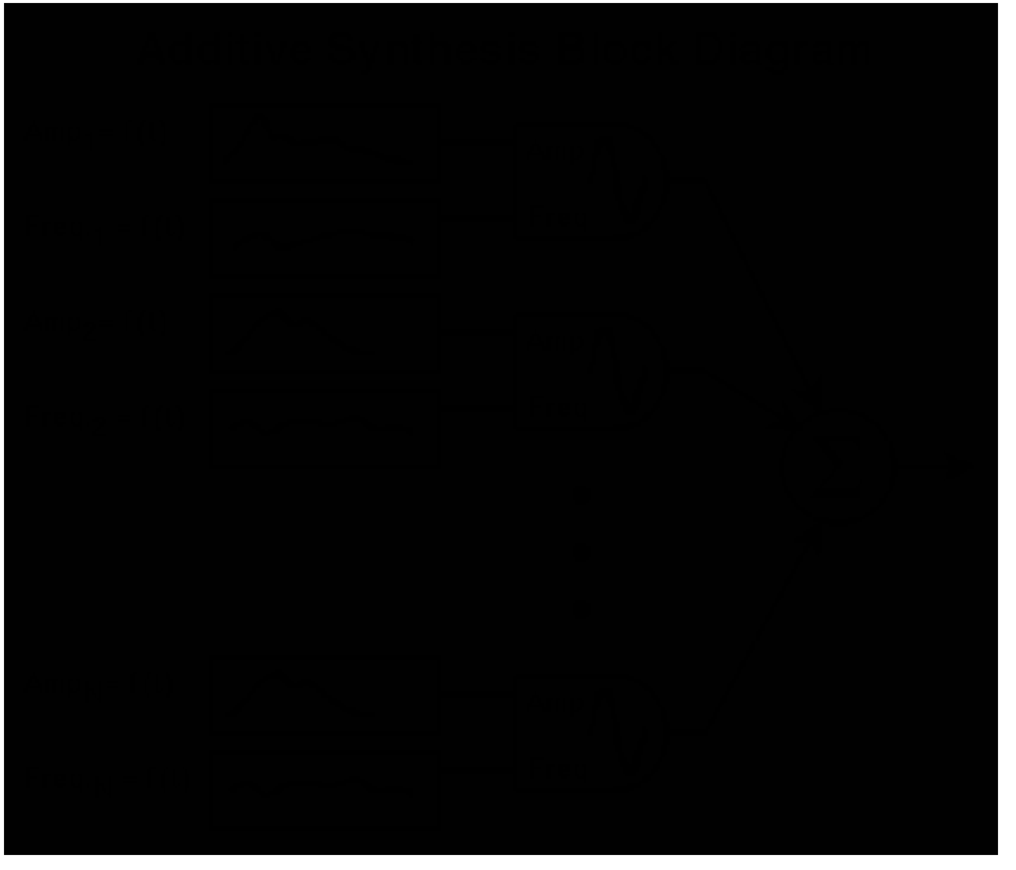
Peak interpolation

```
def peak_interp(mX, pX, ploc):
  # mX: magnitude spectrum, pX: phase spectrum,
  # ploc: locations of peaks
  # iploc, ipmag, ipphase: interpolated values

  val = mX[ploc]
  lval = mX[ploc-1]
  rval = mX[ploc+1]
  iploc = ploc + 0.5*(lval-rval)/(lval-2*val+rval)
  ipmag = val - 0.25*(lval-rval)*(iploc-ploc)
  ipphase = np.interp(iploc, np.arange(0, pX.size), pX)

  return iploc, ipmag, ipphase
```

