Delft University of Technology

Additional Thesis
AES4011-10

Can the use of different interpolation methods lead to improved performance of the HYPE model?

Authors:
Diewertje Dekker (4592190)

September - December 2021

A project done for TU Delft in cooperation with Lund University

1 Abstract

This additional thesis project is performed as preliminary research for a bigger project that they are going to start at Lund University, to investigate whether the use of a different interpolation methods, to link the precipitation data to the sub-basins centers of the HYPE model, lead to improved model performance. In this report, previously performed research is summarised and the limitations in researching this question with the HYPE model are described. A start is made with investigating this question, by answering the question whether different interpolation methods result in a different discharge when it is assumed that all fallen precipitation ends up as discharge. This is investigated for the PO basin in Italy with 4 interpolation methods: NN, IDW, BIL and OK. The effects of the interpolation methods on the computed discharge time series are analysed with the use of the correlation, RE, NSE and KGE. It is shown that for the PO basin, with a 1000 km^2 average sub-basin size and a gridded data set with a resolution of 50 km, the interpolation methods do not produce differences for which you would expect that it could lead to model improvement. However, based on findings from previous research, a next step is proposed, in which we can investigate if we do observe a difference at a different sub-basin scale or data resolution.

2 Introduction

HYPE, the Hydrological Predictions for the Environment (HYPE) model, was developed by the Swedish Meteorological and Hydrological Institute (SMHI). The model describes the flow of water from precipitation to the sea. It was build to provide water information with high spatial resolution for un-gauged locations and to be able to provide enough information of the water flow as input for models describing nutrient transport (Lindström et al., 2010). Previous models were not able to do this, since they did not describe water flow paths in the soil and the mixing of water in rivers and lakes. For HYPE this nutrient transport is even build in.

The HYPE model has been used in Sweden since 2008 and over the years many different versions of it appeared, specified for different regions on the globe. It is used for flood forecasting and local early warning systems (Massazza et al., 2020), estimating inflow of water and substances to oceanography models, soil-water forecasts for gardening companies and climate change impacts assessments for hydro power companies (Hundecha et al., 2016).

HYPE is a semi-distributed, physically based catchment model. It simulates the flow of both water and other nutrients from the moment it falls down as precipitation, till it ends up at the sea. Calculations in the model are made with a daily time step for all the coupled sub-basins in one catchment.

We call the model physically based, meaning that it is based on our understanding of the physics of the hydrological processes and uses physically based equations to describe these processes (Seth, 2016).

Secondly, we call the model semi-distributed, which means that we have divided the catchment in multiple equal sized sub-basins (See Figure 2 on the left).

Currently, HYPE uses precipitation and temperature as mandatory input data (optionally you can provide more input data like snowfall, radiation, wind speed, etc.), models the discharge and nutrient flow, and can give those and different performance assessments as output. For the World Wide HYPE (WW-HYPE), the input data is the GDF2.0 dataset, which is a raster-ed dataset with a resolution of 50 by 50 km. However, the model wants to know the temperature and precipitation at the center point of every sub-basin. Currently that is done by taking the NN. The idea that I make a start with in this additional thesis project, is the question weather linking the raster-ed data set to the sub-basin centers with other interpolation methods could lead to improved model results.

In the next section (section 3), some more detailed information on the HYPE model is provided. In section 4 we will explore previous research that is done on the effect of different interpolation methods to link the data to a hydrological model. In section 5, we will discuss the limitation of researching this question and define the exact question that I will explore in the remaining part of this report. In section 6, we will discuss the methods used, followed by the results is section 7 and a conclusion and discussion in section 8.

