VOX

March 15, 2017

1 Dependencies

In [87]:
```
from __future__ import division

from numpy import *; seterr(all="ignore")
from numpy import linalg
from numpy import random
from matplotlib.pyplot import *
%matplotlib notebook

import wish

from audio.filters import FIR, AR
import audio.frames
import audio.index
import audio.io
from audio.lp import lp
from audio.quantizers import Quantizer
```

In [88]:
```
df=16000
```

2 Sandbox

2.1 FIR

In [89]:
```
fir = FIR([1.0])
print fir(2.0)
print fir(1.0)
print fir([0.0, 7.0, -3.0])
delay = FIR([0.0, 1.0])
fir = delay
print fir(2.0)
print fir(1.0)
print fir([0.0, 7.0, -3.0])
print fir.a
```

```
2.0
1.0
```

```
[ 0.  7. -3.]
0.0
2.0
[ 1.  0.  7.]
[ 0.  1.]
```

3 Audio Sources

3.1 Record Sound

```
In [90]: #data = audio.io.record(3.0, df=df)[0]
         #audio.io.play(data, df=df)
```

3.2 Phones (NLTK)

```
In [91]: aes = audio.index.search("ae", type=audio.index.Phone)
         print aes
         data = aes[0].audio
```

```
    0. ae (had [hv-ae-dcl]).
    1. ae (ask [ae-s]).
    2. ae (rag [r-ae-gcl]).
    3. ae (that [dh-ae-tcl]).
    4. ae (platform [pcl-p-l-ae-tcl-f-ao-m]).
    5. ae (ran [r-ae-nx]).
    6. ae (black [bcl-b-l-ae-kcl]).
    7. ae (man [m-ae-n]).
    8. ae (manufacturer [m-eh-n-y-ix-f-ae-kcl-sh-er-ax]).
    9. ae (clasp [k-l-ae-s-pcl-p]).
   10. ae (hand [hh-ae-n-dcl-d]).
   11. ae (autographs [ao-dx-ix-gcl-g-r-ae-f-s]).
   12. ae (had [hv-ae-dcl-jh]).
   13. ae (ask [ae-s]).
   14. ae (rag [r-ae-gcl-g]).
   15. ae (that [dh-ae-q]).
   16. ae (answered [ae-n-s-ix-dcl-d]).
   17. ae (hand [hh-ae-n]).
   18. ae (glass [gcl-g-l-ae-s]).
   19. ae (muskrat [m-ah-s-kcl-k-r-ae-tcl]).
   20. ae (tadpole [tcl-t-ae-dcl-p-ow-l]).
   21. ae (giraffes [jh-axr-ae-s]).
   22. ae (ask [ae-s-kcl]).
   23. ae (rag [r-ae-gcl-g]).
   24. ae (that [dh-ae-tcl-t]).
   25. ae (example [ix-gcl-z-ae-m-pcl-p-uh-l]).
   26. ae (packing [pcl-p-ae-kcl-k-iy-ng]).
   27. ae (and [q-ae-n-dcl-d]).
```