Peak detection with interpolation



Sinusoidal synthesis

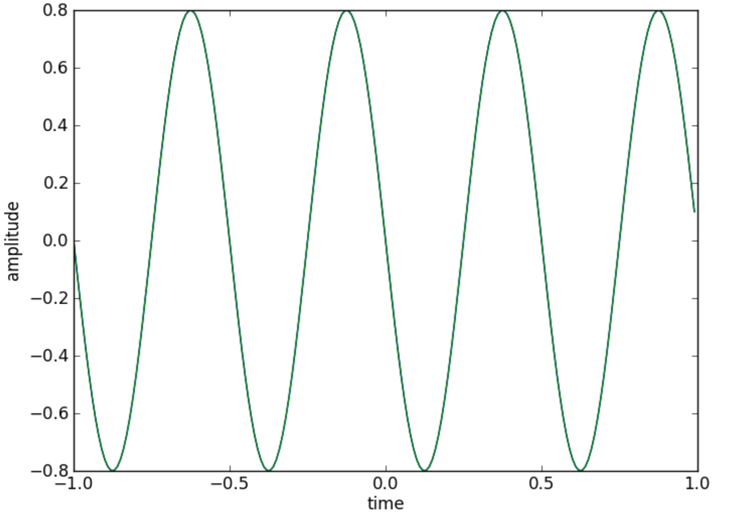


Sinusoidal synthesis

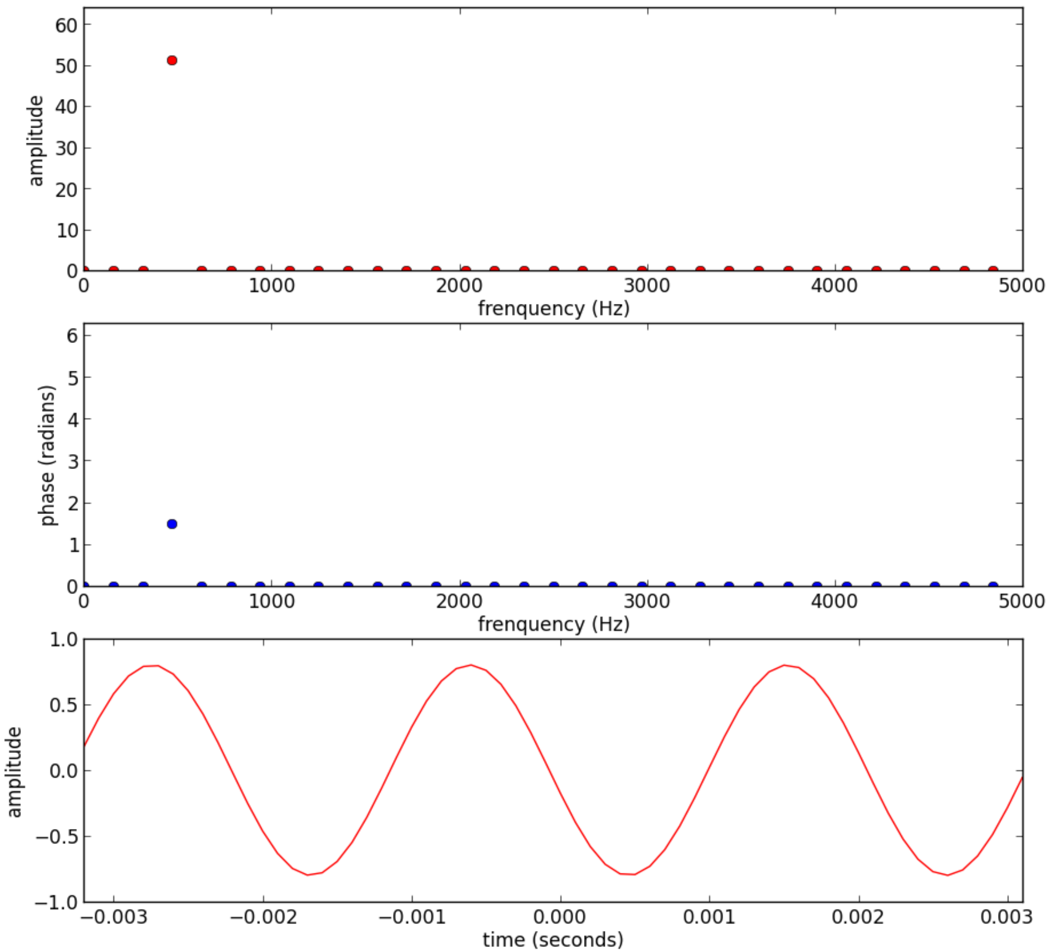
$$y[n] = A_0[n]\cos\left(2\pi f_0[n]n + \varphi_0\right)$$

$A_0[n]$: instantaneous amplitude
$f_0[n]$: instantaneous frequency
φ_0: initial phase