3 Background on HYPE

As mentioned, HYPE is a semi-distributed model. In the case of the World-Wide HYPE (WW-HYPE), the version I will be working with, these sub-basins have an average size of $1000km^2$. For these sub-basins, it is defined what percentage of its area consists of a certain soil-type, land-use type, vegetation-type, or consists of

lakes. All these different soil, landuse- and vegetation types have calibrated parameters that indicate how quickly water moves through. Based on these indications per catchment, the model simulates 5 different processes (Lindström et al., 2010):

1. Snow accumulation and melt: depends on temperature and precipitation data.
   (a) Snow falls if below temp threshold. Air temp depends on elevation of the class.
   (b) Snow melts: degree-day method which depends on land use, same temp threshold.
   (c) Storage and refreezing of melt water and rainfall is disregarded.
2. Water flow in the soil: Part of precipitation goes in soil, rest (when threshold of amount of water in soil is exceeded) is surface flow.
   (a) Depends on soil type. Soil has 3 layers. (See Figure 2 on the right)
   (b) Evapotranspiration occurs from 2 top soil layers.
   (c) Groundwater table is small in Swedish soils, so no separate compartment for groundwater in the model (is included in the soil water).
3. Rivers and lakes: There are 2 types: internal and main. Precipitation gets added to the lake (no snow or ice modelled) + evaporation at the potential rate till lake is dry.
   (a) River flow is delayed in time according to the river length and flood wave velocity. River lengths are approximated as square root of sub-basin area (or given as data input) + attenuation part of delay is realised through damping in a linear reservoir with a recession coefficient (chosen to give the required delay)
   (b) Outflow from all lakes determined by rating curve (same for all-in one model application) (unless regulated lake)
4. Soil temperature: is weighted sum of air temperature, soil temperature previous day and a constant deep soil temperature.
5. Nutrients: This part will not be discussed here, since I will not focus on this. For more information I refer to the paper of Lindström et al., 2010.

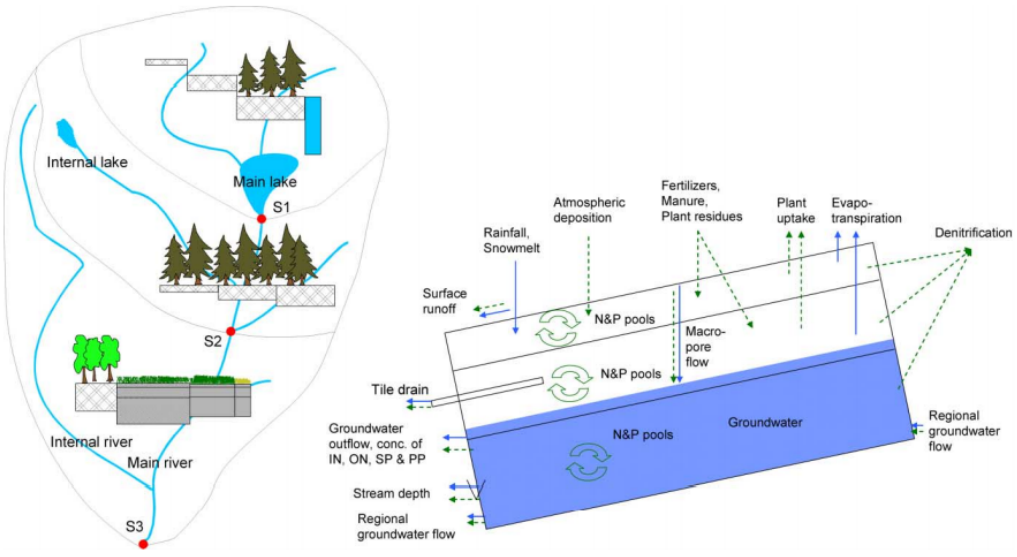


Figure 2: On the left side you can see the schematic division into sub-basins with an indication on vegetation types. On the right side you see a schematic model structure that simulates the three soil layers. Solid and dashed arrows show fluxes of water and elements, respectively. (Lindström et al., 2010)

Some parameters in the model were set using hydrological expertise and knowledge (soft data), but most were calibrated against the observations (discharge and nutrient measurements) in groups of 10 (Lindström et al., 2010). This calibration process is very extensive due to the large amount of parameters and is an ongoing process, since optimisation can only be done for small subsets of parameters and taking different subsets might lead to different results.

