Examen #1

INEL 5607

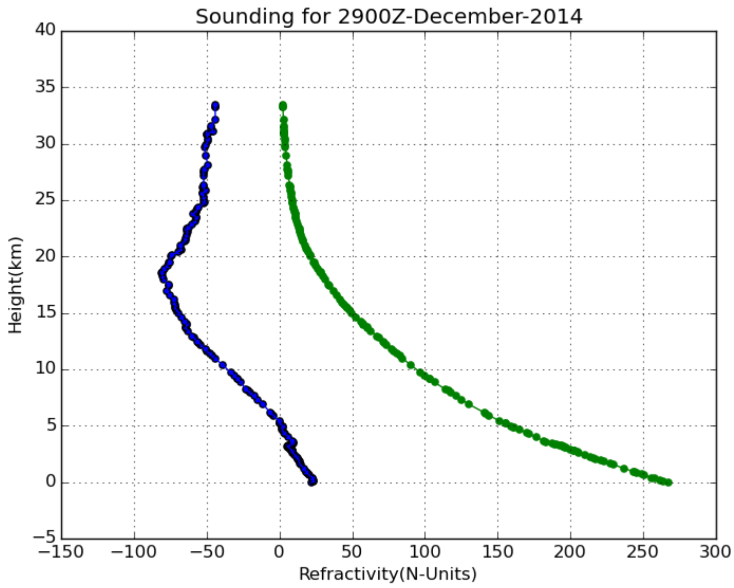
Sebastiani Aguirre-Navarro

March 8, 2015

Problems

1. **a.**

   With or without water vapor, Refractivity decreases as Height continues to increase. As shown in the following figure:



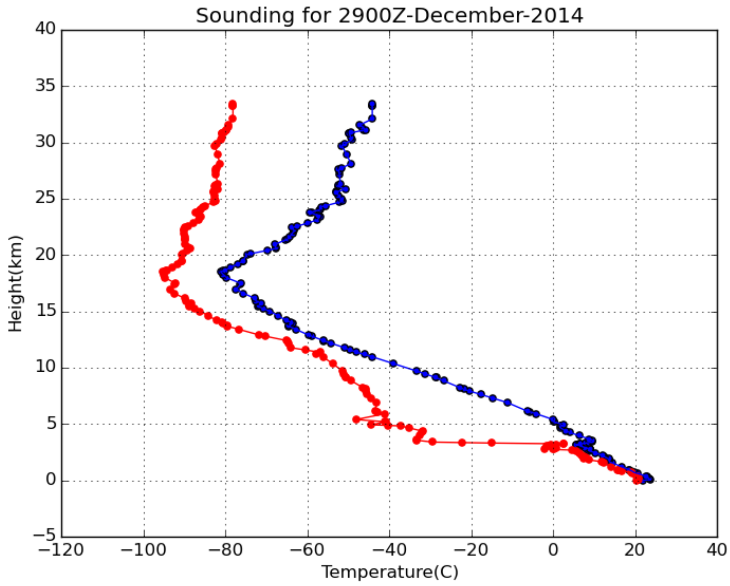
   In the figure, the blue curve is the Refractivity with water vapor pressure taken into account. The green curve is the refractivity without water vapor.

   b.

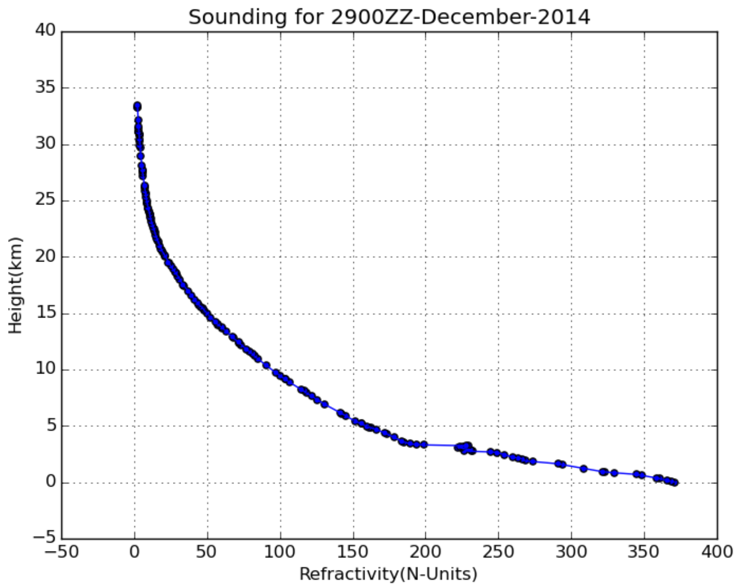
   There is no exagerated departure from the exponential model, as the curve formed by the data decreases in a logarithmic fashion.

   c.

