1)работает клас парсера опций

2) в main вызывает функцию parse\_ascii\_input() в ней вызов класса Station.py и передача файла В Station.py проверка если фаил не пустой вызов метода init\_from\_ascii\_line в нем переводит градусы в радианы скорости и ошибки делит на 1e3 т.е. +0.0 возвращая объект затем проверка имя==имя lon==lon ругается если не уникально или возвращает список

3) далее в main есть ли опция –r region вызов тут же функцию Если поставленна опция -с или –m(veis треонгуляц) то вызывает cut\_rectangle(передает с опци –r18/30/43/36 ,список станций и данных,True)

+--- ----+--ymax

| |

+--- ----+--ymin

| |

xmin xmax

В функции cut\_rectangle проверяет список и проверяет sta\_list\_to\_degrees=False, а было переданно TRUE то slon = degrees(sta.lon) переводит в градусы иначе не переводит остави в радианах проверяет станции по координатам больше минимума периметра и меньше максимума то записывает в новый new\_sta\_lst и его возвращает те станции входящие в периметр

дале вызывает дебаг vprint и смотрит количество станций если меньше 3 вываливается с ошибкой

4)следуя по main Если rigion и метод не veis и нету опции –с вызывает vprint(Фильтрация станций на основе их удаленности от барицентра региона) из опции –r создает среднее по mean\_lon, mean\_lat( в радианах) создает объект bc = Station(lon=mean\_lon, lat=mean\_lat)передав значен

Также создает обект endpt = Station(lon=radians(lonmax), lat=radians(latmax)) с максимальными значениями те границами в радианах

(При вызове объекта Station ждет аргументы в виде \*args, \*\*kargs именнованых и не именнованых списков, вызывается метод set\_none() в нем присваевается всему self.name = None

self.lon = None.lat = None self.ve = None, self.vn = None, self.se = None, self.sn = None, self.rho = None, self.t = None,

если в конструкторе список не роавен 0 вызов init\_from\_ascii\_line

# если именнованный словарик не равен 0 если ключ в station\_member\_names = ['name', 'lat', 'lon', 've', 'vn', 'se', 'sn', 'rho', устанавливает атрибуты ключ и значение

Метод init\_from\_ascii\_line(self, input\_line): перевод градусы в радианы

)Если в Station передать только координаты то вернет обеъкт 'name': None, 'lon': 0.4625122517784973, 'lat': 0.619591884457987, 've': None, 'vn': None, 'se': None, 'sn': None, 'rho': Non

e, 't': None

Далее объект endpt вызывает метод haversine\_distance(передав объект bc со средними lon lat в радианах) haversine\_distance(принимает два параметра sta, R=6372797.560856e0 ) и возвращает объект пред-этим вызваф функцию ArcInRadians(self, sta) в ней в итоге Вернет

R\_zemli\*2\*arcsin(sqrt( sin(LATmax -LATmean)^2 + (cos(max lat)\* cos(mean Lat)) \* sin(LONmax -LONmean)^2 )) это число в модуль и сдвигает точку на два получив в итоге к примеру 720. 77558654534 Определить расстояние по дуге большого круга между двумя точками на сфере учитывая их долготу и широту (self.lat, self.lon в радианах) между средним lon lat и максимальными lon lat из периметра

Дале так как опция –d не указанна а указанна –dmax то –dmax по умолчанию по гаусу то еще умножает\*2\*2.15 и прибавляем к дуге на сфере было 720 стало 2870

Дале из списка данных делаем также только не мax – meea по lon lat из периметра, а данные из списка т.е lon lat по каждой станции – среднее(lon lat) из периметра и сравниваем с предыдущим результатом числом d если меньше или равно записываем в список(только дуги данных\*1/e3 те 0.0lon lat) это растояние в километрах все что попадает перезаписывается в список

Mean\_lon = Дальше средние по lon в градусах

lcm = radians(floor(mean\_lon)) приближает к целому числу округляя вниз

sta\_list\_utm = deepcopy(sta\_list\_ell) # копирование объекта

N, E, Zone, lcm = ell2utm(sta.lat, sta.lon, Ellipsoid("wgs84"), lcm) перевод в кординаты в систему UTM

sta\_list\_utm[idx].lon = E перезаписывает новые координаты X Y в замен градусов

sta\_list\_utm[idx].lat = N перезаписывает новые координаты X Y в замен градусов

далее запись заголовков в выход фаил

5) если была опция –b передает нащ объект с данными и аргументы в клас тензор

Но по умолчанию нет опции по этому пропускаем

6)попадаем если метод ==shein далее если указывали регион

A) строим сетку вызвав клас Grid передав lonmin-max latmin-max станций и min– max и шаги по X и Y регион квадрат из опций

