APG2014S ASSIGNEMENT 1

Yaseen Hull – HLLYAS001

AUGUST 1, 2016

UNIVERSITY OF CAPE TOWN

Table of Contents

Symbol chart:

Symbol	Explanation
M	**Meridian radius of curvature**
N	**Prime vertical radius of curvature**
S	**Distance between two points**
φ	**Geodetic Mean latitude**
α	**Mean Azimuth**
λ	**Geodetic Mean longitude**
φ_1	**First point Geodetic latitude**
φ_2	**Second point Geodetic latitude**
α_1	**Forward azimuth**
α_2	**Reverse Azimuth**
λ_1	**First point Geodetic longitude**
λ_2	**Second point Geodetic longitude**
τ	**tan(φ)**
η^2	$e'^2 \times \cos(\phi)^2$

Method and Theory

Inverse problem

The basic principle of the problem is to obtain a distance, forward and reverse azimuth between two known points of latitude and longitude ($\varphi_1, \lambda_1 : \varphi_2, \lambda_2$). It relies on the solution of the differences ($\Delta\varphi, \Delta\lambda$) as well as mid point latitudes to evaluate specifics of the formula. The process is outlined below using the related formulas in appendix 1A:

Known: (φ_1, λ_1) and (φ_2, λ_2)

Unknown: α_1, α_2 and S

a) **calculate $\Delta\varphi = (\varphi_2 - \varphi_1)$ and $\Delta\lambda = \lambda_2 - \lambda_1$**
b) **Calculate Scos(α) and Ssin(α)**
c) **Calculate S**
d) **Calculate the mean azimuth α =arctan((Ssin(α))/(Scos(α)))**
e) **Calculate $\Delta\alpha = (\alpha_2 - \alpha_1)$ from the direct problem expression**
f) **Finally calculate $\alpha_1 = \alpha - \Delta\alpha$ and $\alpha_2 = \alpha + \Delta\alpha$**

Coding aspect

The complexity of inverse problem solely relies on the interpretation of the formulas and the use of inputting the correct values to evaluate the Inverse formulas. Within the Inverse Problem.py file the code was constructed to be self-contained and to adapt other files used as input. Importing text files are easy, the issue lies in splitting and iterating through each value to create a reference or key for the value. To make things simpler, dictionaries were populated based on the identity of the value.

```
"""
Created on Sat Jul 23 22:08:09 2016
Inverse Problem
@author: yasee_000
"""
import math
#import numpy as np

db = {}
dc = {}
values=[]
d={}
f = open("anaconda.txt","r")
data = f.read()
spl = data.splitlines()
i = 0
while i < len(spl):
    for i in spl:
        values = i.split('\t')
        sta =  values.pop(0)
        values = map(float,values)
        d[sta]=[values.pop(0),values.pop(0),values.pop(0)]
        d2 = {sta:(values.pop(0))-(values.pop(0)/60)-(values.pop(0)/3600)}
        update1 = db.update(d2)
        d3 = {sta:(values.pop(0))+(values.pop(0)/60)+(values.pop(0)/3600)}
        update2 = dc.update(d3)

f.close()
```

The file 'anaconda.txt' (appendix 1B), was manually created to import the necessary information for processing the Inverse problem. Once the text file was opened, everything in the file was read and

added to a list to be split based on the symbol “\n” which is the character used for the “enter” function on the keyboard. Using the splitlines () function each line is taken into consideration when preparing to perform operations and now each line is an element in the list. To loop through the list and “while loop” is used based on the length of the list and for each element in the list it is split based on the character “\t” which is used to indicate a tab space. The first value is then taken out the line using the pop () function and the rest of the values in the line is mapped to float for calculation purposes.

Three dictionaries are initiated and values are added to them based on their identity. The first dictionary has key of the station name and its items are the X, Y and Z values of the point.

The second dictionary contains the latitude values converted to decimal degrees with the associated keys and the third contains the longitude values converted to decimal degrees with associated keys. For both the second dictionary a mirror dictionary is made as to save the values iterating through. Hence dictionary “db” in the above code contains all the latitude values for text file not only one line. The same principle applies to the other dictionaries.