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

A = .8
F0 = 2.0
phi = np.pi/2
fs = 100
t = np.arange(-1, 1, 1.0/fs)
x = A * np.cos(2*np.pi*f0*t+phi)
plt.plot(t, x)
```

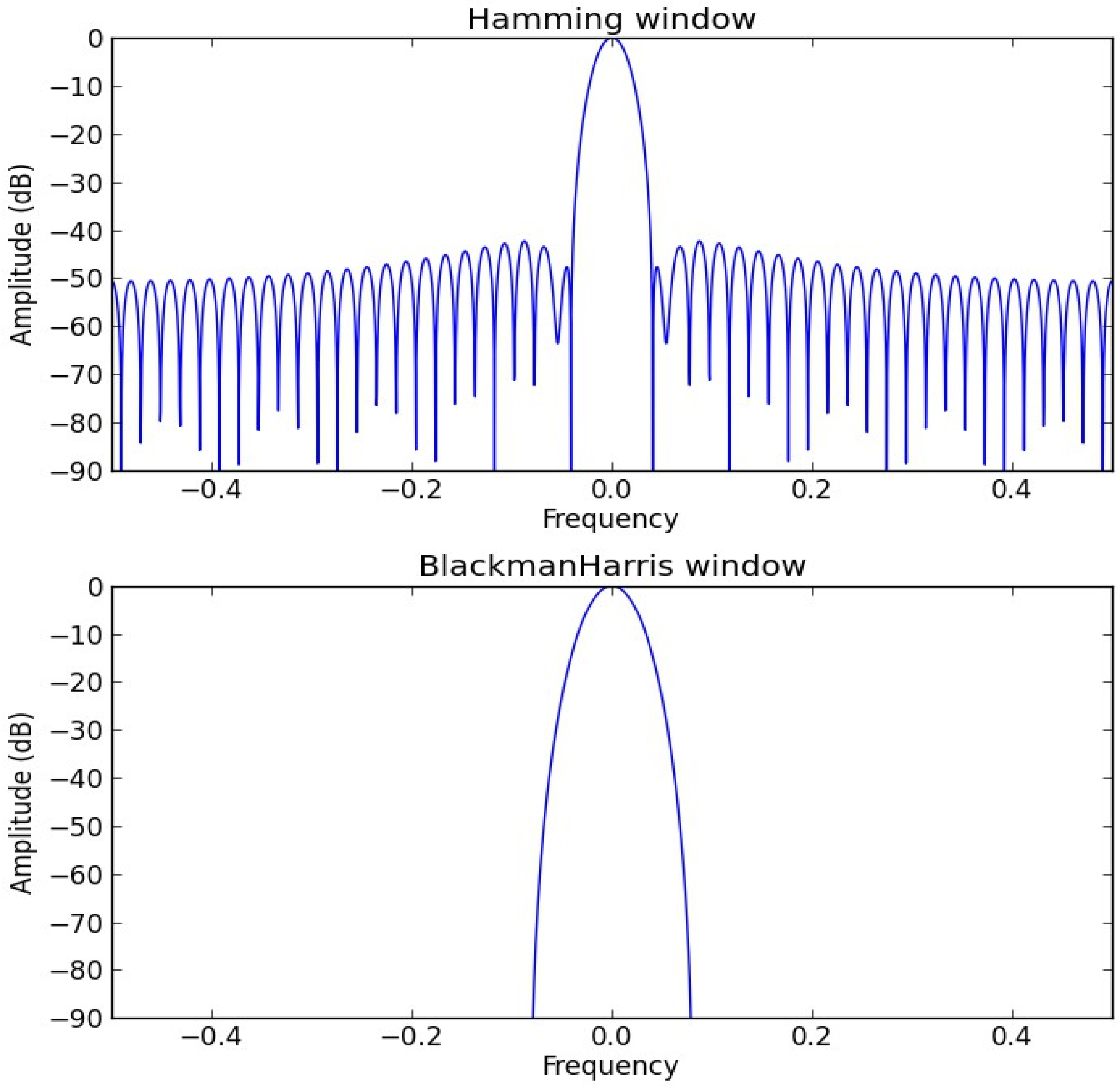


Spectral-based sinusoidal synthesis

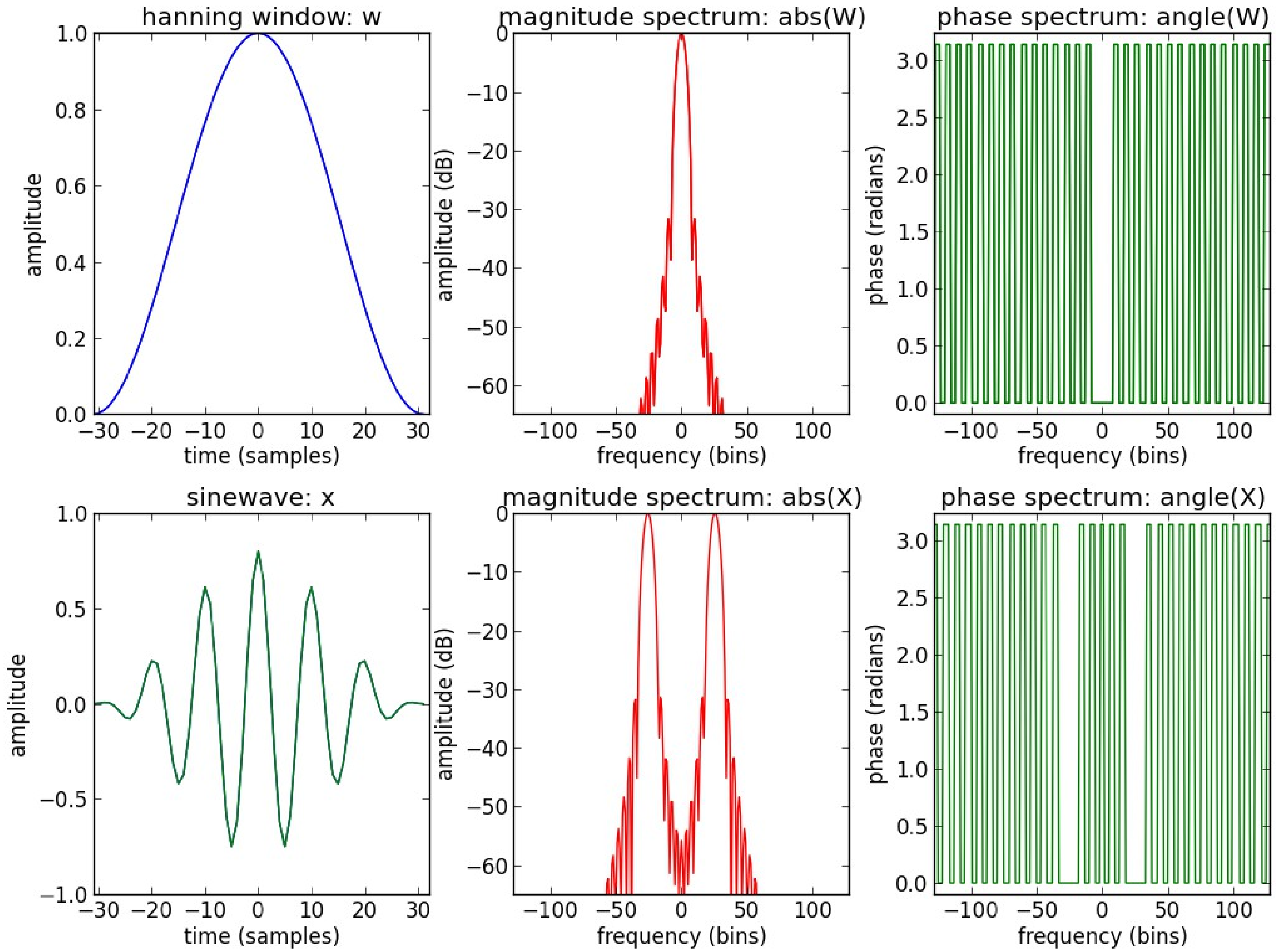



```
F0 = 500.0
Fs = 10000.0
A0 = .8
phi0 = 1.5
Ns = 64
hNs = Ns/2

k = np.int(np.round(Ns*f0/fs))
mY[k] = A0 * Ns
pY[k] = phi0
Y[:hNs] = .5*mY*np.exp(1j*pY)
Y[hNs+1:]=Y[hNs-1:0:-1].conjugate()
plt.subplot(3,1,1)
plt.plot(mY, 'ro')

plt.subplot(3,1,2)
plt.plot(pY, 'bo')