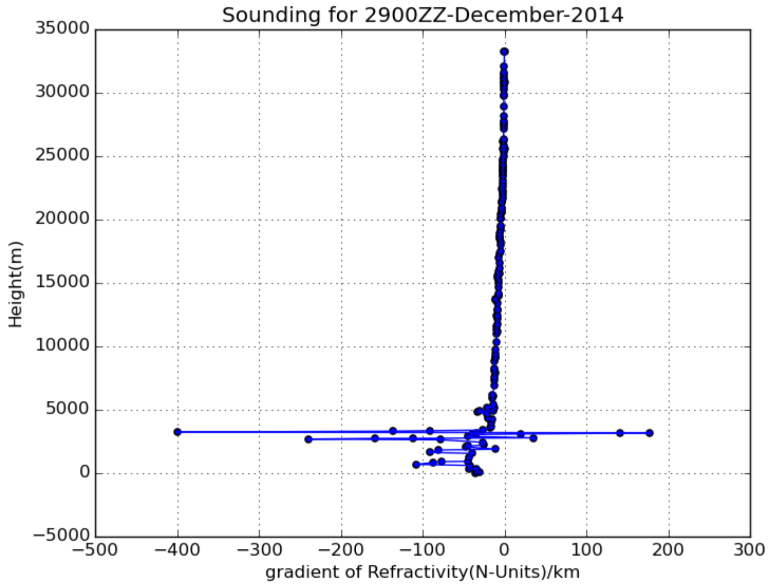
   Using atmospheric sounding data from the 29th of December last year and the python code presented in the appendix.



In this plot, temperature(Blue curve) and dew-point temperature(red curve) are presented along with how their respective height value. We can observe an inverse relationship between the two quantities for each curve. This makes sense, since the higher you go up the colder it is supposed to get.



In this plot, values of refractivity are presented for different values of Height. Have in mind that in this relationship, the number of free electrons was not included in the calculation of refractivity. This is due to the fact that the data was acquired in the troposphere. Analysis of this graph tells us that refractivity also decreases as height increases. This also makes sense, since as we go up the layers of the atmosphere, we get nearer and nearer to vacuum space, meaning that the difference between refractive indexes of the atmosphere and the free space will be close to zero, and hence reduction in refractivity.



This plot is proof of what is happening in the plot above. The gradients have strong tendency to be negative, which is to be expected since refractivity is decreasing throught the trayectory. Also, another point proven here is that the gradients(which are nothing more than differences between each refractivity value at a particular height) approach to zero as height increases. This tells us that the value of the refractive index is indeed approaching that of the free space as we go up the layers.

2. The following sources where used to find this table:

- `http://www.rli-radar.com/rli/hardware/rxm25/overview.html`
- `http://www.rli-radar.com/rli/hardware/rxm25/trx.html`

Antenna	
Field	Value
Shape	Front Fed Parabolic
Diameter	1.8m
Feed Type	Orthogonal Dual Polarization
Gain	41dB
Beamwidth	1.4°
Max Slew Rate	60°/s
Transmitters	
Wavelength	3.18cm
Peak Power	8kW
Final PA Type	Magnetron
Polarization	Dual Linear, H and V
Waveform	single or multiple PRF
Receivers/Signal Processing	
Noise Figure	$\leq$ 5dB
Dynamic Range	105dB (1MHz)
Bandwidth	125MHz
Max. Range Gates	16384