Once constants (a,b,etc) were set, the two v and w files in the image were used to iterate through to reference values in the dictionaries. The v and w files are lists created with the dictionary elements inside them. For instance v contains all the items in the db dictionary. This is, v contains all the keys and latitudes of the dictionary db in list format. Same applies to w expect that w contains all the latitude. This allows us to iterate through the list with simple referencing quotes. (e.g v[0][1] = 1st latitude result).

```
31
32
33 f2 =open('boa.txt','w')
34 f3 =open('viper.txt','w')
35 a = 6378137
36 b = 6356752.314
37 f = 1/298.257224
38 e2 = ((a**2)-(b**2))/(a**2)
39 e2x = ((a**2)-(b**2))/(b**2)
40 n = 0.0001679220
41
42 v = db.items()
43 w =dc.items()
44
45
46 for i in range(len(v)):
47     for k in range(len(v)):
48         dlat = math.radians(v[i][1]-v[k][1])
49         dlong = math.radians(w[i][1]-w[k][1])
50
51         dD = int(v[k][1])
52         mM = int((v[k][1] - dD)*60)
53         sS = round((((v[k][1] - dD)*60)-mM)*60,4)
54         latTo = str(dD)+'\t'+str(-1*mM)+'\t'+str(-1*sS)
55         latto =v[k][1]
56
57         dl = int(w[k][1])
58         ml = int((w[k][1] - dl)*60)
59         sl = round((((w[k][1] - dl)*60)-ml)*60,4)
60         longTo = str(dl)+'\t'+str(ml)+'\t'+str(sl)
61         longto = w[k][1]
62
63         nameLat = v[i][0]+'-'+v[k][0]
64         nameLong = w[i][0]+'-'+w[k][0]
65         nameFrm = v[k][0]
66         nameTo = v[i][0]
67
68         mLat = math.radians((v[i][1]+v[k][1]))/2.0
69         mLong = math.radians((w[i][1]+w[k][1]))/2.0
70
```

To calculate the $\Delta\varphi$ and$\Delta\lambda$ a nested loop was created. The first “for loop” sets a specific value in the list. The second “for loop” is used to obtain all the other values of the list v for the second value while the first for loop keeps the first value constant. Hence we can get all the differences between the each latitudes in the list. Same applies to the longitude. Simultaneously these differences are converted to radians for the Inverse formulas to use. Some other simple formula is also used to reference names, and convert to Degrees, minutes, seconds. The nested loop also takes advantage of the referencing system to get mid latitudes and mid longitudes. The inverse calculations are then undertaken under the conditions that the delta lat and delta long are not equal to zero. In the same passage of conditions the name of the latitude must match the name of the longitude to ensure that the correct values are being referenced (line 71 of inverse problem.py).

Based on the sign (negative or positive) of the Ssinα and Scosα, forward azimuths and reverse azimuths are oriented in radians and then converted to Degrees, minutes and seconds. Once all this is done the values are written to a text file. Remember that we are still in the loop so each line is processed and then written to the text file.

```
85
86
87              x = math.pi
88              if SsinA>0 and ScosA<0:
89                  DegF = math.degrees(DegF +2*x)
90                  DegB = math.degrees(DegB +x)
91              elif SsinA<0 and ScosA<0:
92                  DegF =math.degrees(DegF)
93                  DegB =math.degrees(DegB+x)
94              elif SsinA<0 and ScosA>0:
95                  DegF = math.degrees(DegF +x)
96                  DegB = math.degrees(DegB +2*x)
97              else:
98                  DegF =math.degrees(DegF+x)
99                  DegB = math.degrees(DegB)
100
101             DD = int(DegF)
102             MM = int((DegF - DD)*60)
103             SS = round((((DegF - DD)*60)-MM)*60,2)
104
105             Dd = int(DegB)
106             Mm = int((DegB - Dd)*60)
107             Ss = round((((DegB - Dd)*60)-Mm)*60,2)
108             Azi = str(DD)+'\t'+str(MM)+'\t'+str(SS)+'\t'+str(Dd)+'\t'+str(Mm)+'\t'+str(Ss)
109
110             f2.write(nameLat+'\t'+latTo+'\t'+longTo+'\t'+str(S)+'\t'+Azi+'\n')
111             f3.write(nameLat+'\t'+str(math.radians(latto))+'\t'+str(math.radians(longto))+'\t'+str(S)+'\t'
112 f2.close
```

Direct problem

The direct problem assumes a (φ_2,λ_2) and reverse α after knowing the values of the first point latitude, longitude , forward azimuth to second point and distance to second point . Although to calculate a second an approximation of the mid latitude and mid azimuth must be made to initiate the process. Furthermore the results obtained for delta lat must be reused and this process must be iterated until a stable delta lat and delta azimuth is obtained. Then only the longitude formula can be processed. The process is outlined below based on the formulas in appendix 1B:

Known: φ_1, λ_1, α_1,S

Unknown: φ_2, λ_2, α_2

a) **make initial guess of the mean latitude and mean azimuth (φ,α) i.e {since φ_1 known: $\varphi = \varphi_1$ and $\alpha = \alpha_1$}**
b) **make formulas ; $\alpha_2 - \alpha_1 = \Delta\alpha$ and $\varphi_2 - \varphi_1 = \Delta\varphi$**
c) **substitute the values of the mean latitude(φ) and mean azimuth(α) into the formulae, solve for $\Delta\varphi$, $\Delta\alpha$ under a loop that will iterate till, there is no change in the value of α_2 and φ_2 where $\alpha_2 = \alpha_1 + \Delta\alpha$ and $\varphi_2 = \varphi_1 + \Delta\varphi$**
d) **Finally solve for λ_2 from $\lambda_2 - \lambda_1$ directly.**

Coding aspect

Since the direct problem uses forward azimuth, distance, latitude 1 and longitude 1 of the starting point, results from the Inverse text file ("viper.txt") was imported into the python Direct problem file. All values expect distance in the viper text file have been converted to radians for direct use in the direct problem. The viper file can be seen in Appendix output 1B.

A similar process as to the inverse problem was done for the direct with regards to reading files and and obtaining each line as an element in a list. Furthermore these elements were split by tab characters and these values were appended to one dictionary for referencing. Once the dictionary containing; latitude 1, longitude 1, distance, forward and reverse azimuth was a dictionary.items() method was again used to encapsulate the dictionaries for numerical referencing.

```
'''Direct Problem
   Yaseen Hull '''


import math
d1= {}
dDis = {}
values=[]
f = open("viper.txt","r")
data = f.read()
spl = data.splitlines()
i = 0
while i < len(spl):
    for i in spl:
        values = i.split('\t')
        staFrm =  values.pop(0)
        values = map(float,values)
        glat =  values.pop(0)
        glong = values.pop(0)
        eS = values.pop(0)
        fAzi = values.pop(0)
        bAzi = values.pop(0)
        d1[staFrm] = [glat,glong,eS,fAzi,bAzi]
        update3 = dDis.update(d1)

a = 6378137
b = 6356752.314
f = 1/298.257224
e2 = ((a**2)-(b**2))/(a**2)
e2x = ((a**2)-(b**2))/(b**2)
n = 0.0001679220


x = dDis.items()
```

A “for loop” was then created based on the length of list x in the image above to iterate through the values of the list. Below we can see a sample of how this was done:

```
x = dDis.items()


for i in range(len(x)):
    lat1= (x[i][1][0]) #obtaining lat
    fAzi1 = (x[i][1][3]) #forward azimuth to point 2
    name = x[i][0] #name of point 2'''
    S  = x[i][1][2]

phi2 = lat1
alpha2 = fAzi1
```

First values for latitude 2 and for reverse azimuth had to be approximated and the initial values were set to latitude 1 and forward azimuth respectively. Initialising these variables here aids us in resetting these values once an iterated loop is computed. The “while loop” with an iteration of 20 runs was constructed containing all the formulas for the direct problem. Hence this second latitude and reverse azimuth would be calculated 20 times before a stable result was achieved to input in to the longitude formula. Refer t code in appendices.

Results

Inverse problem

The inverse problem was rather simple considering complex formulas and iterations that had to be processed. With respect to the gauss mid latitude formulas it is the easier of the two (inverse and direct) to code. The challenge in the Inverse problem orientating forward and reverse azimuths in the code as python does not understand the concept of revolution is equal to 360 degrees. Instead radians values were used and by importing math modules for python it was possible to make use of the value of pi in place of adding degrees, since pi = 180 degrees on a circle. The results obtained are in reasonable range of the provided distances and azimuths (refer to appendix output1A). Taking a closer look at the results we see both the distances and directions are minimal metres and seconds to minimal minutes off respectively. This could be due to radians conversions with in python or due to the number of terms used in the formula. If we further expand the formula in appendix 1A by the Taylor series theorem we can get more accurate results.

Direct problem

The direct problem was fairly problematic since many iterations had to be made. And since the values obtained from one iteration had to be reused in the same formula by resetting the variables of lat2 and reverse azimuth. Even though results were obtained, the approximate second latitude obtained seemed to be fairly close to the latitude 1 for the formula. Hence the Direct problem was not able execute correctly.