y = np.real(ifft(Y))
yw[:hNs-1] = y[hNs+1:]
yw[hNs-1:] = y[:hNs+1]
plt.subplot(3,1,3)
plt.plot(yw, 'r')
```

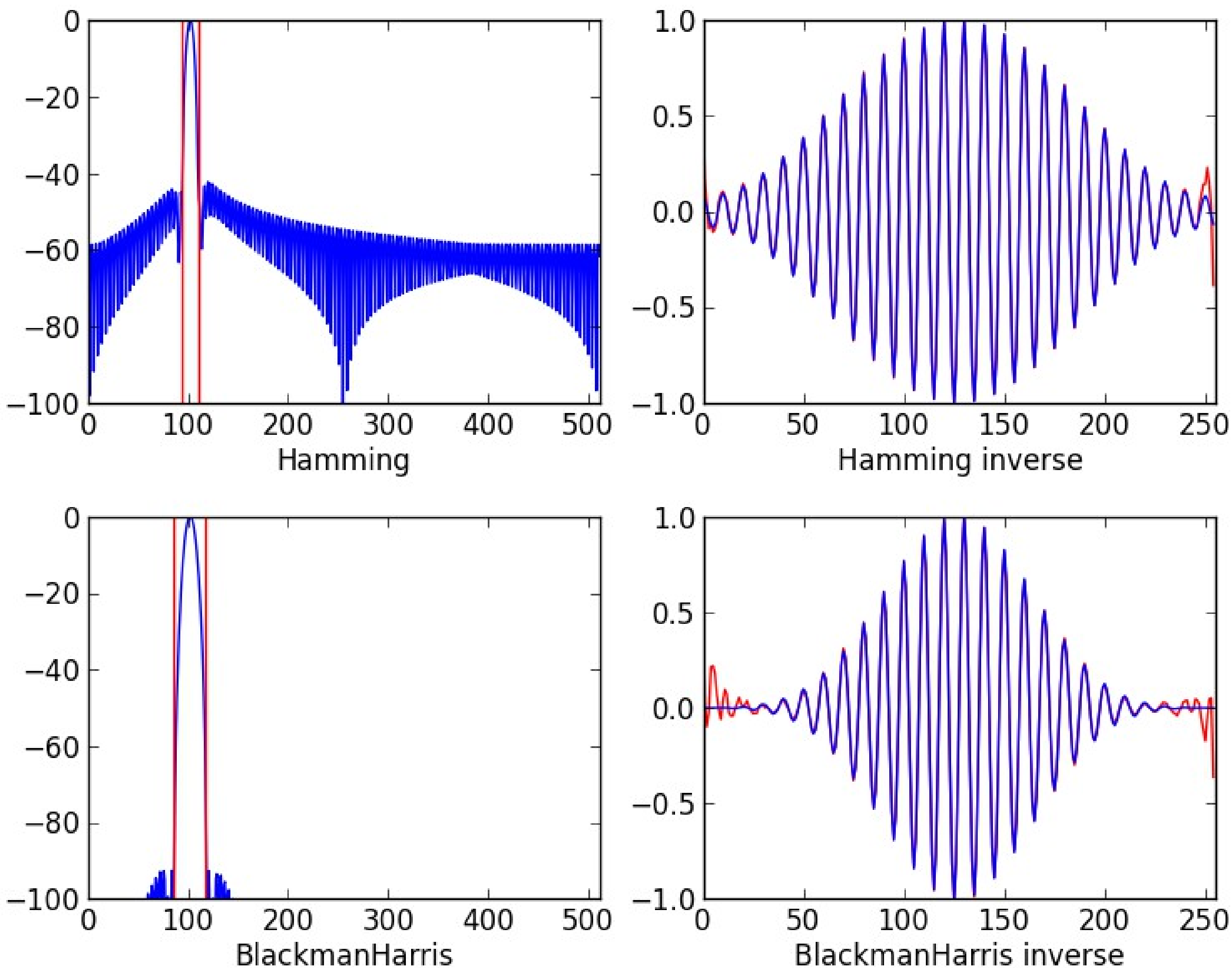
Sinusoids in spectrum



