```
from module import *
import pandas as pd
import numpy as np
from IPython.display import Latex, display_latex
```

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
PRECISION = 3
np.set_printoptions(precision=PRECISION, suppress=True)
pd.options.display.float_format = "{:.3f}".format
```

1 Introducción de datos

$$P\bar{1}ia - P\bar{1}ib = R_A \tan\frac{\alpha}{2} + R_B \tan\frac{\beta}{2}$$

$$P\bar{1}ib - P\bar{1}ic = R_B \tan\frac{\beta}{2} + R_C \tan\frac{\theta}{2}$$

```
# R -> Radios
# < -> Angulos en º
data = np.array(
    [
        # R           # <
        (194.4548865, np.deg2rad(53.30472)),
        (240.6567417, np.deg2rad(71.30472)),
        (128.4983846, np.deg2rad(42.06389)),
    ],
)
_ = pd.DataFrame(data, columns=['radios', 'angulos'])
__ = _.apply({'radios': lambda x: x, 'angulos': np.rad2deg})
radios = _['radios']
```

```
--
```

```
   radios  angulos
0 194.455   53.305
1 240.657   71.305
2 128.498   42.064
```

```
__.sort_values(by='radios')
```

```
   radios  angulos
2 128.498   42.064
0 194.455   53.305
1 240.657   71.305
```

2 Cálculo de subtangentes

```
subtangentes = st_vec(data)
for idx, row in __.iterrows():
    A = np.float32(row['angulos'])
    R = np.float32(row['radios'])
    string = f'$ {R} \\times \\tan{A / 2}º \\qquad = {round(subtangentes[idx], PRECISION)} $'
    display_latex(Latex(string))

_['subtangentes'] = subtangentes
```

```
angulos = copy.deepcopy(_['angulos'])
```

-

$$194.4548797607422 \times \tan 26.652360916137695ž \quad = 97.598$$

$$240.65673828125 \times \tan 35.65235900878906ž \quad = 172.626$$

$$128.49838256835938 \times \tan 21.031944274902344ž \quad = 49.408$$

```
   radios  angulos  subtangentes
0 194.455    0.930        97.598
1 240.657    1.245       172.626
2 128.498    0.734        49.408
```

3 Cálculo de diagonales y de cotas progresivas

```
# Calculo de diagonales
diag_vec(data[::-1], st_=True)
```

```
array([[ 162.046,    1.979],
       [1721.508,    2.909]])
```

```
# Calculo de cotas
cotas = p_total(data, pi=0)
print(cotas)
cotas = cotas + np.abs(np.min(cotas))
cotas += radios[0] * angulos[0] / 2 # <- Cca
pd.DataFrame(cotas.reshape(-1,1), columns=['cotas progresivas'])
```

```
[-1819.106 -1638.196 -1338.698 -1244.361]
```

```
   cotas progresivas
0             90.455
1            271.365
2            570.862
3            665.200
```

4 Elementos iniciales de cuadro de replanteo

```
expanded = expand__(data, pi=np.abs(np.min(cotas)) + 10_054.302, prec=10)
_ = pd.DataFrame(expanded, columns=['radio', 'angulo', 'cota', 'D'])
radio_pre0 = _['radio']
radio_pre0 = list(pd.concat([pd.Series([0]), radio_pre0[:-1]]))
radio_pre0 = np.array(radio_pre0)
_['radio_pre0'] = radio_pre0
# _['gamma'] = gamma_vec(expanded[:,1:3])
# _ = _.apply({'radio': lambda x: x, 'angulo': np.rad2deg, 'cota': lambda x: x, 'D': lambda x: x})
```

-

```
     radio  angulo     cota      D  radio_pre0
0  194.455   0.930 8325.651  0.000       0.000
1  194.455   0.930 8330.000  4.349     194.455
2  194.455   0.930 8340.000 10.000     194.455
3  194.455   0.930 8350.000 10.000     194.455
4  194.455   0.930 8360.000 10.000     194.455
..     ...     ...      ...    ...         ...
57 128.498   0.734 8870.000 10.000     128.498
58 128.498   0.734 8880.000 10.000     128.498
59 128.498   0.734 8890.000 10.000     128.498
60 128.498   0.734 8900.000 10.000     128.498
```

