In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
#%matplotlib
```

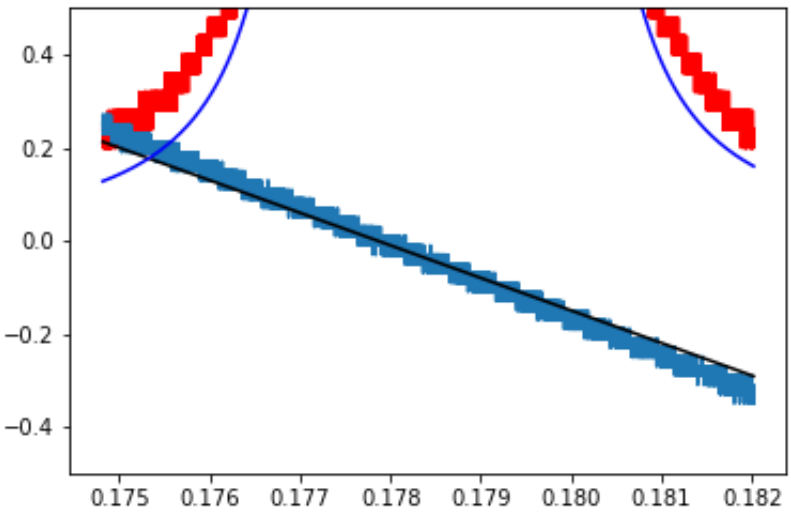
Fit of IRMC transmission

1. oringial data and roughly manual fit

1.1 TEM00 and sidebands zoom in

In [3]:
```
data1 = np.loadtxt(open("/Users/ihong/desktop/work201809/20181010t_
f_cali/T0054ALL.csv",'rb'),delimiter=',',skiprows=16)
t = data1[30000:120000, 0]
d = data1[30000:120000, 1]
r = data1[30000:120000, 3]

A = 5.6
A1 = 1.25
F = 1271
t0 = 0.0286
t1 = 0.0013
FSR = 0.05

k = -70
b = 12.45

y = A/(1+F*np.sin(np.pi*(t-t0)/FSR)**2)+A1/(1+F*np.sin(np.pi*(t-(t0
-t1))/FSR)**2)+A1/(1+F*np.sin(np.pi*(t+(-t1-t0))/FSR)**2)
yh = k*t+b
plt.plot(t, d, 'r')
plt.plot(t, r)
plt.plot(t, yh, 'k')
plt.plot(t, y, 'b')
plt.ylim(-0.5,0.5)
#print(np.shape(t))
```

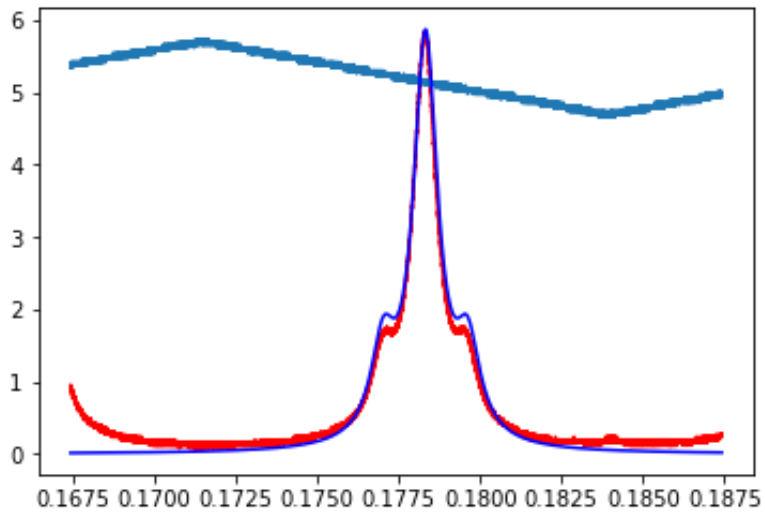
Out[3]: (-0.5, 0.5)



1.2 TEM00 and sideband with the whole ramp

In [4]:
```
data2 = np.loadtxt(open("/Users/ihong/desktop/work201809/20181010t_
f_cali/T0055ALL.csv",'rb'),delimiter=',',skiprows=16)
t2 = data2[:, 0]
d2 = data2[:, 1]
r2 = data2[:, 3]