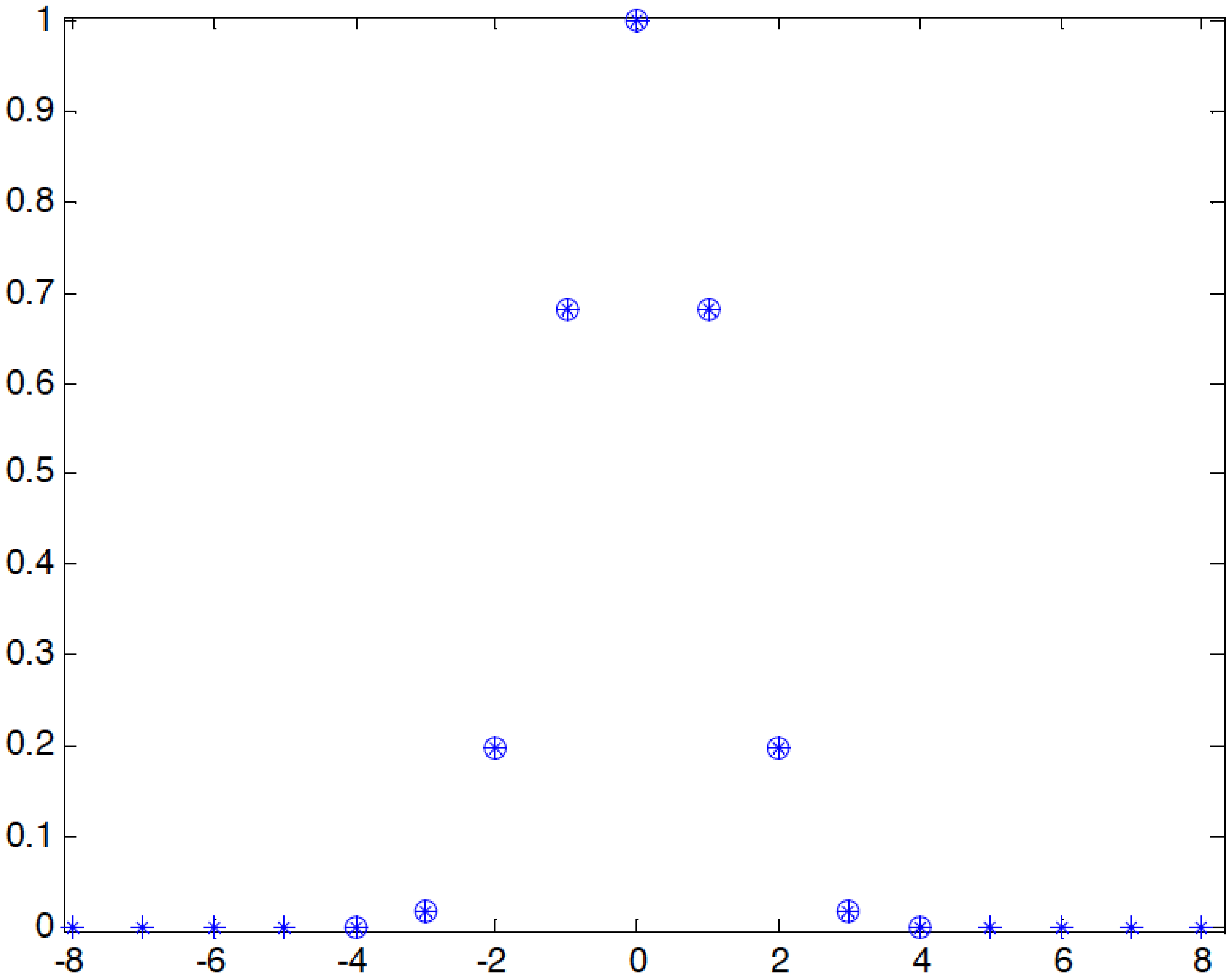
Sinewave spectrum



Inverse of spectral sinusoid

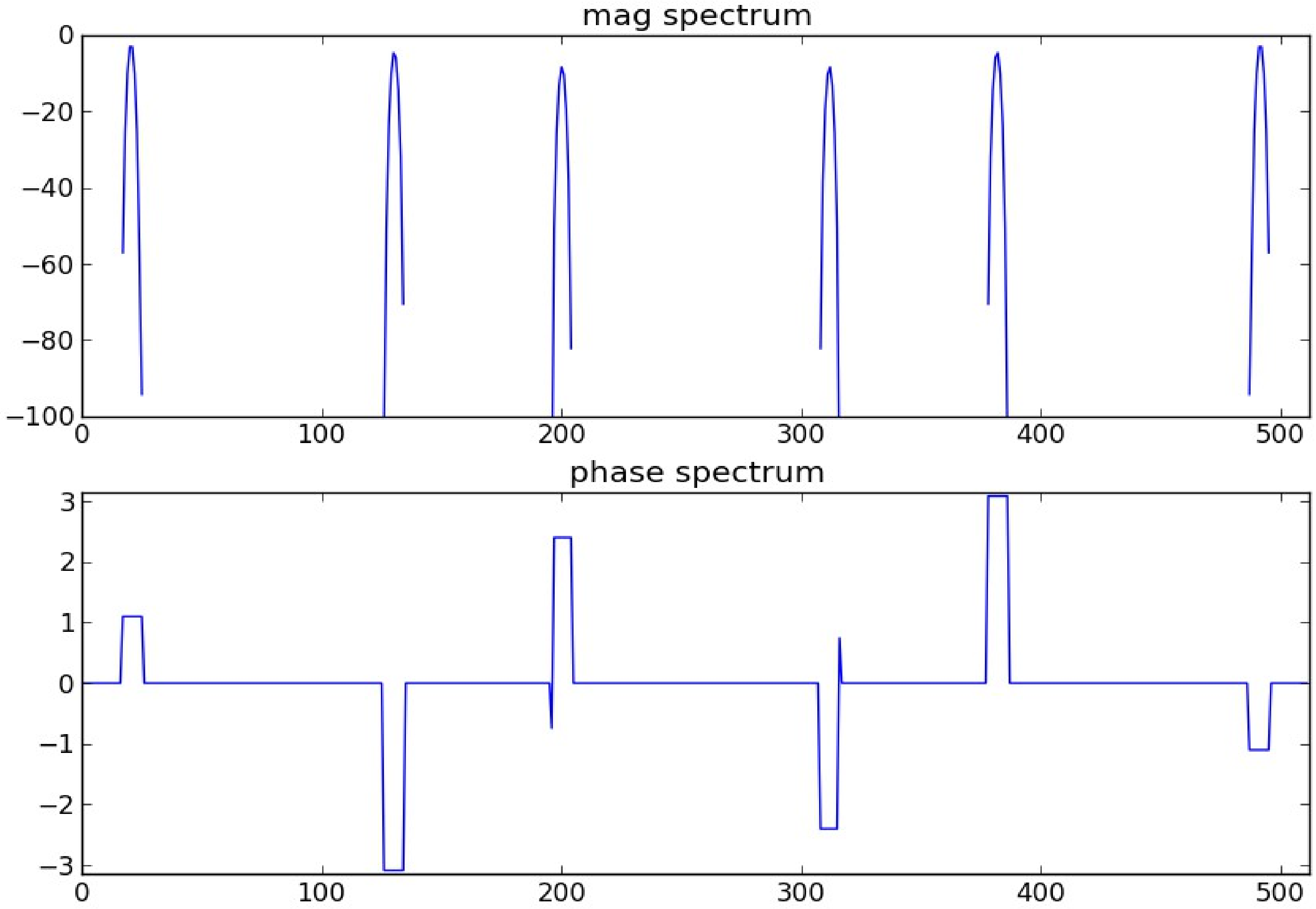


Blackman-Harris main-lobe