```
61   0.000   0.000 8900.396  0.396     128.498

[62 rows x 5 columns]
```

4.1 Cálculo de γ

```
_['gamma'] = _['D'] / (2 * _['radio'])
_
```

```
     radio  angulo     cota      D  radio_pre0  gamma
0  194.455   0.930 8325.651  0.000       0.000  0.000
1  194.455   0.930 8330.000  4.349     194.455  0.011
2  194.455   0.930 8340.000 10.000     194.455  0.026
3  194.455   0.930 8350.000 10.000     194.455  0.026
4  194.455   0.930 8360.000 10.000     194.455  0.026
..     ...     ...      ...    ...         ...    ...
57 128.498   0.734 8870.000 10.000     128.498  0.039
58 128.498   0.734 8880.000 10.000     128.498  0.039
59 128.498   0.734 8890.000 10.000     128.498  0.039
60 128.498   0.734 8900.000 10.000     128.498  0.039
61   0.000   0.000 8900.396  0.396     128.498    inf

[62 rows x 6 columns]
```

4.2 Cálculo de D acumulado

```
def D_acum(dataframe):
    llist = []
    acc = 0
    for idx, row in dataframe.iterrows():
        if row['radio'] != row['radio_pre0']:
            acc = row['D_sum']
        llist.append(row['D_sum'] - acc)
    return llist
```

```
_['D_sum'] = np.add.accumulate(_['D'])
# _['D_sum'] = D_acum(_)
# _['D_sum'] = D_sum(_)
_
```

```
     radio  angulo     cota      D  radio_pre0  gamma   D_sum
0  194.455   0.930 8325.651  0.000       0.000  0.000   0.000
1  194.455   0.930 8330.000  4.349     194.455  0.011   4.349
2  194.455   0.930 8340.000 10.000     194.455  0.026  14.349
3  194.455   0.930 8350.000 10.000     194.455  0.026  24.349
4  194.455   0.930 8360.000 10.000     194.455  0.026  34.349
..     ...     ...      ...    ...         ...    ...     ...
57 128.498   0.734 8870.000 10.000     128.498  0.039 544.349
58 128.498   0.734 8880.000 10.000     128.498  0.039 554.349
59 128.498   0.734 8890.000 10.000     128.498  0.039 564.349
60 128.498   0.734 8900.000 10.000     128.498  0.039 574.349
61   0.000   0.000 8900.396  0.396     128.498    inf 574.745

[62 rows x 7 columns]
```

4.3 Cálculo de γ acumulado

```
_['gamma']
```

```
0    0.000
1    0.011
2    0.026
3    0.026
4    0.026
      ...
57   0.039
58   0.039
59   0.039
60   0.039
61     inf
Name: gamma, Length: 62, dtype: float64
```

```
_['gamma_sum'] = np.add.accumulate(_['gamma'])
# _['gamma_sum'] = gamma_sum(_)
_
```

	radio	angulo	cota	D	radio_pre0	gamma	D_sum	gamma_sum
0	194.455	0.930	8325.651	0.000	0.000	0.000	0.000	0.000
1	194.455	0.930	8330.000	4.349	194.455	0.011	4.349	0.011
2	194.455	0.930	8340.000	10.000	194.455	0.026	14.349	0.037
3	194.455	0.930	8350.000	10.000	194.455	0.026	24.349	0.063
4	194.455	0.930	8360.000	10.000	194.455	0.026	34.349	0.088
..	...	...	...	...	...	...	...	...
57	128.498	0.734	8870.000	10.000	128.498	0.039	544.349	1.344
58	128.498	0.734	8880.000	10.000	128.498	0.039	554.349	1.383
59	128.498	0.734	8890.000	10.000	128.498	0.039	564.349	1.422
60	128.498	0.734	8900.000	10.000	128.498	0.039	574.349	1.461
61	0.000	0.000	8900.396	0.396	128.498	inf	574.745	inf

[62 rows x 8 columns]