28. ae (hand [hv-ae-n-dcl-d]).
29. ae (hand [hv-ae-n-dcl-d]).
30. ae (pathological [pcl-p-ae-th-ax-l-aa-dcl-jh-ix-kcl-k-el]).
31. ae (examples [ix-gcl-g-z-ae-m-pcl-p-el-s]).
32. ae (answers [ae-n-s-axr-z]).
33. ae (after [ae-f-tcl-t-axr]).
34. ae (ask [ae-s]).
35. ae (rag [r-ae-gcl]).
36. ae (that [dh-ae-tcl]).
37. ae (pass [pcl-p-ae-s]).
38. ae (relaxed [r-ih-l-ae-kcl-k-s-tcl]).
39. ae (atmosphere [ae-q-m-ax-s-f-ih-r]).
40. ae (scalp [kcl-k-ae-l-pcl-p]).
41. ae (man's [m-ae-n-z]).
42. ae (lamb's [l-ae-m-z]).
43. ae (ask [ae-s-kcl]).
44. ae (rag [r-ae-gcl-g]).
45. ae (that [dh-ae-tcl-t]).
46. ae (universality [dcl-jh-y-ux-nx-ax-v-er-s-ae-l-ax-tcl-t-iy]).
47. ae (masquerade [m-ae-s-kcl-k-axr-r-ey-dcl]).
48. ae (tax [tcl-t-ae-kcl-s]).
49. ae (imagination [ix-m-ae-dcl-jh-ix-n-ey-sh-ix-n]).
50. ae (ask [ae-s-kcl]).
51. ae (rag [r-ae-gcl-g]).
52. ae (that [dh-ae-q]).
53. ae (battery [bcl-b-ae-dx-er-iy]).
54. ae (data [dcl-d-ae-dx-eh]).
55. ae (analysis [n-ae-l-ax-s-ix-s]).
56. ae (have [hh-ae-v]).
57. ae (family [f-ae-m-l-iy]).
58. ae (have [hv-ae-v]).
59. ae (graph [gcl-g-r-ae-f]).
60. ae (axis [ae-kcl-s-ix-s]).
61. ae (ask [ae-s-kcl-k]).
62. ae (rag [r-ae-gcl-g]).
63. ae (that [dh-ae-tcl]).
64. ae (catkins [kcl-k-ae-kcl-k-ix-n-s]).
65. ae (national [n-ae-sh-en-el]).
66. ae (have [hv-ae-v]).
67. ae (challenge [ch-ae-l-ix-n-jh]).
68. ae (ask [ae-s-kcl]).
69. ae (rag [r-ae-gcl-g]).
70. ae (anti [ae-nx-ix]).
71. ae (action [ae-kcl-sh-en]).
72. ae (and [q-ae-n-dcl]).
73. ae (travelers [tcl-t-r-ae-l-axr-z]).
74. ae (now [n-ae]).
75. ae (graph [gcl-g-r-ae-f]).