4 Previously performed research on the interpolation of precipitation data for semi-distributed hydrological modelling

Here I will present some of the research performed on the different interpolation techniques for precipitation data linked to hydrological modelling, and their most important conclusions. I will only focus on the research that is done with assessments of the interpolation techniques using a semi-distributed hydrological model, since this is a different criteria than assessing it by cross-validation on a grid. When using a hydrological model, we want to get the precipitation value for the sub-catchment, that best represents the volume of precipitation for that whole sub-catchment, when you multiply this value by the area. While when we do cross-validation, we just want the most accurate precipitation value for that exact location. Therefore we can see these as 2 different tasks.

For HYPE specifically, there is no research done yet regarding different interpolation techniques to link the precipitation data to the model. However, such research is performed using other semi-distributed hydrological models.

Cheng et al., 2017 found that all methods they tested performed better with heavy rain, but that Nearest Neighbour (NN) performs best for light rain. They accessed this by looking at the runoff output of the hydrological model SWAT and comparing different seasons to each other. They conclude that the interpolation methods should be chosen depending on the amount of rain.

But not only the amount of rain is important. Ruelland et al., 2008 show that interpolation methods that only produce the right volume of precipitation, sometimes perform less than methods that keep the spatial variability. They analysed this by predicting the runoff with a semi-distributed hydrological model (Hydrostrahler). They did re-calibration aimed at optimizing the NSE criterion. He concludes that the model is sensitive to both volumetric and spatial differences. Inverse Distance Weighting (IDW) gave the most realistic results.

Also according Segond et al., 2007, who where using the semi-distributed model ROBR, the spatial variability in rainfall does affect the hydrological model outputs. They mention that in previous research it was revealed that the spatial variability of rainfall is important at all sub-catchment scales and that its importance increases as the sub-catchments size decreases, since the damping of the catchment becomes less. They state that only when there is not enough variability in rainfall to overcome the damping effect of the catchment, detailed knowledge of the spatial variability would not be needed. They mention that Smith et al., 2004 found the guide lines that knowledge of the spatial variability was needed to model a 795 km^2 catchment, while an catchment average was enough for a 1645 km^2 catchment. This was concluded when using a data set with a resolution of 4 km. So for smaller catchments, the damping effect is less and therefore it is more sensitive to the spatial variability in rain.

This could be related to the conclusions of Haberlandt and Kite, 1998. They found that the improvements that can be obtained with different interpolation techniques are related to the size of the sub-basins used. They tested several interpolation techniques and accessed them with the hydrological model SLURP. They found that there was only an improvement possible (when using different interpolation techniques) for river gauges of which the drainage area was small enough ($< 100\ 000\ km^2$). Smaller sub-basins are more sensitive to improvement in precipitation estimates than larger ones, at least when modelling a macro-scale watershed with a coarse rain observations network. This could be related to the dampening effect that Segond et al., 2007 and Smith et al., 2004 talk about.

Schuurmans and Bierkens, 2007, who used a fully distributed model, and Chaubey et al., 1999, found that ignoring the spatial variability of rain can lead to errors in the model parameters to compensate for errors you would otherwise have in the output. So it does not necessarily affect your runoff, but it does affect the correctness of the parameters in your model. This argues for that re-calibration would be needed, every time you use a new precipitation data set.

Those papers can be summarised as that it is dependent on the catchment size combined with the grid-size of the data, whether the spatial variability should be taken into account, or that an average of the sub-catchment is sufficient. So it depends on the catchment size and data resolution whether different interpolation methods are going to have an effect or not. Moreover, recalibration is needed when switching precipitation data, to not work with errors in the model parameters that are not for that data set.