Formula Sheet

General formulas

All evaluated for the mean value of latitude:

Meridian radius curvature

$$M = \frac{a(1-e^2)}{\sqrt{(1-e^2(\sin\phi)^2)^3}}$$

The equation for the Prime vertical radius curvature

$$N = \frac{a}{\sqrt{1-e^2(\sin\phi)^2}}$$

The equation for Tau

$\tau = \tan\phi$

The equation of Eta

$$\eta^2 = e'^2\cos^2\phi$$

Appendix 1A

$$S\cos\alpha = M\Delta\phi + \frac{M}{24}\Delta\phi\left[(3\eta^2 - 3\eta^2\tau^2 + 3\eta^4)\frac{\Delta\phi^2}{1+\eta^2} - (2+3\tau^2+2\eta^2)\Delta\lambda^2(\cos\phi)^2\right] + \ldots\ldots$$

$$S\cos\alpha = N\cos\phi\Delta\lambda + \frac{N\cos\phi\Delta\lambda}{24}[(1+\eta^2-9\eta^2\tau^2)\frac{\Delta\phi^2}{1+\eta^2} - \tau^2\Delta\lambda^2(\cos\phi)^2] + \ldots\ldots \qquad S = \sqrt{(S\cos\alpha)^2 + (S\sin\alpha)^2}$$

$$\alpha = \tan^{-1}(\frac{S\sin\alpha}{S\cos\alpha})$$

$\alpha_1 = \alpha - 0.5(\alpha_2 - \alpha_1)$ forward azimuth

$\alpha_2 = \alpha + 0.5(\alpha_2 - \alpha_1)$ reverse azimuth

$\alpha_2 - \alpha_1$ – found in direct problem.

Appendix 1B

$$\phi_2 - \phi_1 = \frac{S}{M}(cos\alpha) + \frac{S^3}{24M^3}\left(\frac{1}{(1-\eta^2)^2}\right)[(2+3\tau^2+2\eta^2)(sin\,\alpha)^2 cos\alpha + 3(-\eta^2+3\eta^2\tau^2-\eta^4)] + \cdots$$

$$\lambda_2 - \lambda_1 = \frac{S}{Ncos\phi}(sin\alpha) + \frac{S^3}{24N^3 cos\phi}[(\tau^2(sin\alpha)^3 + (-1-\eta^2+9\eta^2\tau^2)(\cos\alpha)^2 sin\alpha]$$

$$a_2 - a_1 = \frac{S}{N}\tau sin\alpha + \frac{S^3}{24N^3}\tau[(2+\tau^2+2\eta^2)(\sin\alpha)^3 + (2+7\eta^2+9\eta^2\tau^2+5\eta^4)(\cos\alpha)^2 sin\alpha] + \cdots$$

Appendix output 1A

Inverse results:

staTo -StaFrm	DD	MM	SS	DD	MM	SS	S	DD	MM	SS	DD	MM	SS
PRET-HNUS	-34	25	28.6671	19	13	23.0264	1299279.84551	224	33	51.1	39	59	18.08
PRET-TDOU	-23	4	47.67	30	23	2.43	362919.495145	36	24	11.83	215	32	3.96
PRET-RBAY	-28	47	43.96	32	4	42.19	506395.936225	131	12	13.94	312	56	31.6
PRET-ULDI	-28	17	15.33	31	25	15.33	420902.863957	131	33	12.65	312	58	42.35
HNUS-PRET	-25	43	55.2935	28	16	57.4873	1299279.84551	44	33	51.1	219	59	18.08
HNUS-TDOU	-23	4	47.67	30	23	2.43	1661651.44576	43	46	43.55	218	20	27.2
HNUS-RBAY	-28	47	43.96	32	4	42.19	1368704.36284	66	24	19.24	239	36	52.05
HNUS-ULDI	-28	17	15.33	31	25	15.33	1343841.95095	62	55	17.72	236	31	33.54
TDOU-PRET	-25	43	55.2935	28	16	57.4873	362919.495145	216	24	11.83	35	32	3.96
TDOU-HNUS	-34	25	28.6671	19	13	23.0264	1661651.44576	223	46	43.55	38	20	27.2
TDOU-RBAY	-28	47	43.96	32	4	42.19	655542.324593	164	37	8.28	345	21	28.25
TDOU-ULDI	-28	17	15.33	31	25	15.33	586230.264771	169	32	42.22	349	59	34.09
RBAY-PRET	-25	43	55.2935	28	16	57.4873	506395.936225	311	12	13.94	132	56	31.6
RBAY-HNUS	-34	25	28.6671	19	13	23.0264	1368704.36284	246	24	19.24	59	36	52.05
RBAY-TDOU	-23	4	47.67	30	23	2.43	655542.324593	344	37	8.28	165	21	28.25
RBAY-ULDI	-28	17	15.33	31	25	15.33	85493.4681176	311	1	34.22	131	20	25.08
ULDI-PRET	-25	43	55.2935	28	16	57.4873	420902.863957	311	33	12.65	132	58	42.35
ULDI-HNUS	-34	25	28.6671	19	13	23.0264	1343841.95095	242	55	17.72	56	31	33.54
ULDI-TDOU	-23	4	47.67	30	23	2.43	586230.264771	349	32	42.22	169	59	34.09
ULDI-RBAY	-28	47	43.96	32	4	42.19	85493.4681176	131	1	34.22	311	20	25.08