76. ae (had [hv-ae-dcl]).
77. ae (ask [ae-s]).
78. ae (rag [r-ae-gcl-g]).
79. ae (that [dh-ae-tcl]).
80. ae (staff [tcl-t-ae-f]).
81. ae (planning [pcl-p-l-ae-nx-ix-ng]).
82. ae (had [hv-ae-dcl-jh]).
83. ae (ask [ae-s-kcl]).
84. ae (rag [r-ae-gcl]).
85. ae (that [dh-ae-tcl]).
86. ae (example [ih-gcl-g-z-ae-m-pcl-p-el]).
87. ae (fancy [f-ae-n-tcl-s-iy]).
88. ae (have [hv-ae-v]).
89. ae (valuables [v-ae-y-ix-bcl-b-el-s]).
90. ae (bank [bcl-b-ae-ng-kcl]).
91. ae (had [hv-ae-dcl-jh]).
92. ae (ask [ae-s-kcl]).
93. ae (carry [kcl-k-ae-r-iy]).
94. ae (rag [r-ae-gcl]).
95. ae (that [dh-ae-q]).
96. ae (tad [tcl-t-ae-dcl-d]).
97. ae (and [q-ae-n]).
98. ae (and [q-ae-n]).
99. ae (companions [kcl-k-em-pcl-p-ae-n-y-ix-n-tcl-s]).
100. ae (had [hv-ae-dcl-d]).
101. ae (ask [ae-s-kcl]).
102. ae (carry [kcl-k-ae-r-iy]).
103. ae (rag [r-ae-gcl-g]).
104. ae (that [dh-ae-tcl]).
105. ae (as [q-ae-z]).
106. ae (clarify [kcl-k-l-ae-axr-f-ay]).
107. ae (tantalizing [tcl-t-ae-n-tcl-t-ax-l-ay-z-ix-ng]).
108. ae (muskrat [m-ah-s-kcl-k-r-ae-tcl]).
109. ae (apples [ae-pcl-p-el-z]).
110. ae (tranquilizers [tcl-t-r-ae-ng-kcl-k-w-ax-l-ay-z-ax-h-s]).
111. ae (ask [ae-s-kcl]).
112. ae (rag [r-ae-gcl-g]).
113. ae (that [dh-ae-q]).
114. ae (distracted [dcl-z-ax-h-s-tcl-t-r-ae-kcl-t-ih-dcl-d]).
115. ae (outcast [q-aw-tcl-k-ae-s-tcl-t]).
116. ae (crab [kcl-k-r-ae-bcl]).
117. ae (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
118. ae (stab [s-tcl-t-ae-bcl-b]).
119. ae (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
120. ae (had [hv-ae-dcl]).
121. ae (ask [ae-s-kcl]).
122. ae (rag [r-ae-gcl-g]).
123. ae (that [dh-ae-tcl-t]).

```
  124. ae (tax [tcl-t-ae-kcl-k-s]).
  125. ae (have [hv-ae-v]).
  126. ae (antagonistic [q-ae-n-tcl-t-ae-gcl-ix-n-ih-s-tcl-t-ih-kcl]).
  127. ae (antagonistic [q-ae-n-tcl-t-ae-gcl-ix-n-ih-s-tcl-t-ih-kcl]).
  128. ae (wrap [r-ae-pcl-p]).
  129. ae (standby [s-tcl-t-ae-n-dcl-b-ay]).
  130. ae (practical [pcl-p-r-ae-kcl-t-ax-h-kcl-k-el]).
  131. ae (had [hv-ae-dcl-d]).
  132. ae (ask [ae-s]).
  133. ae (rag [r-ae-gcl-g]).
  134. ae (that [dh-ae-q]).
  135. ae (triumphant [tcl-t-r-ae-ah-m-f-ih-n-q]).
  136. ae (can [k-ae-n]).
  137. ae (gab [gcl-g-ae-bcl]).
  138. ae (habit [hv-ae-bcl-b-ih-tcl]).
  139. ae (had [hv-ae-dcl-jh]).
  140. ae (ask [ae-s]).
  141. ae (rag [r-ae-gcl-g]).
  142. ae (that [dh-ae-tcl]).
  143. ae (camp [kcl-k-ae-m-pcl]).
  144. ae (al [q-ae]).
  145. ae (valley [v-ae-l-ih]).
  146. ae (calf [kcl-k-ae-f]).
  147. ae (mask [m-ae-s]).
```

```
In [92]: you = audio.index.search("you", type=audio.index.Word)[0]
         print you
         ixs = audio.index.search("ix")
         data = ixs[0].audio
         plot(data); axis("tight")
```

```
you [y-ix]
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[92]: (0.0, 710.0, -0.051177978515625, 0.054534912109375)
```

```
In [93]: sentence = audio.index.search(type=audio.index.Utterance)[0]
         for word in sentence:
             print word
             for phone in word:
                 print 4*" ", phone
```

```
        data = sentence.audio
        print data
        audio.io.play(data, df=16000.0)

h# (a crab challenged me but a quick stab vanquished him).
a [q-ey]
     q (a [q-ey]).
     ey (a [q-ey]).
crab [kcl-k-r-ae-bcl]
     kcl (crab [kcl-k-r-ae-bcl]).
     k (crab [kcl-k-r-ae-bcl]).
     r (crab [kcl-k-r-ae-bcl]).
     ae (crab [kcl-k-r-ae-bcl]).
     bcl (crab [kcl-k-r-ae-bcl]).
challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]
     ch (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     ae (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     l (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     ax (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     n (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     dcl (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     jh (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     dcl (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
     d (challenged [ch-ae-l-ax-n-dcl-jh-dcl-d]).
epi (a crab challenged me but a quick stab vanquished him).
me [m-iy]
     m (me [m-iy]).
     iy (me [m-iy]).
pau (a crab challenged me but a quick stab vanquished him).
but [b-eh-dx]
     b (but [b-eh-dx]).
     eh (but [b-eh-dx]).
     dx (but [b-eh-dx]).
a [ix]
     ix (a [ix]).
quick [kcl-k-w-ih-kcl-k]
     kcl (quick [kcl-k-w-ih-kcl-k]).
     k (quick [kcl-k-w-ih-kcl-k]).
     w (quick [kcl-k-w-ih-kcl-k]).
     ih (quick [kcl-k-w-ih-kcl-k]).
     kcl (quick [kcl-k-w-ih-kcl-k]).
     k (quick [kcl-k-w-ih-kcl-k]).
stab [s-tcl-t-ae-bcl-b]
     s (stab [s-tcl-t-ae-bcl-b]).
     tcl (stab [s-tcl-t-ae-bcl-b]).
     t (stab [s-tcl-t-ae-bcl-b]).
     ae (stab [s-tcl-t-ae-bcl-b]).
     bcl (stab [s-tcl-t-ae-bcl-b]).
```

```
     b (stab [s-tcl-t-ae-bcl-b]).
vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]
     v (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     ae (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     ng (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     kcl (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     k (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     w (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     ix (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     sh (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     tcl (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
     t (vanquished [v-ae-ng-kcl-k-w-ix-sh-tcl-t]).
him [hh-ih-m]
     hh (him [hh-ih-m]).
     ih (him [hh-ih-m]).
     m (him [hh-ih-m]).
h# (a crab challenged me but a quick stab vanquished him).
[  9.15527344e-05   9.15527344e-05   6.10351562e-05 ...,   3.05175781e-05
   0.00000000e+00   6.10351562e-05]
```