A next point of importance is the incorporation of the variable of elevation into the interpolation technique. This has been shown to improvements in the modelling results for monthly and annual precipitation (Ly et al., 2013). However, Haberlandt and Kite, 1998, who accessed interpolation results using the hydrological model SLURP, think that the relation is questionable in the case of daily observations, since the correlation is only 0.06, while it was on average 0.52 for annual precipitation. In contradiction, the research of Carrera-Hernández and

Gaskin, 2007 shows that the interpolation of daily events was improved by the use of elevation as a secondary variable, even when there was low correlation. They did not access their results using a semi-distributed hydrological model, but they set up the experiment with hydrological modelling in mind.

In a paper from Tobin et al., 2011, they compared different interpolation methdos for hourly precipitation in Alpine terrain by comparing the runoff predicted by a semi-distributed GSM-SOCONT model. They found that Krigging with External Drift (KED) with elevation data input performs better than other tested interpolation methods. However, note that they did this without re-calibrating the model for each technique, so we cannot conclude that this actually is the best method or that it is due to that KED with elevation cancels out some errors caused by the parameters.

In a paper of Chen et al., 2017, they tested several interpolation techniques on the Fuhe river in southeastern China using a semi-distributed model. They analysed them both on hourly, daily and monthly time-scales. They concluded that PCRR performed the best, since it can correct for effects due to terrain differences. PCRR stands for Principal Component Regression with Residual, and it uses macroscopic and microscopic topographic factors (such as latitude and longitude, elevation, slope, etc.) as covariates.

Lastly, Masih et al., 2011 tested Inverse Distance Weighting with Elevation (IDEW) on the SWAT model that normally uses an average between the closest 4 stations, and they found that IDEW improves the performance.

This we can summarise as that it seems that including elevation as a covariate leads to better runoff results.

Therefore it is worth looking at other covariates to include. Wagner et al., 2012 tested 7 different interpolation techniques for a location with scare rainfall measurements. They accessed the results both by crossvalidation and by analysing the output of the hydrological model SWAT. They found that regression interpolation methods perform the best and the most suitable covariates for rainfall interpolation were identified as: (i) distance in wind direction from the main orographic barriere and as (ii) a 0.05 degree pattern of mean annual rainfall derived from satellite data acquired by the Tropical Rainfall Measureing Mission (TRMM).

Moreover, Wagner et al., 2012 also found that the timescale of the data set is important in determining which interpolation technique performs the best. This means that interpolation methods can perform differently for hourly, daily, monthly or yearly data sets.

Johansson and Chen, 2005 have researched the effect of wind on the performance of a rainfall-runoff model, when wind information is included during the interpolation. They found 2 main factors in favour of using wind information in the interpolation: (1) a better description of the seasonal distribution; and (2) a lower sensitivity to reductions in the number of meteorological stations.

The review of Ly et al., 2013 states that the comparison between common deterministic methods (such as NN and IDW) and different types of geo-statistical methods is not made that often for daily rainfall. Moreover, they state that there are not that many studies that have tried to incorporate multivariate geostatistical methods (including elevation or other covariates) for daily rainfal interpolation (this is done for monlty and annual). So from that we can conclude that even though a lot of this research is done on larger time scales, for the daily time scale which HYPE works with, not much is tested yet.

As an overall conclusion, we can say that it is not well determined yet which methods perform 'best', especially for daily data sets. However, most studies point out that the 'best' interpolation method can vary with time-scale of the data, amount of precipitation and size of the sub-basins (which determines how important the spatial variability of the precipitation is). They also point out that including elevation and maybe wind data has the potential to improve the interpolation performances.

5 Limitations and my research question

In the introduction, I mentioned that the question I am supposed to look at with this project, is whether the use of different interpolation methods to obtain a precipitation value at the sub-basin center, can lead to an improvement of the HYPE model. In section 4, I have listed several papers that invest this matter, using different semi-distributed hydrological models, leading to the conclusion that the best interpolation method can vary with time-scale, amount of precipitation and the size of the sub-basins. It was also pointed out, that recalibration would be needed for every new input data set, derived by a different interpolation technique. This forms an important limitation.