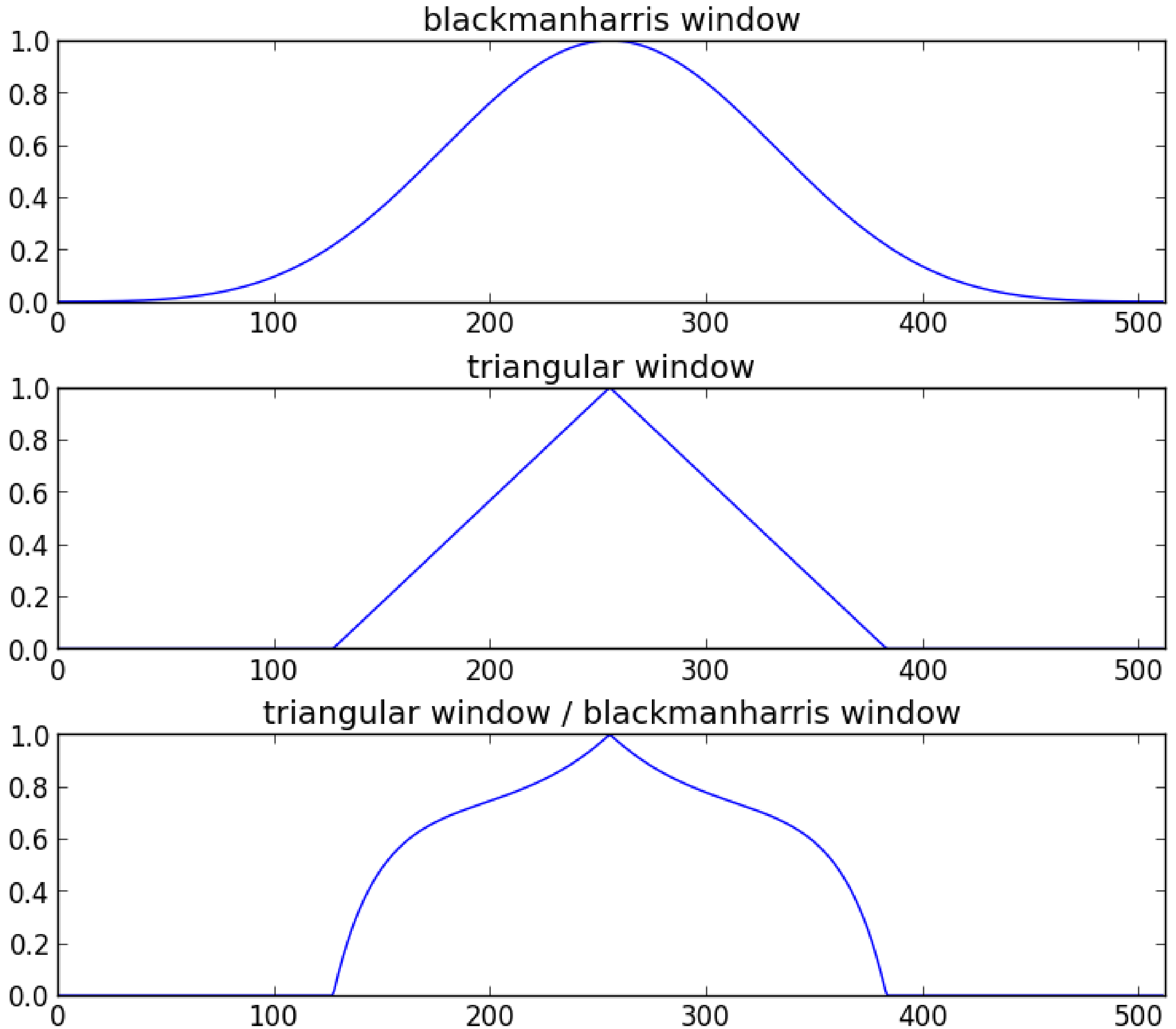
Synthesis spectrum

iploc = 20.5, 130.3, 200.2
ipmag = -2.2, -4.3, -8.2
ipphase = 1.1, 3.2, 2.4

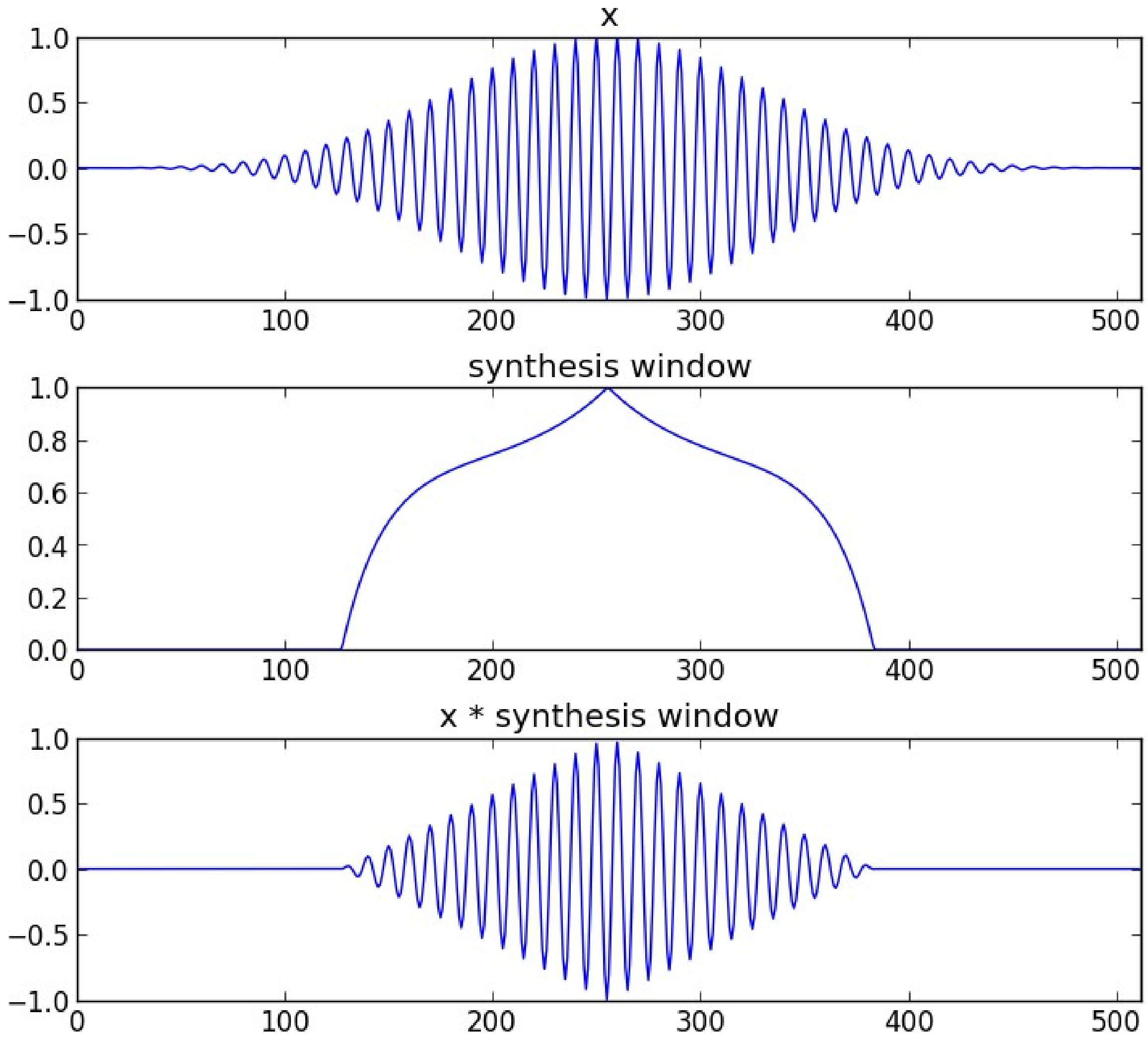


