```
#------------------------------------------------------------------------------
#
# Signed Distance Function Calculator
# **********************************
#
# This SGeMS plugin calculates the anisotropic signed distances for each data
# point and each rock type
#
# AUTHOR: Roberto Mentzingen Rolo
#
#------------------------------------------------------------------------------

#!/bin/python
import sgems
import math
import numpy as np

#Calculates the distances
def dist(x1, y1, z1, x2, y2, z2):
    return math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2)

#Defines the rotation and dilatation matrices
def rot(range1, range2, range3, azimuth, dip, rake, vetor):

    if azimuth >= 0 and azimuth <=270:
    alpha = math.radians(90-azimuth)
    else:
    alpha = math.radians(450-azimuth)
    beta = -math.radians(dip)
    phi = math.radians(rake)

    rot_matrix = np.zeros((3,3))

    rot_matrix[0,0] = math.cos(beta)*math.cos(alpha)
    rot_matrix[0,1] = math.cos(beta)*math.sin(alpha)
    rot_matrix[0,2] = -math.sin(beta)
    rot_matrix[1,0] = (range1/range2)*(-
math.cos(phi)*math.sin(alpha)+math.sin(phi)*math.sin(beta)*math.cos(alpha))
    rot_matrix[1,1] = (range1/range2)*(math.cos(phi)*math.cos(alpha)+math.sin(phi)*math.si
n(beta)*math.sin(alpha))
    rot_matrix[1,2] = (range1/range2)*(math.sin(phi)*math.cos(beta))
    rot_matrix[2,0] = (range1/range3)*(math.sin(phi)*math.sin(alpha)+math.cos(phi)*math.si
n(beta)*math.cos(alpha))
    rot_matrix[2,1] = (range1/range3)*(-
math.sin(phi)*math.cos(alpha)+math.cos(phi)*math.sin(beta)*math.sin(alpha))
    rot_matrix[2,2] = (range1/range3)*(math.cos(phi)*math.cos(beta))

    vetor = np.array(vetor)

    return np.dot(rot_matrix, vetor)

#Transform the data with the ratation/dilatation matrices
def anis_search(X, Y, Z, range1, range2, range3, azimuth, dip, rake):

    X_linha = []
    Y_linha = []
    Z_linha = []

    for i in range(len(X)):
    vet = [X[i],Y[i],Z[i]]

    vet_rot = rot(range1, range2, range3, azimuth, dip, rake, vet)

    X_linha.append(vet_rot[0])
    Y_linha.append(vet_rot[1])
    Z_linha.append(vet_rot[2])
```

```
63.
64.     return X_linha, Y_linha, Z_linha
65.
66. #Shows every parameter of the plugin in the command pannel
67. def read_params(a,j=''):
68.   for i in a:
69.     if (type(a[i])!=type({'a':1})):
70.       print j+"['"+str(i)+"']="+str(a[i])
71.     else:
72.       read_params(a[i],j+"['"+str(i)+"']")
73.
74. class signed_distances:
75.     def __init__(self):
76.         pass
77.
78.     def initialize(self, params):
79.         self.params = params
80.         return True
81.
82.     def execute(self):
83.
84.         '''''#Execute the funtion read_params
85.         read_params(self.params)
86.         print self.params'''
87.
88.         #Get the grid and rock type propery
89.         grid = self.params['propertyselectornoregion']['grid']
90.         prop = self.params['propertyselectornoregion']['property']
91.
92.         #Error message
93.         if len(grid) == 0 or len(prop) == 0:
94.             print 'Select the rocktype property'
95.             return False
96.
97.         #Get the X, Y and Z coordinates and RT property
98.         X = sgems.get_property(grid, '_X_')
99.         Y = sgems.get_property(grid, '_Y_')
100.             Z = sgems.get_property(grid, '_Z_')
101.             RT = sgems.get_property(grid, prop)
102.
103.             elipsoide = self.params['ellipsoidinput']['value']
104.             elipsoide_split = elipsoide.split()
105.
106.             range1 = float(elipsoide_split[0])
107.             range2 = float(elipsoide_split[1])
108.             range3 = float(elipsoide_split[2])
109.
110.             azimuth = float(elipsoide_split[3])
111.             dip = float(elipsoide_split[4])
112.             rake = float(elipsoide_split[5])
113.
114.             X, Y, Z = anis_search(X, Y, Z, range1, range2, range3, azimuth, dip, rake)
115.
116.             #Creates a list of all rock types
117.             rt_list = []
118.             for i in RT:
119.                 if i not in rt_list and not math.isnan(i):
120.                     rt_list.append(i)
121.
122.             #Sort the rock type list in crescent order
123.             rt_list = [int(x) for x in rt_list]
124.             rt_list.sort()
125.
126.             #Create a empty distance matrix
127.             dist_matrix = np.zeros(shape = ((len(rt_list)), (len(RT))))
128.
129.             #Calculates the signed distances, and append it in the distance matrix
130.             for i in range(len(rt_list)):
```

```
                    rock = rt_list[i]

                    for j in range(len(RT)):

                        if math.isnan(RT[j]):
                            dist_matrix[i][j] = float('nan')

                        elif RT[j] == rock:
                            dsmin = 1.0e21

                            for k in range(len(RT)):

                                if RT[j] != RT[k] and not math.isnan(RT[k]):
                                    if (dist(X[j], Y[j], Z[j], X[k], Y[k], Z[k])) < dsmin:

                                        dsmin = (dist(X[j], Y[j], Z[j], X[k], Y[k], Z[k]))


                                dist_matrix[i][j] = -dsmin

                        else:
                            dsmin = 1.0e21

                            for k in range(len(RT)):

                                if RT[k] == rock:
                                    if (dist(X[j], Y[j], Z[j], X[k], Y[k], Z[k])) < dsmin:

                                        dsmin = (dist(X[j], Y[j], Z[j], X[k], Y[k], Z[k]))


                                dist_matrix[i][j] = dsmin

                #Creates the signed distances properties
                lst_props_grid=sgems.get_property_list(grid)

                for k in range(len(dist_matrix)):
                    prop_final_data_name = 'Signed_Distances_RT_' + str(rt_list[k])

                    if (prop_final_data_name in lst_props_grid):
                        flag=0
                        i=1
                        while (flag==0):
                            test_name=prop_final_data_name+'-'+str(i)
                            if (test_name not in lst_props_grid):
                                flag=1
                                prop_final_data_name=test_name
                            i=i+1

                    list = dist_matrix[k].tolist()
                    sgems.set_property(grid, prop_final_data_name, list)

                return True

            def finalize(self):
                return True

            def name(self):
                return "signed_distances"

        ##########################################################################
        def get_plugins():
            return ["signed_distances"]
```