Table 1: TropiNet(RXM-25) Hardware Description

3. Using the following equation for radar constant:

$$C = \frac{\pi^3 c P_t G^2 \tau \theta^2 L_m L_r}{1024 ln 2 \lambda^2}$$

a. With

- $P_t = 10^6 W$
- $G = 43dB$
- $\lambda = 0.1101m$
- $\tau = 10^{-6}s$
- $\theta = 1.92 \times 10^{-2}$ radians
- $L_r = 0.603$
- $L_m = 1$

$C = 9.559 \times 10^{13}$

b. Using the following equation to calculate Minimum Detectable Signal:

$$MDS = 10Log_{10}(\frac{kT}{1mW}) + 10Log_{10}(\frac{B}{1Hz}) + 10 * Log_{10}(F)$$

And with:

- $k = 1.38 \times 10^{-23}(\frac{W\dot{s}}{K})$
- $T = T_{room} = 293K$
- $B = 0.3MHz, 0.75MHz, 1MHz$
- $F = 3.4dB$

We get:

B	MDS
0.3MHz	-140.4dBm
0.75MHz	-136.5dBm
1MHz	-135.2dBm

c. Using the following equation to calculate reflectivity:

$$Z = C + P_r + 20 * Log_{10}(r) = 10Log_{10}(C) + MDS + 20 * Log_{10}(r)$$

We get:

Z	r
27.6dBZ	10km
33.6dBZ	20km
37.1dBZ	30km
39.6dBZ	40km

d. The only thing that changes with the X-Band Chill is that $\lambda = 3cm$, therefore $C = 6.5 \times 10^{10}$ and using the same MDS value:

Z	r
-4.1dBZ	10km
1.9dBZ	20km
5.4dBZ	30km
7.9dBZ	40km

4. **a.** $\tau = 400ns$
   $PRT = \tau \frac{P_{peak}}{P_{ave}}$
   $f_p = \frac{1}{PRT} = 3750Hz$
   Range resolution: $\Delta r = \frac{c*\tau}{2} = 60m$
   Max. Unambiguous range: $\frac{c}{2f_p} = 40,000m$

   b. With:

   - $G = 42dB$
   - $\tau = 400ns$
   - $\theta = 1.4°$
   - $f = 9410MHz$
   - $P_t = P_{peak} = 8.0kW$
   - $\lambda = 0.0319m$

   $C = 6.2 \times 10^{12} = 127.91dB$

   c. Using the MDS equation, with $F = 5dB$ and $B = 1Mhz$
   $MDS = -108.93dBm$

   d. The values of reflectivity using the above results are:

Z	r
39.0dBZ	10km
45.0dBZ	20km
48.5dBZ	30km
51.0dBZ	40km

5. $G = 42dB$
   $P_t = 8.0kW$
   $R = 10km$
   $D = 3mm$

   a. Power transmitted: $\frac{P_t G_t}{4\pi R^2} = 0.1009\frac{W}{m^2}$

   b. Because TropiNet is X-Band, $\lambda = 3cm$ and the drop of rain falls in the Rayleigh region. Therefore,

$$\sigma_b = \frac{\pi^5}{\lambda^4}|K|^2 D^6$$

   with $|K|^2 = 0.93$ since the target is water. The result is:

$$\sigma_b = 2.6 \times 10^{-7}$$

Power density at the target:

$$\frac{P_t G_t}{4\pi R^2}\frac{\sigma_b}{4\pi R^2} = 2.06 \times 10^{-17} W/m^2$$

c. With:

$$P_r = \frac{P_t \sigma \lambda^2}{(4\pi)^3 R^4} G^2 = 4.9 \times 10^{-20}$$

d. Using:

$$Z = \frac{nD^6}{V_c}$$

and solving for n we obtain that

$$n = 1.0846 \times 10^{26}$$

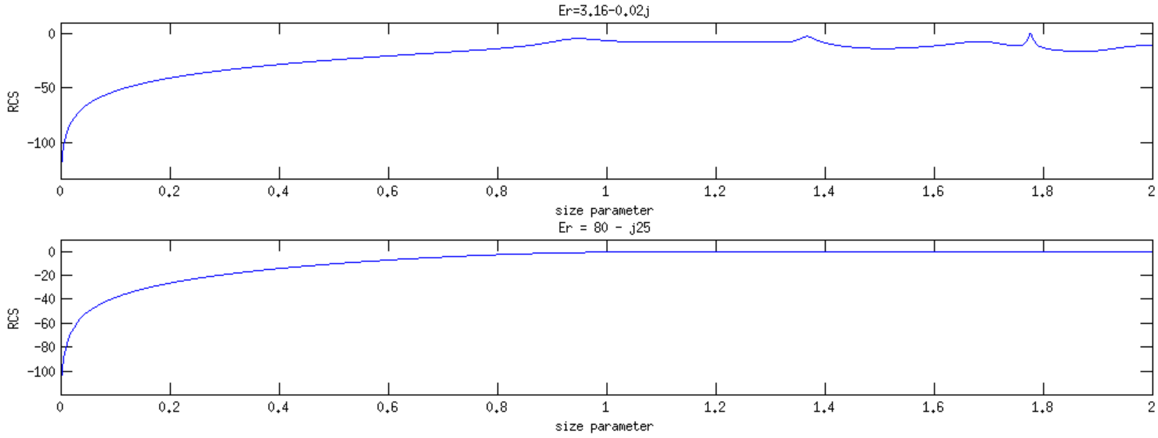
drops of water.

6. For TropiNet, we calculated earlier that $R_{un,max} = 40km$, while for Nexrad, using its PRF we obtain $R_{un,max} = 355.45km$. Assuming the target is above the radar, $H_0 = 0$. The radius of the Earth is approximated by $R = 6374km$. Using

$$H = \sqrt{R_{un,max}^2 + R'^2 + 2 * R_{un,max} R' \sin(\phi)} - R'$$

- For H = 1km
  TropiNet $\phi = 1.19°$ Nexrad $\phi = -1.97°$
- For H = 3km
  TropiNet $\phi = 4.06°$ Nexrad $\phi = -1.65°$

7. Using a function in matlab, here are the radar cross section plots using Mie's backscattering formulas:



The first plot represents the Mie solution for water particles, since $\epsilon_r = 80 - j25$. The second plot represents the Mie solution for ice particles, since $\epsilon_r = 3.16 - 0.02j$. As the size parameter increases, the normalized radar cross section approaches zero.