если региона небыло

B) то класу передаются данные не превышающие число d и шаги по x y

A)в конструкторе (x,y max)(x,y min)(x,y шаг) strict\_upper\_limit=False, upper\_limit\_epsilon=1e10

X,y pts = int(floor((x\_max-x\_min) / float(x\_step))) так же для y

Далее print('strict\_upper\_limit ==False) идут проверки xy\_step > 5e-11

self.xpts > 0 and self.ypts > 0

x\_min + self.xpts \* x\_step <= x\_max + upper\_limit\_epsilon

assert y\_min + self.ypts \* y\_step <= y\_max + upper\_limit\_epsilon

возвращает обеъект с координатами шагом в 0.5 ну или от тех параметров еоторые указанны сетке

(24.5, 34.5)

(25.5, 34.5)

(26.5, 34.5)

(27.5, 34.5)

(28.5, 34.5)

(24.5, 35.5)

(25.5, 35.5)

(26.5, 35.5)

(27.5, 35.5)

(28.5, 35.5)

(24.5, 36.5)

(25.5, 36.5)

(26.5, 36.5)

(27.5, 36.5)

(28.5, 36.5)

**7)** Далее проверка мультипроцесса если нет вызывает функцию compute\_\_(grd, sta\_list\_utm, lcm, fout, fstats, vprint, \*\*dargs) сетка, данные, средние lon градусах, фаил для записи, фаил статистики для записи, отладочные сообщения, аргументы

В теле функции переменные node\_nr, nodes\_estim = 0, 0 ?

Сетку пилит на clat, clon = radians(y), radians(x) затем перводит в XY

N, E, ZN, lcm = ell2utm(clat, clon, Ellipsoid("wgs84"), utm\_lcm) теперь данные и сетка в одних системах координат UTM

далее вызов объекта Strain(N, E, clat<0e0, данные, аргументы )возвращает

'\_\_xcmp\_\_': 997001.1463937794,

'\_\_ycmp\_\_': 4667222.742759417,

'\_\_in\_shemisphere\_\_': False,

'\_\_zweights\_\_': None,

'\_\_lweights\_\_': None,

'\_\_options\_\_': {'ltype': 'gaussian', 'Wt': 24, 'dmin': 1, 'dmax': 500, 'dstep': 1, 'd\_coef': None, 'cutoff\_dis': 2.15, 'weighting\_function': 'shen','verbose\_mode': False},

'\_\_parameters\_\_': {'Ux': 0.0, 'Uy': 0.0, 'omega': 0.0, 'taux': 0.0, 'tauxy': 0.0, 'tauy': 0.0},

'\_\_vcv\_\_': None,

'vprint': <function ShenStrain.\_\_init\_\_.<locals>.<lambda> at 0x000000000A2AA488>}

Принимает Класс объекта= список сетку, список станций столько раз сколько в сетке строк ,параметры

Далее вызов метода beta\_angles() в нем вызов метода azimouths = azimouths(self) это список углов между данными и сеткой

ΔX = sta.lon-self.\_\_xcmp\_\_ для одной марки к каждой точке сетки

ΔY = sta.lat-self.\_\_ycmp\_\_ для одной марки к каждой точке сетки

и az = арктангенс в радианах(ΔX, ΔY), normalized to [0, 2π)

причем из с марки значение по X вычитается с каждым значением X из сетки, также по Y чем больше шаг в опции тем сетка имеет меньший шаг и больше значений

далее сортируется список углов в обратном порядке не по значению из них строится список где где 3.14 \*2 + (миним-максим угол)

#### ### ### '2\*3.14 + ugolmin-ugol max', И в for 2-1 -3-2 4-3 итд далее так и запихивает в список потом дописывает углы между 2-1 3-2 4-3 итд