A = 5.6
A1 = 1.25
F = 1222
t0 = 0.02832
t1 = 0.0013
FSR = 0.05

y2 = A/(1+F*np.sin(np.pi*(t2-t0)/FSR)**2)+A1/(1+F*np.sin(np.pi*(t2-
(t0-t1))/FSR)**2)+A1/(1+F*np.sin(np.pi*(t2+(-t1-t0))/FSR)**2)
plt.plot(t2, d2, 'r')
plt.plot(t2, r2)
plt.plot(t2, y2, 'b')
#print(np.shape(t))
```

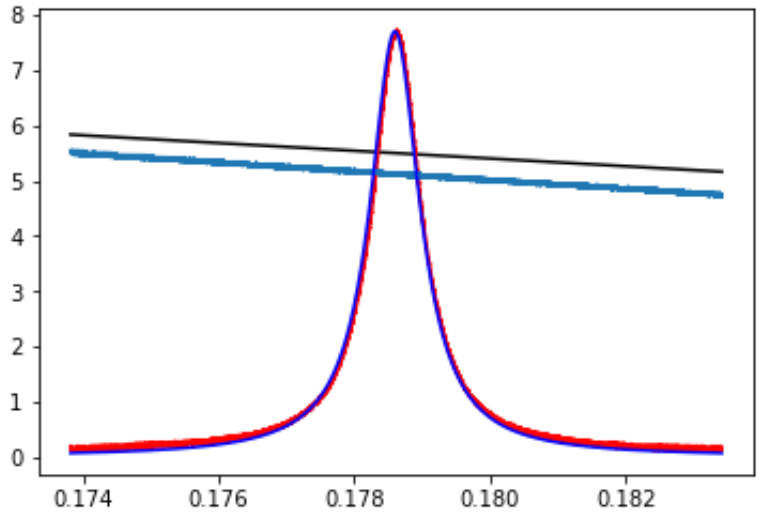
Out[4]: [<matplotlib.lines.Line2D at 0x1015763b00>]



1.3 only TEM00

In [5]:
```
data3 = np.loadtxt(open("/Users/ihong/desktop/work201809/20181010t_
f_cali/T0056ALL.csv",'rb'),delimiter=',',skiprows=16)
t3 = data3[40000:100000, 0]
d3 = data3[40000:100000, 1]
r3 = data3[40000:100000, 3]

A3 = 7.7
F3 = 1271
t03 = -0.0214
FSR3 = 0.05

k3 = -70
b3 = 18
yh3 = k3*t3+b3
plt.plot(t3, yh3, 'k')
y3 = A3/(1+F3*np.sin(np.pi*(t3-t03)/FSR3)**2)
plt.plot(t3, d3, 'r')

plt.plot(t3, r3)
plt.plot(t3, y3, 'b')
#print(np.shape(t))
```

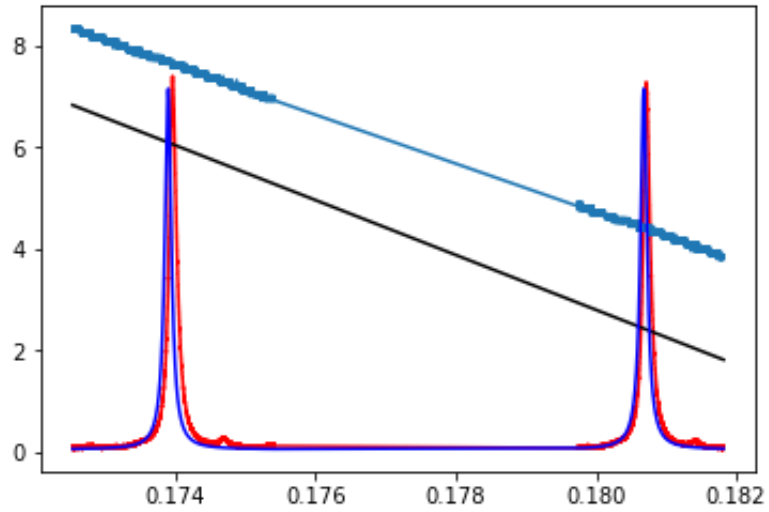
Out[5]:
```
[<matplotlib.lines.Line2D at 0x101a1f1b38>]
```



1.4 two TEM00

In [6]:
```
data4 = np.loadtxt(open("/Users/ihong/desktop/work201809/20181010t_
f_cali/T0057ALL.csv",'rb'),delimiter=',',skiprows=16)
t4 = np.hstack((data4[32000:50000, 0], data4[77000:90000, 0]))
d4 = np.hstack((data4[32000:50000, 1], data4[77000:90000, 1]))
r4 = np.hstack((data4[32000:50000, 3], data4[77000:90000, 3]))