Appendix

atmospheric-sounding.py

This is the python code used to fetch and process the atmospheric sounding data:

```
#!bin/python

#Author: Sebastiani Aguirre-Navarro
#Date: 3-March-2015 @ 9:49 PM
#atmospheric-sounding.py

from argparse import ArgumentParser
from bs4 import BeautifulSoup
from math import exp
import numpy as np
import matplotlib.pyplot as plt
import requests


#constants
SERVICE_URL = "http://www.weather.uwyo.edu/cgi-bin/sounding?region=naconf&TYPI
HEADERS = {"User-Agent": "Mozilla/5.0 (X11; U; Linux i686) Gecko/20071127 Fir

#utility functions

class SoundingTable:
    def __init__(self):
        self.name = ''
        self.tableHeaders = ''
        self.numericData = np.array([])
        self.refractivities = np.array([])
        self.gradient_N = np.array([])
        self.no_vapor_Ns = np.array([])

    def plot_all(self):
        heightKM = np.array([row[1]/1000.0 for row in self.numericData])
        temperature = np.array([row[2] for row in self.numericData])
        dew_temperature = np.array([row[3] for row in self.numericData])

        #plotting Height vs Temperature
        plt.title("Sounding for %s-December-2014" % self.name)
        plt.scatter(temperature, heightKM)
        plt.plot(temperature, heightKM)
        plt.scatter(dew_temperature, heightKM, color="red")
```

```
        plt.plot(dew_temperature, heightKM, 'r')
        plt.xlabel('Temperature(C)')
        plt.ylabel('Height(km)')
        plt.grid(True)
        plt.savefig('%s-height-temperature.png' % self.name)
        plt.clf()

        #plotting Height vs Refractivity without vapor
        plt.title("Sounding for %s-December-2014"% self.name)
        plt.scatter(self.no_vapor_Ns, heightKM, color="green")
        plt.plot(self.no_vapor_Ns, heightKM, 'g')
        plt.scatter(temperature, heightKM)
        plt.plot(temperature, heightKM)
        plt.xlabel('Refractivity(N-Units)')
        plt.ylabel('Height(km)')
        plt.grid(True)
        plt.savefig('%s-height-refractivity-no-Vapor.png' % self.name)
        plt.clf()

        #plotting Height vs Refractivity
        plt.title("Sounding for %sZ-December-2014" % self.name)
        plt.scatter(self.refractivities, heightKM)
        plt.plot(self.refractivities, heightKM)
        plt.xlabel('Refractivity(N-Units)')
        plt.ylabel('Height(km)')
        plt.grid(True)
        plt.savefig('%s-height-refractivity.png' % self.name)
        plt.clf()
        height = np.array([row[1] for row in self.numericData][:-1])


        #plotting Height vs Gradient of Refractivity
        plt.title("Sounding for %sZ-December-2014" % self.name)
        plt.scatter(self.gradient_N, height)
        plt.plot(self.gradient_N, height)
        plt.xlabel('gradient of Refractivity(N-Units)/km')
        plt.ylabel('Height(m)')
        plt.grid(True)
        plt.savefig('%s-height-gradient_refractivity.png' % self.name)
        plt.clf()


def refractivity(temp, pressure, dew_pointT):
    #Transform temp from C -> K
```

```python
    temp_K = temp + 273.15

    #1 hPa = 1 mbars, so no conversion required

    x = dew_pointT*(18.729 - dew_pointT/227.3)/(dew_pointT + 257.87)
    Pw = 6.1121*exp(x)

    #return refractivity
    return (77.6/temp_K)*(pressure + (4810*Pw)/temp_K)

def noVaporRefractivity(temp, pressure):
    temp_K = temp + 273.15
    return (77.6/temp_K)*pressure

#options parser
parser = ArgumentParser()
parser.add_argument("--year", type=str, help="year of interest")
parser.add_argument("--month", type=str, help="month of interest")
parser.add_argument("--From", type=str, help="from Day/{00 or 12}")
parser.add_argument("--to", type=str, help="to Day/{00 or 12}")
parser.add_argument("--station", type=str, default="78526", help="station num
args = parser.parse_args()

#fetching data
url = SERVICE_URL % (args.year, args.month, args.From, args.to,
args.station)
req = requests.get(url, headers=HEADERS)
html_text = BeautifulSoup(req.text)

pre_info = html_text.find_all('pre')[::2]
tables = list()
names = ["2900Z", "2912Z"]
k=0
for pre_tag in pre_info:
    table = SoundingTable()
    table.name = names[k]
    k += 1
    pre_split = pre_tag.string.split("------------------------------------------
    table.tableHeaders = pre_split[1]
    string_data = pre_split[2].strip().split('\n')
    string_data = [filter(lambda x: x != '', sublist) for sublist in map(lamb
    table.numericData = np.array([map(lambda x: float(x), sublist) for sublis
    tables.append(table)
```

```
print "creating refractivity array"
for table in tables:
    table.refractivities = np.array([refractivity(row[2], row[0], row[3]) for
    table.no_vapor_Ns = np.array([noVaporRefractivity(row[2], row[0]) for row
    gradient_N = list()
    print "creating gradients"
    i = 0
    length = table.refractivities.size - 1
    while i < length:
        grad = (table.refractivities[i+1] - table.refractivities[i])/(table.n
        gradient_N.append(grad)
        i += 1
    table.gradient_N = gradient_N

    print "plotting..."
    table.plot_all()
```

runmie.m and mie.m

The code is divided in two parts, one defines the function, the other runs it and constructs the plots. This code is based on that of a paper:
http://arrc.ou.edu/~rockee/NRA_2007_website/Mie-scattering-Matlab.pdf