#'2\*3.14 + ugolmin-ugol max 2-1 3-2 4-3 …

#[5.98062956108725, 0.16228832611652677, 0.04927039350273532, 0.09099702647307395]

#[1.440519003517112, 0.0034828520547216524, 0.0011259026626598834, 4.838057548945093]

Возвращает тако масив

Далее идет проверка по каждой ситроке списка это и каждый объект максимальный угол из масива в градусах <= по умолчанию 180 град если попадает то вызывает метод estimate()

Расчет estimate()

В объекте уже присутствуют некоторые данные

**'\_\_xcmp\_\_': 432190.9621175188, '\_\_ycmp\_\_': 3956479.681554048, '\_\_in\_shemisphere\_\_': False, '\_\_zweights\_\_': None, '\_\_lweights\_\_': None,**

**'\_\_options\_\_': {'ltype': 'gaussian', 'Wt': 24, 'dmin': 1, 'dmax': 500, 'dstep': 1,'d\_coef':None,'cutoff\_dis': 2.15, 'weig**

**hting\_function': 'shen',**

**'verbose\_mode': False},**

**'\_\_parameters\_\_': {'Ux': 0.0, 'Uy': 0.0, 'omega': 0.0, 'taux': 0.0, 'tauxy':0.0, 'tauy': 0.0}, '\_\_vcv\_\_': None**

**Поиск ВЕСОВ**

Дале проверка метод шейна да -> если нет options\_\_['d\_coef'] то вызов метода по поиску

Перед этим проверив если dmin >= dmax шаг <0

Вызов метода lwghts, zwghts, d = self.find\_optimal\_d()

Создает масив нумпай = d от 0 до шаг 499 задается в опциях

Допустим миним 1 максимум 20 шаг 1 то каждое значение из масива умножается на коэф 2.15 d = [1\*2.15 2\*2.15 3\*2.15 … 20\*2.15]

Также по количеству шагов создается список где (X сетки – X марки)^2+ (Yсетки –Y марки)^2 проверка меньше или равен чем коэф d \*2.15 и т д и если в этом новом списке количество марок больше 3 то

Далее создает новый список и присваевает результат метода new\_sta\_lst = self.filter\_sta\_wrt\_distance(d)

В методе filter\_sta\_wrt\_distance(d=None)

Создает объект сс = Station передает узлы сетки которые прошли проверку по углу BETA

Тут вызывает из Station метод squared\_distance\_from(передает сс т. е. узлы сетки ) в методе

dlon = (сетка(lon) – pol2(lon))/1e3 координата сетки по X – координато по X каждой станции

dlat = (сетка(lat) – pol2(lat))/1e3 координата сетки по Y – координато по Y каждой станции

вернет (dlat\*dlat + dlon\*dlon) делить но 1e3 чтобы квадрат расстояния не был слишком большим

далее создает список если эвклидовое растояние меньше 2.15\*step

т е шаг 10 узел прошел всего 1 то проверка (координаты узла – каждой марки)^2<=2.15\*шаг 1

затем на шаг 2 итд 4.3 6.4 8.6 10.7 12.8 15.0 17.2 19.3

затем если в новом списке количество таких решений больше 3 то

l\_weights(передав новый список ):

то вызов l\_weights() и z\_weights() передва новый списчик

9) вес l\_ weights Вызов метода lwghts, zwghts, d = self.find\_optimal\_d()

Создает масив нумпай = d от 0 до шаг пример 499 добавляет этот список к каждому объекту

Следом вызывает метод self.filter\_sta\_wrt\_distance(d) передав масив numpy с шагом от до

В методе вызов объекта Station передав сетку из d создается лимит d = [1\*2.15 2\*2.15 3\*2.15 … 20\*2.15]

И вызывая еще один метод squared\_distance\_from(cc) передав сетку проверяет квадратное расстояние

т е (X сетки – X марки)^2+ (Yсетки –Y марки)^2 меньше или равен чем коэф d \*2.15 потом d=2\*2.15 и т д и если в этом новом списке количество марок больше 3 вызов метода l\_weights(new\_sta\_lst) и z\_weights(new\_sta\_lst)с объектами прошедшую проверку по лимиту метод l\_weights вернет в зависимости от опций либо  
или

Тогда как z\_weights = n\*θ(i)/4π в методе вызывается метод ompute\_theta\_angles(объект список станций с сеткой) там θ = β1 + β2 для каждой точки i угол тета разница азимутов предыдущего (i-1) минус следующий точка (i+1). Все углы θ будут находиться в диапазоне (0, 2π).