A4 = 7.1
F4 = 1500
t04 = 0.0044
FSR4 = 0.00678

k4 = -540
b4 = 100

y4 = A4/(1+F4*np.sin(np.pi*(t4-t04)/FSR4)**2)+0.05
yh4 = k4*t4+b4
plt.plot(t4, d4, 'r')
plt.plot(t4, yh4, 'k')
plt.plot(t4, r4)
plt.plot(t4, y4, 'b')
print(np.shape(t4))
```

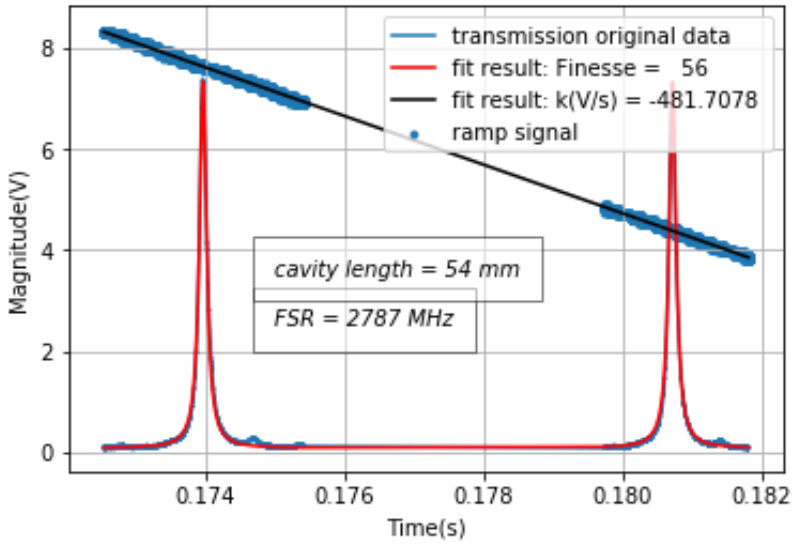
```
(31000,)
```



2. fit by program to get Finesse value

In [7]:
```
def airy(x, a, f, x0, fsr, o):
    return a/(1+f*np.sin(np.pi*(x-x0)/fsr)**2)+o
```

In [14]:
```
popt4, pcov4 = curve_fit(airy, t4, d4, bounds=([7.1, 800, 0.00442,
0.006759, 0.01], [7.3, 2500, 0.00499, 0.006781, 0.09]))

plt.plot(t4, d4, label='transmission original data')
plt.scatter(t4, r4, marker=".", label='ramp signal')

popth4, pcovh4 = curve_fit(hvd, t4, r4, bounds=([-600, 80], [-300,
150]))

Fi = np.pi*np.sqrt(popt4[1])/2

plt.plot(t4, airy(t4, *popt4), 'r',   label='fit result: Finesse =
%4.0f '% Fi)

FSR1 = -1*popt4[3]*popth4[0]*cal
fsr1 = ("%.0f" % FSR1)

