**replicate_arrays( a, y, n)**

Replicate the array n times and return an array which shape is (length_ary * n)

Example:

a_lat_b = [4, 5]

result = replicate_array(a_lat_b,3) -> result = [4, 5, 4, 5, 4, 5]

Input parameters:

- double pa[] :  array to be replicated
- int length_a: length of array
- int n       : num replications

Output parameters:

- double py[]   : replicated array of length n_times * length_array

**calc_distance( a_x_u, a_y_u, b_x, b_y, dist)**

Given the coordinates of n_users and n_beams. Calculate the euclidean distance between all the users to all beams

Input parameters:

- a_x_u, a_y_u (float arrays of length U)
- b_x, b_y (float arrays of length B)
- U, B - integers, calculated in the python wrapper

Output parameters

- distances - array float of size U*B of pairwise distances arranged as
- [d_u1b1, d_u1b2,    d_u2b1, d_u2b2,    d_u3b1, d_u3b2].

**centroid_aux( a_demand, ov_allocation_u_b, a_coordinate_u, coo_b)**

Function centroids. Similar to combineArrays def centroid_aux( a_demand, ov_allocation_u_b ,  a_coordinate_u) returns array coo_b

Input parameters:

- demand: doubles length U
- allocation double or int length U*B
- coordinate double length U

Output parameter:

- coo_b : doubles, length B