Как с бэта берется азимут между ΔX = sta.lon-self.\_\_xcmp\_\_ и ΔY = sta.lat-self.\_\_ycmp\_\_ для одной марки к каждой точке сетки

и az = арктангенс в радианах(ΔX, ΔY), normalized to [0, 2π)

возвращает (0.5e0\*degrees(θ)) + (0.25\*360/nθ) ) \* nθ / (( 1+0.25)\*360e0 )

Дале сравниваем количекство веса l и z

ВЕС W = Σ{L(i)\*Z(i)} сумма помноженных весов l на Z

Дальше проверка если вес W больше или равен опции wt по умолчанию 24 то вернет веса lwghts, zwghts, и масив шагов d

Далее добавляем веса к объекту self.\_\_zweights\_\_ = zwghts

self.\_\_lweights\_\_ = lwghts

**Матрицы для расчета тензора деформаций**

вызов метода A, b = self.ls\_matrices()

матрица весов создается в методе make\_weight\_matrix() N на 1

тут построит квадратный корень из матрицы весов W <- P^(1/2) Каждый элемент в

возвращаемая матрица, будет весом для соответствующей станции

долгота и широта компонента. Например, если \_\_stalst\_\_ = ['dyng', 'ankr', ...],

тогда W[0] — x-вес dyng, W[1] — y-вес dyng,

W[2] — x-вес ankr, W[3] — y-вес ankr ....

вернет

w\_x = (1/σe) \* sqrt(Z(i)\*L(i))

w\_y = (1/σn) \* sqrt(Z(i)\*L(i))

w\_x = (1/σe) \* sqrt(Z(i)\*L(i))

w\_y = (1/σn) \* sqrt(Z(i)\*L(i))

…

Создается матрица N на 6

С объекта и метода distance\_from() вытаскиваются :

dx, dy, dr = xyr[idx] # X-Xgrid, Y-Ygrid,

Заполняется матрица А\*весовую матрицу W

**A[i] = [ Wx\*j for j in [1e0, 0e0, dx, dy, 0e0, dy] ]**

**A[i+1] = [ Wy\*j for j in [0e0, 1e0, 0e0, dx, dy, -dx] ]**

И матрица B умноженная на весовую матрицу W

**b[i] = sta.ve \* Wx**

**b[i+1] = sta.vn \* Wy** где скорости умножаются на вес

две матрицы возвращаются

Если мы решим A\b, мы получим вектор параметров:

[ Ux, Uy, τx, τxy, τy, ω ] линейная регрессия в помощь

Порядок заполнения азимута марка X Y – 1 точка сетка=и X Y, вторая марка X Y – 1 точка сетки, 1 марка X Y – 2 точка сетки X Y

filter\_sta\_wrt\_distance nlst = [ s for s in self.\_\_stalst\_\_ if s.squared\_distance\_from(cc) <= limit\*limit ]

Метод триангуляции

1. Вызывается метод parse\_ascii\_input() получаем список входного файла с марками и корди скорост ошибками в sta\_list\_ell

Так как в опциях регион то количество марок те строк во входном файле те уже списка если мене 3 то ошибка

2)В numpy запихиваем список sta\_list\_ell и вызываем объект Делоне засовывая в класс массив

tri = Delaunay(массив) в for разворачиваем объект там треугольники в виде [1 3 0] [1 0 2]

Далее sta\_list\_utm # копирование объекта sta\_list\_ell) затем преобразование координат в EN 0

N, E, Zone, lcm = ell2utm(sta.lat, sta.lon, Ellipsoid("wgs84"), lcm)

sta\_list\_utm[idx].lon = E

sta\_list\_utm[idx].lat = N

3)Создание центра треугольника: trng[0..3 это элементы треугольника

cx = (sta\_list\_utm[trng[0]].lon + sta\_list\_utm[trng[1]].lon + sta\_list\_utm[trng[2]].lon)/3e0

cy = (sta\_list\_utm[trng[0]].lat + sta\_list\_utm[trng[1]].lat + sta\_list\_utm[trng[2]].lat)/3e0

4)в for:

вызов объекта sstr= передав в конструктор ShenStrain(cx,cy,false, вершины треугольника, equal\_weights)

вызов метода sstr =estimate()

те деформация расчитывет для каждого треугол отдельно?