L = 3*10**11/2/FSR1/1000000
l = ("%.0f" % L)

plt.plot(t4, hvd(t4, *popth4),  'k',   label='fit result: k(V/s) =
%6.4f '% popth4[0])

plt.text(0.175, 2.5, 'FSR = {} MHz'.format(fsr1), style='italic',
        bbox={'facecolor':'white', 'alpha':0.5, 'pad':10})
plt.text(0.175, 3.5, 'cavity length = {} mm'.format(l), style='ital
ic',
        bbox={'facecolor':'white', 'alpha':0.5, 'pad':10})

plt.xlabel('Time(s)')
plt.ylabel('Magnitude(V)')
#plt.title('Fit of mode cleaner output beam')
#plt.xlim([0.0121,0.0192])
#plt.ylim([0.0003,0.0006])
plt.legend()
plt.grid()
plt.show()

# residual sum of squares
ss_res4 = np.sum((t - airy(t, *popt4)) ** 2)

# total sum of squares
ss_tot4 = np.sum((t - np.mean(airy(t, *popt4))) ** 2)

# r-squared
r24 = 1 - (ss_res4 / ss_tot4)

print(popt4)
```




```
[7.26372293e+00 1.27170642e+03 4.99000000e-03 6.75901024e-03
 7.43131509e-02]
```

In [289]:
```
Fsim = np.pi*np.sqrt(np.sqrt(0.92*0.99975))/(1-np.sqrt(0.99975*0.92
))
print(Fsim)
```

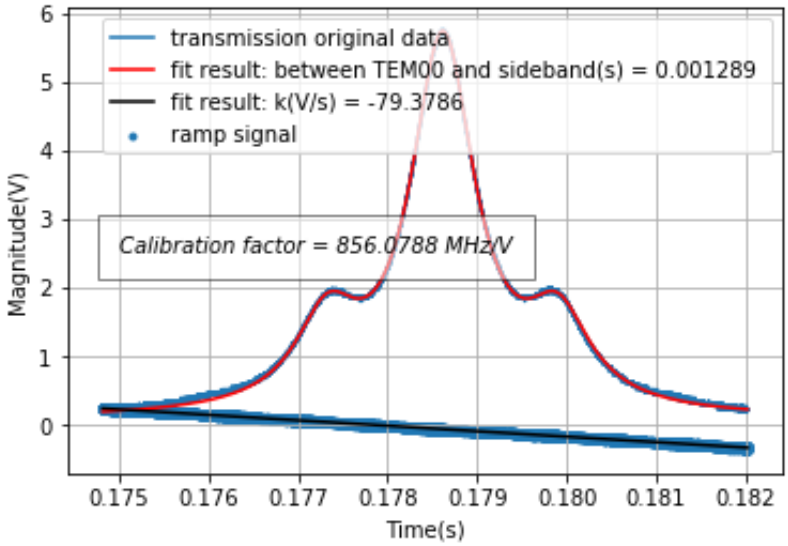
```
75.12381131420992
```

3. fit by program to get calibration factor

In [9]:
```
def airy2(x, a, a1, f, x0, x1, fsr, o):
    return a/(1+f*np.sin(np.pi*(x-x0)/fsr)**2)+a1/(1+f*np.sin(np.pi
*(x-x0+x1)/fsr)**2)+a1/(1+f*np.sin(np.pi*(x-x0-x1)/fsr)**2)+o
```

In [10]:
```
def hvd(x, k, b):
    return k*x+b
```

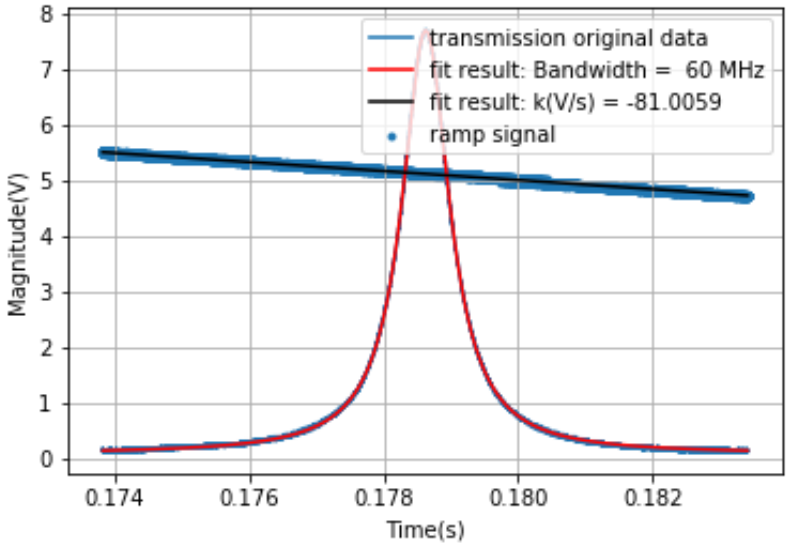
In [12]:
```
popt1, pcov1 = curve_fit(airy2, t, d, bounds=([5.4, 1.15, 1171, 0.0
284, 0.0012, 0.03, 0.01], [5.8, 1.35, 1371, 0.0288, 0.0014, 0.07, 0
.08]))