runmie.m

```
clear all
close all
close hidden

i = 1;
for k=0:0.001:2
    sigma = mie(k, 3.16-1i*0.02, 0);
    sigs(i) = sigma;
    i = i + 1;
end

figure
ax1 = subplot(2,1,1)
i=0:0.001:2;
sigs = sigs/max(sigs);
sigdB = 10*(log10(sigs));
plot(ax1, i, sigdB)
```

```
l=1;
for m=0:0.001:2
    sigma = mie(m, 80 - 25*1i, 0);
    sigs(l) = sigma;
    l = l + 1;
end


ax2 = subplot(2,1,2)

sigs = sigs/max(sigs);
sigdB = 10*(log10(sigs));
plot(ax2, i, sigdB)
axis([ax1, ax2], [0 2 -Inf 10])
```

mie.m

```
function [sig] = mie(x,refrel,nang)



 mxang=1500;
 nmxx=150000;

 s1=zeros(1,2*mxang-1);
 s2=zeros(1,2*mxang-1);
 d=zeros(1,nmxx);
 amu=zeros(1,mxang);
 pi=zeros(1,mxang);
 pi0=zeros(1,mxang);
 pi1=zeros(1,mxang);
 tau=zeros(1,mxang);

 if (nang < 2)
   nang = 2;
 end

  pii = 4.*atan(1.);
  dx = x;

  drefrl = refrel;
  y = x*drefrl;
  ymod = abs(y);
```

```
xstop = x + 4.*x^0.3333 + 2.;
nmx = max(xstop,ymod) + 15;
nmx=fix(nmx);


      nstop = xstop;



      dang = 0.;
      if (nang > 1)
        dang = .5*pii/ (nang-1);
      end
      for j=1: nang
          theta =  (j-1)*dang;
          amu(j) = cos(theta);
      end
      for j=1: nang
          pi0(j) = 0.;
          pi1(j) = 1.;
      end

      nn = nmx - 1;
      for n=1: nn
          en = nmx - n + 1;
          d(nmx-n) = (en/y) - (1./ (d(nmx-n+1)+en/y));
      end

      psi0 = cos(dx);
      psi1 = sin(dx);
      chi0 = -sin(dx);
      chi1 = cos(dx);
      xi1 = psi1-chi1*1i;
      sig =0.;
      gsca = 0.;
      p = -1;
      for n=1: nstop
          en = n;
          fn = (2.*en+1.)/ (en* (en+1.));

          psi = (2.*en-1.)*psi1/dx - psi0;
          chi = (2.*en-1.)*chi1/dx - chi0;
```

```
 xi = psi-chi*1i;

 if (n > 1)
     an1 = an;
     bn1 = bn;
end

 an = (d(n)/drefrl+en/dx)*psi - psi1;
 an = an/ ((d(n)/drefrl+en/dx)*xi-xi1);
 bn = (drefrl*d(n)+en/dx)*psi - psi1;
 bn = bn/ ((drefrl*d(n)+en/dx)*xi-xi1);

 sig = sig + (2.*en+1.)* (abs(an)^2+abs(bn)^2);
 gsca = gsca + ((2.*en+1.)/ (en* (en+1.)))* ...
    ( real(an)* real(bn)+imag(an)*imag(bn));

 if (n > 1)
              gsca = gsca + ((en-1.)* (en+1.)/en)*...
            ( real(an1)* real(an)+imag(an1)*imag(an)+...
              real(bn1)* real(bn)+imag(bn1)*imag(bn));

 end

 for j=1: nang

     pi(j) = pi1(j);
     tau(j) = en*amu(j)*pi(j) - (en+1.)*pi0(j);
     s1(j) = s1(j) + fn* (an*pi(j)+bn*tau(j));
     s2(j) = s2(j) + fn* (an*tau(j)+bn*pi(j));
 end

 p = -p;
 for j=1: nang-1
     jj = 2*nang - j;
     s1(jj) = s1(jj) + fn*p* (an*pi(j)-bn*tau(j));
     s2(jj) = s2(jj) + fn*p* (bn*pi(j)-an*tau(j));
 end
 psi0 = psi1;
 psi1 = psi;
 chi0 = chi1;
 chi1 = chi;
 xi1 = psi1-chi1*1i;

 for j=1: nang
```

```
            pi1(j) = ((2.*en+1.)*amu(j)*pi(j)- (en+1.)*pi0(j))/...
                en;
            pi0(j) = pi(j);
        end
end


sig = (2./ (dx*dx))*sig;

clear s1 s2
```