Inverse distance weights

This exercise is about inverse distance weights.

Task 1

Given the data shown in table 1:

x	y	value
410	553	83
-475	923	72.5
-350	-522	72.2
705	-87	79.6
575	-1027	63.2
230	-1802	83.1
-500	-2002	72.1

Calculate the inverse distance weighted mean of the following locations by solving the calculations *by hand*:

x	y	value
210	50	
-455	-1500	
970	2300	

Task 2

You should be familiar with the inverse distance calculations by now. There are two MATLAB/Octave functions in this folder: `inverse_distance_fixed_radius.m` and `inverse_distance_fixed_neighbors.m` . They calculate the inverse distance mean for one given point (`[x, y]`). In both functions there is a part that looks like:

```
function v = inverse_distance_fixed_radius(xi, yi, vi, x, y, r)

  [...]

  % REPLACE FROM HERE
  v = mean(vi)
  % UNTIL HERE
end
```

In this version, both functions will just return the mean of all observations as the inverse distance weighted mean. Replace this code, to return the actual value.

a) fixed radius

The function `inverse_distance_fixed_radius.m` should only use observations from `vi` which's coordinates (`[xi, yi]`) are within the given radius `r` around the desired point of interest (`[x, y]`).

a) fixed neighbors

The function `inverse_distance_fixed_neighbors.m` should only use the `n` observations from `vi` which's coordinates (`[xi, yi]`) are closest to the desired point of interest (`[x, y]`).

Task 3

Run the code `test_single.m` against your code versions. Understand what the script is doing and be sure that it does not raise any errors anymore. These are so called *unit tests*. That means it will raise errors until your code is producing the correct values. This should help you with Task 2.

Task 4

Finish the function `inverse_distance.m` . This function can be used to apply either function to a regular grid of given `gridsize` . The method can be toggled by setting `method` to either 1 (fixed radius) or 2 (fixed neighbor), while `arg` is then either the radius `r` or the number of neighbors `n` .

Replace the two passages marked as:

```
function vgrid = inverse_distance(xi, yi, vi, gridsize, method, arg)
```

```
    [...]

    % REPLACE FROM HERE
    % apply the correct function to all cells
    vgrid = ones(size(vgrid)) * mean(vi);
    % TO HERE

    [...]
  end
```

You can solve this part by looping through all elements of the result grid `vgrid` .

Task 5

Use your code from task 4 to genereate a IDW interpolation from the given precipiation observations. Make a decision for the used method and parameters and describe this decision (in class).