В методе estimate() создает матрису в функции matricec(cx cy)

Создает вес умножив сигмц=1 на матрице весов из 1-ц[1,1,1…от кол входн данных]

Возвращает массив из единиц

Дале создает объект station передав бари центр треугольника там из кординат входных пунктов вычитает сентр возвращает три значения

∆lon = site.lon-bc.lon ∆lat = site.lat-bc.lat и ∆r =

5) Создается матрица из 0 6\*6 И 1\*6 из 0

A B

[[0. 0. 0. 0. 0. 0.] [[0.]

[0. 0. 0. 0. 0. 0.] [0.]

[0. 0. 0. 0. 0. 0.] [0.]

[0. 0. 0. 0. 0. 0.] [0.]

[0. 0. 0. 0. 0. 0.] [0.]

[0. 0. 0. 0. 0. 0.]] [0.]]

В for по sta\_lst вход данных веса Wx = W[i] первое значение из матриц весов

Wy = W[i+1] второе значение из матриц весов

Затем для каждого центра угла два значения матриц

A[перв знач] = Wx \* [ 1, 0, dx, dy, 0, dy)

A[втор знач] = Wy \* [0, 1, 0, dx, dy, -dx]

b[перв знач] = sta.ve \* Wx

b[втор знач] = sta.vn \* Wy

и i += 2 то есть значения в for увеличеваем на 2 для след марки и центра те на 3 марки 1 центр треугольника

и возвращает матрицы A и B

Дале VcV = numpy.dot(A.T, A) умножение матрицы Aтранспонир на A те скаляр произвед

Дале смотрит размер если 6\*6 то это вычисление по триангуляции для каждого бари центра три марки если матрица больше 6\*6 то ето алгоритм по узлам если меньше то ошибка

6) Дале расстояние" между векторами Ax = b может быть минимизировано. Поиск коэф x в матричнойс системе линейных уравнений по методу наименьших квадратов

Если матрица a является квадратной как в треангуляционном алгоритме и обладает полным рангом, то решение x (до ошибки округления) является точным решением системы. В противном случае, если количество линейно независимых строк матрицы a меньше, равно или больше числа линейно независимых столбцов, то x не будет "точным" решением.

estim, res, rank, sing\_vals = numpy.linalg.lstsq(A, b) поиск коэф

estim ' =Ux, Uy, τxч, τxу, τyy, ω

Сумма квадратичная остатков евклидовая 2 норма для каждого столбца ' = res

Ранг матрицы A ', = rank

Сингулярное значение матрицы A ' = sing\_vals

Дальше вызом метода print\_details()

Там перевод центр треугольника в градусы и

emean, ediff, taumax, staumax, emax, semax, emin, semin, azim, sazim, \

dilat, sdilat, sec\_inv, ssec\_inv = self.cmp\_strain(self.\_\_vcv\_\_)

а так как в тревнг алгоритме .\_\_vcv\_\_ == None то добавляет к каждому X \*1e3

а в методе cmp\_strain

x1 = self.\_\_parameters\_\_['taux'] ## strain/yr

x2 = self.\_\_parameters\_\_['tauxy'] ## strain/yr

x3 = self.\_\_parameters\_\_['tauy'] ## strain/yr

cov = pi / 180e0

## estimate principle strain rates emax, emin, maximum shear tau\_max,

##+ and dextral tau\_max azimuth

emean = (x1+x3) / 2e0 ## strain/yr

ediff = (x1-x3) / 2e0 ## strain/yr

taumax= sqrt(x2\*\*2 + ediff\*\*2) ## strain/yr

#print(x1,x2,x3)

#print(x1\*1e9,x2\*1e9,x3\*1e9)

#time.sleep(1)

emax = emean+taumax ## strain/yr

emin = emean-taumax ## strain/yr

azim = -atan2(x2, ediff) / cov / 2.0e0 ## degrees arctan(y,x)

azim = 90e0+azim

dexazim = azim+45e0-180e0

dilat = x1+x3 ## strain/yr

sec\_inv = sqrt(x1\*x1+2e0\*x2\*x2+x3\*x3)

и запись в списчик