3.3 Sentence (NLTK)

```
In [94]: utterances = audio.index.search(type=audio.index.Utterance)
         n = 10
         data = utterances[n].audio
         audio.io.play(data, df=16000.0)
```

4 Short-Term Prediction

```
In [95]: class STP(Quantizer):
             "Short-Term Predictor"
             def __init__(self, order=16, method="autocorrelation"):
                 self.fir = FIR(a=r_[1.0, zeros(order)])
                 self.ar  = AR(a=zeros(order))
                 self.order = order
                 self.method = method
             def encode(self, data):
                 if self.method == "covariance" and self.order >= len(data):
                     raise ValueError("not enough data samples")
                 a = lp(data, order=self.order, method=self.method)
                 self.fir.a[:] = r_[1.0, -a]
                 error = self.fir(data)
                 return (a, error)
             def decode(self, data):
                 a, error = data
                 self.ar.a[:] = a
```

```
            return self.ar(error)
```

```
In [96]: def stp_error(data, T=0.02, order=16, method="autocorrelation"):
             length = len(data)
             n = int(T * df) # number of samples for T s at the given frequency.
             frames = audio.frames.split(data, n, pad=True)
             stp = STP(order=order, method=method)
             error = zeros(n*len(frames))
             for i, frame in enumerate(frames):
                 a, error_frame = stp.encode(frame)
                 error[i*n:(i+1)*n] = error_frame
             return error[:length]
```

```
In [97]: figure()
         n = len(data)
         t = r_[0:n] / df
         plot(t,data, "k", alpha=0.1); axis("tight")
         error = stp_error(data)
         plot(t,error, "r")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[97]: [<matplotlib.lines.Line2D at 0x7f02fe70a610>]
```

```
In [98]: error = stp_error(data)
         SNR2 = mean(data*data)/mean(error*error)
         print "SNR", 10*log10(SNR2), "dB"
         audio.io.play(data, df=df)
         audio.io.play(error, df=df)
         M = amax(abs(error))
         audio.io.play(error/M, df=df)
```

```
SNR 18.3333924274 dB
```

5 Pitch Estimation

```
In [99]: def ltp_parameters(history, frame,
                            offset_min=1, offset_max=None,
                            gain_min=0.0, gain_max=inf,
                            SNR_min=1.0, SNR_max=inf,
                            returns="offset, gain"):
             p = len(history)
             data = r_[history, frame] # full data
```

```
    m = len(frame)
    n = len(data)
    nxcorrs = zeros(p+1)
    gains = zeros(p+1)
    SNRs = zeros(p+1)
    valids = zeros(p+1, dtype=bool)
    frame_norm = linalg.norm(frame)
    normed_frame = frame / frame_norm
    for i in range(p + 1):
        windowed_data = data[n-i-m:n-i]
        windowed_data_norm = linalg.norm(windowed_data)
        normed_windowed_data = windowed_data / windowed_data_norm
        nxcorr = nxcorrs[i] = dot(normed_frame, normed_windowed_data)
        SNR = SNRs[i] = 1.0 / sqrt(1 - nxcorr*nxcorr)
        #print ">", SNR
        gain = gains[i] = nxcorr / windowed_data_norm * frame_norm
        valid = True
        if offset_min is not None:
            valid = valid and (offset_min <= i)
        if offset_max is not None:
            valid = valid and (i <= offset_max)
        valid = valid and (gain_min <= gain <= gain_max)
        valid = valid and (SNR_min <= SNR <= SNR_max)
        valids[i] = valid

    criteria = SNRs.copy()
    criteria[logical_not(valids)] = -inf
    offset = argmax(criteria)
    if not valids[offset]: # everything is invalid!
        raise ValueError("no valid set of parameters")
    else:
        gain = gains[offset]
        nxcorr = nxcorrs[offset]
        SNR = SNRs[offset]

    return wish.grant(returns)
```

```
In [100]: N = 100
          data_ = (0.7 * sin(r_[0:N]/N * 2*pi*4) + 0.10 * random.uniform(-1,1,N)) * (1.0 + 2.0*r
          #history[::7] = 1.0
          history, frame = data_[:-25], data_[-25:]
          m = len(history)
          n = len(frame)
          offset_min = 5
          offset, gain, nxcorrs, SNRs, valids = ltp_parameters(history, frame, offset_min=offset
                                                               returns="offset, gain, nxcorrs, S
          print "offset:", offset, "gain:", gain
```

```
figure()
plot(data_, "k", alpha=0.5, label="data")
plot(arange(0,n)+m, frame, "b", label="reference")
plot(arange(m-offset, m-offset+n), frame/gain, "r", label="matched")
axis("tight")
legend(loc=0)
```

```
offset: 50 gain: 1.48490834685
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[100]: <matplotlib.legend.Legend at 0x7f02fe59b7d0>
```

```
In [101]: figure()
          m = arange(len(SNRs))
          plot(m[offset_min:],SNRs[offset_min:], "r", alpha=0.25, label="SNR")
          n = arange(len(SNRs))
          plot(n[valids],SNRs[valids], "bx", linewidth=1.0,label="valid")
          xlabel("offset")
          ylabel("SNR (linear scale)")
          axis("tight")
          legend(loc=0)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[101]: <matplotlib.legend.Legend at 0x7f02fe54b8d0>
```