```
radio_pre0 / 2
```

```
array([  0.   ,  97.227,  97.227,  97.227,  97.227,  97.227,  97.227,
        97.227,  97.227,  97.227,  97.227,  97.227,  97.227,  97.227,
        97.227,  97.227,  97.227,  97.227,  97.227,  97.227, 120.328,
       120.328, 120.328, 120.328, 120.328, 120.328, 120.328, 120.328,
       120.328, 120.328, 120.328, 120.328, 120.328, 120.328, 120.328,
       120.328, 120.328, 120.328, 120.328, 120.328, 120.328, 120.328,
       120.328, 120.328, 120.328, 120.328, 120.328, 120.328, 120.328,
       120.328, 120.328,  64.249,  64.249,  64.249,  64.249,  64.249,
        64.249,  64.249,  64.249,  64.249,  64.249,  64.249])
```

4.4 Cálculo de lc

```
# def lc_calc(dataframe):
arcos = angulos * radios
arcos = pd.concat((pd.Series((0,)),arcos))
```

```
arcos
```

```
0     0.000
0   180.910
1   299.498
2    94.338
dtype: float64
```

```
# Gamma_sum se encuentra en radianes
_['lc'] = ( np.sin(_['gamma_sum']) * radio_pre0 * 2 )
```

```
# _['lc'] = np.sqrt(_['lc'])
# _['lc1'] = lc_sum(_)
# _['lc'][19:] -= g_
# _['lc'][49:] -= q_
_['lc - D_sum'] = _['lc'] - _['D_sum']
_
```

```
c:\Users\Cesar\.conda\envs\ds\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: invalid valu
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

	radio	angulo	cota	D	radio_pre0	gamma	D_sum	gamma_sum	\
0	194.455	0.930	8325.651	0.000	0.000	0.000	0.000	0.000	
1	194.455	0.930	8330.000	4.349	194.455	0.011	4.349	0.011	
2	194.455	0.930	8340.000	10.000	194.455	0.026	14.349	0.037	
3	194.455	0.930	8350.000	10.000	194.455	0.026	24.349	0.063	
4	194.455	0.930	8360.000	10.000	194.455	0.026	34.349	0.088	
..	...	...	...	...	...	...	...	...	
57	128.498	0.734	8870.000	10.000	128.498	0.039	544.349	1.344	
58	128.498	0.734	8880.000	10.000	128.498	0.039	554.349	1.383	
59	128.498	0.734	8890.000	10.000	128.498	0.039	564.349	1.422	
60	128.498	0.734	8900.000	10.000	128.498	0.039	574.349	1.461	
61	0.000	0.000	8900.396	0.396	128.498	inf	574.745	inf	

	lc	lc - D_sum
0	0.000	0.000
1	4.349	-0.000
2	14.346	-0.003
3	24.333	-0.016
4	34.304	-0.045
..	...	...
57	250.414	-293.935
58	252.473	-301.876
59	254.149	-310.200
60	255.441	-318.908
61	NaN	NaN

[62 rows x 10 columns]

4.4.1 Cálculo de azimuts

Ya que utilizamos un sentido antihorario, tenemos que utilizar un azimut inverso al del útlimo punto y restando las diferencias generadas en lugar de sumarlas.

$$\text{Azimut}_{\text{inverso}} = 180\check{z} - \text{Azimut}$$

```
az_init = np.pi - np.deg2rad(110.38194444)
_['azimut_inv'] = _['gamma_sum'] + az_init
_
```

	radio	angulo	cota	D	radio_pre0	gamma	D_sum	gamma_sum	\
0	194.455	0.930	8325.651	0.000	0.000	0.000	0.000	0.000	
1	194.455	0.930	8330.000	4.349	194.455	0.011	4.349	0.011	
2	194.455	0.930	8340.000	10.000	194.455	0.026	14.349	0.037	
3	194.455	0.930	8350.000	10.000	194.455	0.026	24.349	0.063	
4	194.455	0.930	8360.000	10.000	194.455	0.026	34.349	0.088	
..	...	...	...	...	...	...	...	...	
57	128.498	0.734	8870.000	10.000	128.498	0.039	544.349	1.344	
58	128.498	0.734	8880.000	10.000	128.498	0.039	554.349	1.383	
59	128.498	0.734	8890.000	10.000	128.498	0.039	564.349	1.422	

```
60 128.498   0.734 8900.000 10.000     128.498  0.039 574.349      1.461
61   0.000   0.000 8900.396  0.396     128.498    inf 574.745        inf

        lc  lc - D_sum  azimut_inv
0    0.000       0.000       1.215
1    4.349      -0.000       1.226
2   14.346      -0.003       1.252
3   24.333      -0.016       1.278
4   34.304      -0.045       1.303
..     ...         ...         ...
57 250.414    -293.935       2.559
58 252.473    -301.876       2.598
59 254.149    -310.200       2.637
60 255.441    -318.908       2.676
61     NaN         NaN         inf

[62 rows x 11 columns]
```