popth, pcovh = curve_fit(hvd, t, r, bounds=([-90, 12], [-60, 16]))

plt.plot(t, d, label='transmission original data')
plt.scatter(t, r, marker=".", label='ramp signal')

#Fi = np.pi*np.sqrt(popt1[2])/2

cal = 87.6/(-1*popth[0]*popt1[4])

c=("%.4f" % cal)

plt.plot(t, airy2(t, *popt1),  'r',   label='fit result: between TE
M00 and sideband(s) = %6.6f '% popt1[4] )

plt.plot(t, hvd(t, *popth),  'k',   label='fit result: k(V/s) = %6.
4f '% popth[0])

plt.text(0.175, 2.5, 'Calibration factor = {} MHz/V'.format(c), sty
le='italic',
        bbox={'facecolor':'white', 'alpha':0.5, 'pad':10})

plt.xlabel('Time(s)')
plt.ylabel('Magnitude(V)')
#plt.title('Fit of mode cleaner output beam')
#plt.xlim([0.0121,0.0192])
#plt.ylim([0.0003,0.0006])
plt.legend()
plt.grid()
plt.show()

# residual sum of squares
ss_res4 = np.sum((t - airy2(t, *popt1)) ** 2)

# total sum of squares
ss_tot4 = np.sum((t - np.mean(airy2(t, *popt1))) ** 2)

# r-squared
r24 = 1 - (ss_res4 / ss_tot4)

print(popt1)
print(popth)
```




```
[5.40000010e+00 1.25574144e+00 1.33844358e+03 2.86037532e-02
 1.28910158e-03 5.00038776e-02 8.00000000e-02]
[-79.37855928  14.12091703]
```

4. fit by program to get bandwidth

In [15]:
```
popt3, pcov3 = curve_fit(airy, t3, d3, bounds=([7.6, 900, -0.0217,
0.049, 0.01], [7.8, 1600, -0.00211, 0.051, 0.09]))

popth3, pcovh3 = curve_fit(hvd, t3, r3, bounds=([-90, 15], [-60, 20
]))

plt.plot(t3, d3, label='transmission original data')
plt.scatter(t3, r3, marker=".", label='ramp signal')

Fi = -1*np.pi*np.sqrt(popt3[1])/2

BW = popt3[3]*popth3[0]*cal/Fi

bw = ("%.0f" % BW)

plt.plot(t3, airy(t3, *popt3), 'r',   label='fit result: Bandwidth
=  {} MHz'.format(bw))
plt.plot(t3, hvd(t3, *popth3),  'k',   label='fit result: k(V/s) =
%6.4f '% popth3[0])

plt.xlabel('Time(s)')
plt.ylabel('Magnitude(V)')
#plt.title('Fit of mode cleaner output beam')
#plt.xlim([0.0121,0.0192])
#plt.ylim([0.0003,0.0006])
plt.legend()
plt.grid()
plt.show()

# residual sum of squares
ss_res4 = np.sum((t - airy(t, *popt4)) ** 2)

# total sum of squares
ss_tot4 = np.sum((t - np.mean(airy(t, *popt4))) ** 2)

# r-squared
r24 = 1 - (ss_res4 / ss_tot4)

print(popt3)
```




```
[ 7.61106185e+00  1.31392112e+03 -1.82134831e-02  4.92100847e-02
  8.52449162e-02]
```