```
def genspecsines(iploc, ipmag, ipphase, N):

  Y = np.zeros(N, dtype = complex)
  hN = N/2

  for i in range(0, iploc.size):
    loc = iploc[i]
    if loc<1 or loc>hN-1: continue
    binremainder = round(loc)-loc;
    lb = np.arange(binremainder-4, binremainder+5)
    lmag = uf.genbh92lobe(lb) * 10**(ipmag[i]/20)
    b = np.arange(round(loc)-4, round(loc)+5)

    for m in range(0, 9):
      if b[m] < 0:
        Y[-b[m]]+=lmag[m]*np.exp(-1j*ipphase[i])
      elif b[m] > hN:
        Y[b[m]]+=lmag[m]*np.exp(-1j*ipphase[i])
      elif b[m]==0 or b[m]==hN:
        Y[b[m]]+=lmag[m]*np.exp(1j*ipphase[i])+lmag[m]*np.exp(-1j*ipphase[i])
      else:
        Y[b[m]]+=lmag[m]*np.exp(1j*ipphase[i])

    Y[hN+1:] = Y[hN-1:0:-1].conjugate()
```

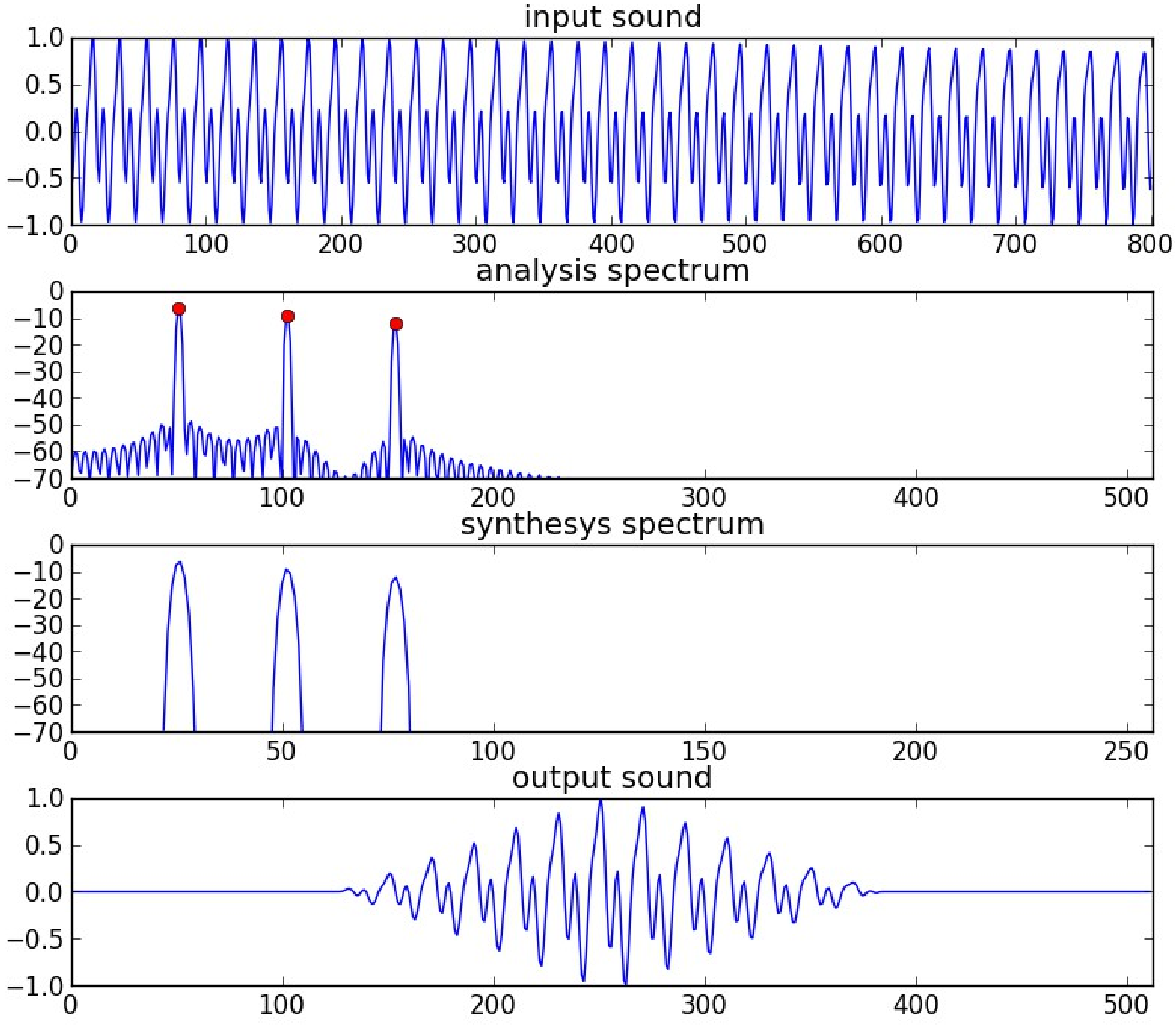
Synthesis window



Synthesis window

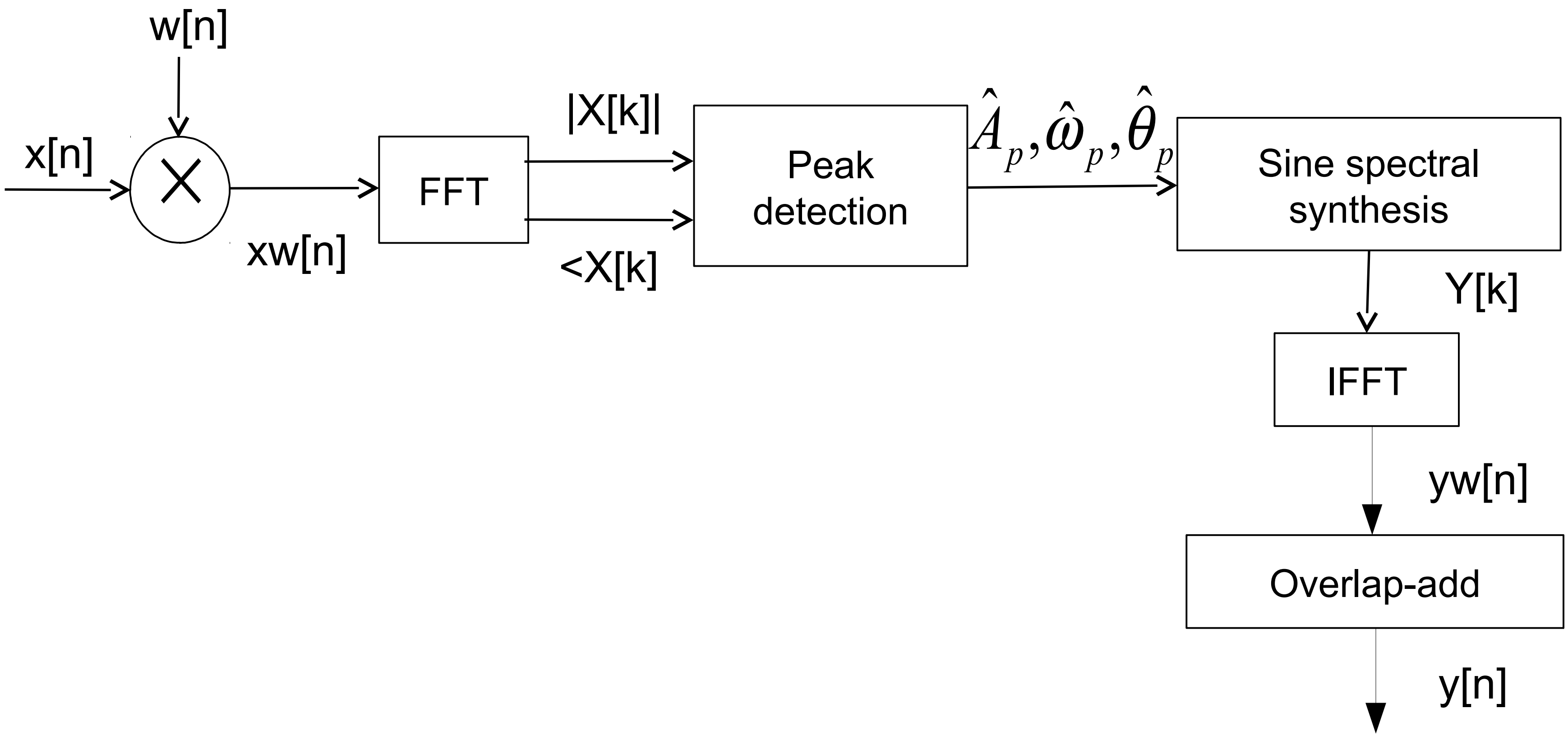


Analysis / Synthesis



M = 801
window = hamming(M)
N = 1024
t = -40
freqs = 100.2, 200.3, 300.2
amps = .99, .7, .5
fs = 2000
