Kriging

In this last exercise we will focus on Kriging interpolation. In the kriging folder, you will find a `kriging.m` function. It basically works just like the `inverse_distance.m` function of the first exercise. Only a few function arguments have changed. The `kriging.m` function will create the interpolation grid and apply another function, `ordinary_krige.m` to each of the grid cells, just like the IDW implementation. This way, it is possible to call the different IDW variants and the kriging from within the same function.

Note: This exercise is extensive. Solve tasks 1 - 5 and choose **one** of the remaining tasks.

Task 1

Apply the `kriging.m` routine to the observations in `artificial.csv` . To speed up the calculations, set the `max_p` parameter to 10. Use a gridsize of 1 (meter) and use your best guess for a spherical model from the last exercise.

Task 2

Apply the `inverse_distance.m` function from the first exercise, choose comparable parameters and plot both examples next to each other.

- Describe the differences between the results.
- Was it worth the effort? Does Kriging produce the better results here?

Task 3

In task 1 it was stated that limiting the `max_p` parameter will substantially speed up the processing time. Why?

Task 4

In the last exercise you compared two variogram models fitted manually and by a least squares optimization. You had to describe who was more successful, you or your computer. Now take the best parameter sets you and your computer found to fit a spherical variogram and interpolate a result (you can use the result from Task 4). Can you see differences in the result? Describe them if any.

Task 5

In the code of `ordinary_krige.m` you will find the code:

```
% PLACE YOUR ARGUMENT BASED SOLUTION HERE
% get the distances and the index of points to be used
d_in = sorted_d(1:max_p);
idx_in = idx(1:max_p);

% find the neighbors
x_in = xi(idx_in);
y_in = yi(idx_in);
z_in = zi(idx_in);
n = length(idx_in);
```

```
    % if not enough points are found, return NaN
    if n < 4
      z = NaN
      return
    end
    % TO HERE
```

Task 5 a)

This check does not make much sense here. Why?

Task 5 b)

It was intended to use the range parameter as a search distance to find neighboring points for an interpolation (`x_in` , `y_in` , `z_in`). Then, the `max_p` parameter should limit the points to be used *inside* this radius and the check above should give `NaN` if there are not enough points found. Implement this solution.

Task 5 c)

Also introduce a `min_p` parameter that replaces the hard-coded 4 in the check.

Task 6

In the last exercise we fitted not only a spherical, but also an exponential and a gaussian variogram. Use these two models and the parameters found in the last exercise to perform kriging. Plot the results next to each other and describe the differences.

Task 7

Use any of the kriging interpolations you performed so far. Changing the `max_p` and `min_p` parameters can be utilized to speed up the calculations, while yielding essentially the same results. How small can both parameters get until the results are substantially different? Can you test this in a systematic way?

Task 8

Use any of the kriging interpolations you performed so far. How far can one decrease the sample size of `artificial.csv` until kriging does not work stable anymore? What about IDW? Can you base a meaningful result on less observations than needed for Kriging?

Task 9

If you look closely, our kriging routine is still missing something. We can use the `kriging.m` function to interpolate a regular grid. The function does not return the kriging variance. To implement this, you need to make `ordinary_krige.m` to return that as a second return value and then catch the value into another grid in `kriging.m`, that has the same shape as the interpolation grid. The function will return `[vgrid sgrid]` then.

Task 10

Note: This task has higher difficulty.

Use your best parameter set for the spherical model to interpolate the data in `artifical.csv` . Conduct a small sensitivity analysis for the variogram parameters. Test at least 10 different values for each variogram parameter, each within a range of possible values (each of the 10 values should still yield a useful variogram). Judge the performance either by the interpolation result (visual inspection), or make a leave-one-out cross validation.

That means you will run `ordinary_krige.m` for each observation in the sample. Omit each point from the kriging but use its coordinates as x and y parameter. Then the difference of the real z to the one yielded by the function can be interpreted as an *interpolation error*.