6 Long-Term Prediction

```
In [102]: class LTP(Quantizer):
              def __init__(self, order, **options):
                  self.fir = FIR(a=r_[1.0, zeros(order)])
                  self.history = zeros(order)
                  self.ar = AR(a=zeros(order))
                  self.order = order
                  self.options = options
              def encode(self, frame):
                  a = zeros_like(self.fir.a)
                  a[0] = 1.0
                  try:
```

```
                    offset, gain = ltp_parameters(self.history, frame, **self.options)
                    a[offset] = - gain
                except ValueError:
                    offset, gain = 0, 0.0
                self.fir.a[:] = a
                error = self.fir(frame)
                self.history = r_[self.history[len(frame):], frame]
                return (offset, gain), error
            def decode(self, data):
                (offset, gain), error = data
                a = zeros_like(self.ar.a)
                a[offset-1] = gain
                self.ar.a[:] = a
                return self.ar(error)
```

In [103]:
```
f_min = 50.0
f_max = 400.0
order_ltp = int(df/f_min)
print order_ltp
offset_min = int(df/f_max)
print offset_min
```

```
320
40
```

In [104]:
```
def ltp_error(data, T=0.005, order=order_ltp, **options):
    length = len(data)
    n = int(T * df) # number of samples for T s at the given sampling frequency.
    frames = audio.frames.split(data, n, pad=True)
    ltp = LTP(order=order, **options)
    error = zeros(n*len(frames))
    offset = zeros_like(error)
    gain = zeros_like(error)
    for i, frame in enumerate(frames):
        (offset_, gain_), error_frame = ltp.encode(frame)
        error[i*n:(i+1)*n] = error_frame
        offset[i*n:(i+1)*n] = ones_like(error_frame) * offset_
        gain[i*n:(i+1)*n] = ones_like(error_frame) * gain_
    error = error[:length]
    offset = offset[:length]
    gain = gain[:length]
    return error, offset, gain
```

In [105]:
```
stp_error_ = stp_error(data)
ltp_error_, offset, gain = ltp_error(stp_error_, offset_min=offset_min, SNR_min=1.1)

figure()
n = len(data)
```

```
t = r_[0:n] / df

plot(t,data, "k", alpha=0.1, label="audio"); axis("tight")
plot(t, stp_error_, "r", label="STP error")
plot(t, ltp_error_, "g", label="LTP error")
legend(loc=0)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[105]: <matplotlib.legend.Legend at 0x7f02fe453b50>
```

```
In [106]: figure()
          n = len(data)
          t = r_[0:n] / df
          plot(t, df / offset)
          plot(t, 2*df / offset, "k.", alpha=0.5, ms=0.25)
          plot(t, 3*df / offset, "k.", alpha=0.25, ms=0.25)

          axis([t[0], t[-1], 00.0, 400.0])
          ylabel("Frequency (Hz)")
          xlabel("Time (s)")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[106]: <matplotlib.text.Text at 0x7f02fe484d10>
```

```
In [107]: figure()
          n = len(data)
          t = r_[0:n] / df
          plot(t, gain)
          axis("tight")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[107]: (0.0, 2.5599375000000002, 0.0, 4.4183025880819482)
```

```
In [108]: audio.io.play(data, df=df)
          A = amax(abs(stp_error_))
          audio.io.play(stp_error_/A, df=df)
          A = amax(abs(ltp_error_))
          audio.io.play(ltp_error_/A, df=df)
```