Appendix output 1B

staTo -StaFrm	lat1	long1	S	forward azi	reverse azi
PRET-HNUS	-0.600823132917	0.335505739729	1299279.84551	3.9193845886	0.697928451691
PRET-TDOU	-0.402820391475	0.53030098537	362919.495145	0.635357205928	3.7617855316
PRET-RBAY	-0.50257706046	0.559873456365	506395.936225	2.28993956829	5.46187018654
PRET-ULDI	-0.493711612043	0.548398595272	420902.863957	2.29604193635	5.46250410097
HNUS-PRET	-0.449108576424	0.493625108192	1299279.84551	0.777791935013	3.83952110528
HNUS-TDOU	-0.402820391475	0.53030098537	1661651.44576	0.764083560516	3.81076739368
HNUS-RBAY	-0.50257706046	0.559873456365	1368704.36284	1.15899188808	4.18206125182
HNUS-ULDI	-0.493711612043	0.548398595272	1343841.95095	1.09818887702	4.12815716252
TDOU-PRET	-0.449108576424	0.493625108192	362919.495145	3.77694985952	0.620192878012
TDOU-HNUS	-0.600823132917	0.335505739729	1661651.44576	3.90567621411	0.669174740093
TDOU-RBAY	-0.50257706046	0.559873456365	655542.324593	2.87314299998	6.02763152966
TDOU-ULDI	-0.493711612043	0.548398595272	586230.264771	2.95911955051	6.10852678503
RBAY-PRET	-0.449108576424	0.493625108192	506395.936225	5.43153222188	2.32027753295
RBAY-HNUS	-0.600823132917	0.335505739729	1368704.36284	4.30058454167	1.04046859823
RBAY-TDOU	-0.402820391475	0.53030098537	655542.324593	6.01473565357	2.88603887607
RBAY-ULDI	-0.493711612043	0.548398595272	85493.4681176	5.42843075493	2.29232069928
ULDI-PRET	-0.449108576424	0.493625108192	420902.863957	5.43763458994	2.32091144738
ULDI-HNUS	-0.600823132917	0.335505739729	1343841.95095	4.23978153061	0.986564508929
ULDI-TDOU	-0.402820391475	0.53030098537	586230.264771	6.1007122041	2.96693413144
ULDI-RBAY	-0.50257706046	0.559873456365	85493.4681176	2.28683810134	5.43391335287

Appendix 2A

Direct results:

```
RBAY-HNUS -38"-31'-0.71"
ULDI-TDOU -17"-52'-0.18"
HNUS-PRET -17"-7'-0.91"
HNUS-RBAY -23"-17'-0.84"
ULDI-RBAY -29"-18'-0.01"
TDOU-HNUS -44"-26'-0.17"
TDOU-RBAY -34"-28'-0.96"
PRET-RBAY -31"-44'-0.68"
HNUS-ULDI -22"-16'-0.67"
PRET-HNUS -42"-17'-0.3"
TDOU-ULDI -33"-28'-0.96"
RBAY-ULDI -27"-46'-0.78"
RBAY-PRET -22"-40'-0.43"
RBAY-TDOU -17"-21'-0.75"
PRET-TDOU -20"-25'-0.72"
ULDI-HNUS -39"-10'-0.14"
PRET-ULDI -30"-45'-1.0"
HNUS-TDOU -11"-55'-0.31"
ULDI-PRET -23"-10'-0.76"
TDOU-PRET -28"-21'-0.09"
```

The results obtained are only the second latitudes, as these are wrong the longitude would also be incorrect. The naming conventions follows the ‘TO-From’ hence thses latitudes are for the first name in the line.

APG2014s

REPORT FOR

STUDENT: Yaseen Hull

STUDENT NUMBER: HLLYAS001

DECLARATION

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. These calculations/report/plans are my own work.

3. I have not allowed and will not allow anyone to copy my work with the intention of passing it off as his or her own work.

Signature: _____

Date: _____