4.5 Cálculo de coordenadas

$$N = N_{\text{anterior}} + L_c \times \cos(\text{Azimut}_{\text{inverso}}) E = E_{\text{anterior}} - L_c \times \sin(\text{Azimut}_{\text{inverso}})$$

```
N_init = 8_822_222
E_init = 482_777
_['delta_N'] = _[['azimut_inv', 'lc']].apply(lambda x: np.cos(x['azimut_inv']) * x['lc'], axis=1)
_['delta_E'] = _[['azimut_inv', 'lc']].apply(lambda x: -np.sin(x['azimut_inv']) * x['lc'], axis=1)
_['N'] = _['delta_N'] + N_init
_['E'] = _['delta_E'] + E_init
_
C:\Users\Cesar\AppData\Local\Temp\ipykernel_13260\2004008128.py:3: RuntimeWarning: invalid value encountere
  _['delta_N'] = _[['azimut_inv', 'lc']].apply(lambda x: np.cos(x['azimut_inv']) * x['lc'], axis=1)
C:\Users\Cesar\AppData\Local\Temp\ipykernel_13260\2004008128.py:4: RuntimeWarning: invalid value encountere
  _['delta_E'] = _[['azimut_inv', 'lc']].apply(lambda x: -np.sin(x['azimut_inv']) * x['lc'], axis=1)
```

```
     radio  angulo     cota      D  radio_pre0  gamma   D_sum  gamma_sum  \
0  194.455   0.930 8325.651  0.000       0.000  0.000   0.000      0.000
1  194.455   0.930 8330.000  4.349     194.455  0.011   4.349      0.011
2  194.455   0.930 8340.000 10.000     194.455  0.026  14.349      0.037
3  194.455   0.930 8350.000 10.000     194.455  0.026  24.349      0.063
4  194.455   0.930 8360.000 10.000     194.455  0.026  34.349      0.088
..     ...     ...      ...    ...         ...    ...     ...        ...
57 128.498   0.734 8870.000 10.000     128.498  0.039 544.349      1.344
58 128.498   0.734 8880.000 10.000     128.498  0.039 554.349      1.383
59 128.498   0.734 8890.000 10.000     128.498  0.039 564.349      1.422
60 128.498   0.734 8900.000 10.000     128.498  0.039 574.349      1.461
61   0.000   0.000 8900.396  0.396     128.498    inf 574.745        inf

        lc  lc - D_sum  azimut_inv  delta_N  delta_E           N          E
0    0.000       0.000       1.215    0.000   -0.000 8822222.000 482777.000
1    4.349      -0.000       1.226    1.469   -4.093 8822223.469 482772.907
2   14.346      -0.003       1.252    4.497  -13.623 8822226.497 482763.377
3   24.333      -0.016       1.278    7.031  -23.295 8822229.031 482753.705
4   34.304      -0.045       1.303    9.064  -33.085 8822231.064 482743.915
..     ...         ...         ...      ...      ...          ...        ...
57 250.414    -293.935       2.559 -209.111 -137.768 8822012.889 482639.232
58 252.473    -301.876       2.598 -216.074 -130.594 8822005.926 482646.406
59 254.149    -310.200       2.637 -222.458 -122.900 8821999.542 482654.100
```

```
60 255.441    -318.908       2.676 -228.224 -114.733 8821993.776 482662.267
61     NaN         NaN         inf       NaN      NaN         NaN        NaN

[62 rows x 15 columns]
```

5 Guardando resultados

```
_.to_excel("examen.xlsx", index=False)
```