Ns = 512

Implementation



```
def sine_model(x, fs, w, N, t):
  hN = N/2
  hM = (w.size+1)/2
  Ns = 512
  H = Ns/4
  hNs = Ns/2
  pin = max(hNs, hM)
  pend = x.size - max(hNs, hM)
  w = w / sum(w)
  ow = triang(2*H);
  sw[hNs-H:hNs+H] = ow
  bh = blackmanharris(Ns)
  bh = bh / sum(bh)
  sw[hNs-H:hNs+H] = sw[hNs-H:hNs+H] / bh[hNs-H:hNs+H]
  while pin<pend:
    xw = x[pin-hM:pin+hM-1] * w
    fftbuffer[:hM] = xw[hM-1:]
    fftbuffer[N-hM+1:] = xw[:hM-1]
    X = fft(fftbuffer)
    mX = 20 * np.log10( abs(X[:hN]) )
    ploc = peak_detection(mX, hN, t)
    pmag = mX[ploc]
    pX = np.unwrap( np.angle(X[:hN]) )
    iploc, ipmag, ipphase = peak_interp(mX, pX, ploc)
    plocs = iploc*Ns/N;
    Y = genspecsines(plocs, ipmag, ipphase, Ns)
    fftbuffer = np.real( ifft(Y) )
    yw[:hNs-1] = fftbuffer[hNs+1:]
    yw[hNs-1:] = fftbuffer[:hNs+1]
    y[pin-hNs:pin+hNs] += sw*yw
    pin += H
  return y
```

References

- https://ccrma.stanford.edu/~jos/sasp/Spectrum_Analysis_Sinusoids.html

Credits