**Pseudocode**

```
BEGIN
    Import necessary libraries:
        numpy, matplotlib, netCDF4, pandas, math, shapely,
        matplotlib.tri, pickle, time
    #Record the start time for measuring execution duration
    Open the netCDF file of the low-resolution mesh file and
    extract variables:
        mesh2d_node_x, mesh2d_node_y, mesh2d_node_z
    Define a list of mesh points with x, y, and z coordinates
    Sort the mesh points based on x-coordinate (ascending order)
    Defining cross sections out of mesh points:
    #Set resolution threshold for cross-section separation
    Loop through mesh points to find the cross-sections:
    For each point:
            Calculate difference in x-coordinate from the
            previous point
            If the difference exceeds the resolution threshold:
                Increment cross-section counter
                Update starting point for next cross-section
    sort each cross section based on y-coordinate (descending
    order)
    Define parameters for mesh resolution and water level steps.
    Loop through cross sections:
          Loop through each cross sections:
               Determine list of different parameters related
               to each cross section:
               the starting and ending x-coordinates for the main
               channel, the highest y-coordinate in entire cross
               section and lowest y-coordinate inside the main
               channel
               Calculate the number of water level steps based
               on highest-y and lowest-y and defined water level
               step.
               Loop through each Water levels:
                    i. Calculate water level and water depth for
                    each step.
                   ii. Calculate the right-side width and the
                    left side width for the current water level.
                   a. iii. Calculate the width for the current
                      water level.


    #The water depth vs the AR product(Discharge is related to
    Area* hydraulic radius ^(2/3))
    Define a water level resolution to calculate the hydraulic
    radius.
    Loop through cross sections:
          Loop through each cross sections:
                Loop through each Water levels:
                       Store depth, water level, and area,
                       wetted perimeter calculations for each
                       sorted cross section.
```

34. Calculate the hydraulic radius (AR) for the current water depth.

35. **Determining** a list of polygon points for each pair of cross sections.

36. **Loop** through cross sections:

37. making lists of X,Y coordinates of the current cross section (r),and the next cross section (r + 1), (Add the first point of the current cross section (r) to close the polygon)

38. Zip the lists of x and y coordinates to create a list of tuple points representing the polygon.

39. **Open** the bed level file with x, y and z coordinates

40. Create a list of tuples representing the points (x, y, z) and sort the list based on x coordinates.

41. **Loop** through each list of tuple points representing the polygon :

42. **Define** the polygon using Shapely.

43. **Loop** through bed level points:

44. **If** the point is inside or touches the polygon, add it to the list of points

45. inside the polygon.

46. **Calculate** and store the volumes of mesh and bed levels and the differences between them.

47. **Loop** through cross sections (from the second one):

48. **Loop** through each Water levels:

49. **Calculate** the depth for the current water level.

50. **Find** the closest depth in the list of depth AR product list calculated in step 30.

51. with the same AR product, find the water level in next cross section.

52. **Define** a polygon between the current water level and the previous.

53. **Perform** a triangulation on the list of the mesh points.

54. **Calculate** the volume of the current triangle and add it to the total volume.

55. **Save** the total mesh volume for this specific water level.

56. **Find** the points inside the water level polygon.

57. **Create** a list of points inside the water level polygon and the points of polygon itself.

58. **Perform** a triangulation on the list of combined points.

59. **Calculate** the volume of the current triangle and add it to the total volume.

60. **Save** the total bed level volume for this specific water level.

61. **Append** the volume of mesh, bed levels, and

the difference between them to the respective lists.

62. #Determining the mesh points which could be moved vertically (ACTIVE_POINTS) for each water level:

63. **Loop** through cross sections (starting from the second):

64. **Loop** through water levels:

65. **Save** the index of mesh points available to move (mesh points between the current water level and the previous one.) (Active_points)

66. **Save** the number of active points in a list (CN_All)

67. **Calculate** the difference in volume for mesh points and bed level points(Dif)

68. **Loop** through CN_All list:

69. **Define** a total volume difference equal to the related [Dif](in case there are some water levels with no active points.)

70. **If** the next value is greater than zero divide the total volume difference by the number of the active points and width of the mesh cells to find the offsetting value and save the affset values [OFFSET_V].

71. **If** the **next** value of CN_All is zero, go to the next CN_All, and add the volume difference to the Total volume difference.

In summary, the code initializes OFFSET_V for the active points related to water levels. If there are no active points for some water levels, the active points from lower water levels will compensate for the volume differences of the upper water levels with no active points. The code also creates a new list of lists, Y_CROSS_NEW, which contains the elevations of mesh nodes detracted by related offset points.