The HYPE model has multiple hundreds of parameters. Only 11 of these parameters are often used for re-calibration, while the rest is kept as was determined during the set-up of HYPE (SMHI, n.d.). On top of that there are 9 parameters that are used for precipitation corrections, for example to make the model fit better to a certain area ("Hype model documentation", n.d.). This means that there would be at least 20 parameters

to recalibrate when using different precipitation data. Moreover, one can argue that all parameters would need recalibration, since most of them are found by optimisation to the runoff data with the current input precipitation data (Arheimer et al., 2020), and therefore they all include corrections for errors in the input data. Or at least it is unknown to us which ones do. Therefore, the whole model would need to be re-calibrated, which is not an option in the time span of a Master thesis project.

This leads to the fact that we will not be able to access possible improvements to HYPE by using a different interpolation method to obtain the precipitation input data.

Also, an analysis of the sensitivity of HYPE to these changes in input data cannot be analysed. One would say that the HYPE model would be sensitive to the choice of interpolation method, when a small change in input precipitation, would lead to a large difference in modelled discharge. However, these responses to a change in input precipitation depend on the parameters in the model and those parameters include corrections for the errors in the input data set that was used during the calibration.

The easiest example is to take one of the correction parameters: precorr, the precipitation correction. This is a correction factor that is multiplied with the input to determine the precipitation in a sub-basin ("Hype model documentation", n.d.). If precorr would be really large in the model that is calibrated towards the NN input (current HYPE model), then a small change in precipitation input would lead to a large change in runoff. However, this is just the effect of the current calibration and when we recalibrate, this precorr will be different and the model response to a change in input precipitation will be different (maybe not sensitive at all). So this model response to a change in input precipitation that you find, is specific to the calibrated model and is not necessarily the same when you recalibrate.

So with HYPE itself, it is not possible to analyse the effect of a different interpolation method, or it's sensitivity to different interpolation methods, since recalibration is not an option. Since we have seen in section 4 that other researches did analyse the effect of the interpolation methods with other models, it was considered to analyse the effect with a simple model consisting of just a few parameters and linking the results to HYPE. But due to the non-linearity of the models, it is not possible to argue that the findings with that model would be similar in the case of HYPE. Therefore, conclusions made about the best interpolation method for other models cannot be used to make a judgement on which methods would cause an improvement for HYPE (therefore they were also not mentioned in section 4). The only information we can use from that research is the factors they found that the choice of interpolation method depends on, such as sub-basin size, time-scale and amount of precipitation. And that the addition of a covariate like elevation or wind direction can improve the interpolation.

Due to the above stated limitations around the original research question, this additional thesis will not focus on answering this question. Only some first steps will be made, so that Lund University can continue on that in a larger research project. Therefore, what I will investigate in this additional thesis project is:

Do different interpolation methods result in a different discharge when we assume that all fallen precipitation would end up as discharge? And how do these agree with the discharge observations?

These question can be answered without running the model and can give us a first indication into which interpolation technique is the most promising. Since if the raw data time-series (computed with different interpolation techniques) follows the discharge observations better, we can expect to also obtain a better result after running the model. At least, this is the assumption that we make and will come back to in the discussion section.

6 Methods and data

To answer the research question, I will take 3 different steps. 1) Produce the code to perform 4 different interpolation methods and create the *Pobs.txt* and *ForceKey.txt* files that the WW-HYPE would need as input, so that these can be easily generated and used in future research. 2) Analyse the effect on the generated precipitation values at the sub-basin centers, to get an impression on if there is a difference when we use a different interpolation method. 3) Analyse the discharge we would obtain when all the precipitation would just stream down as discharge and compare this with the actual discharge observations. We do not expect to obtain the same time series as the discharge observations, since the model is not used, which takes evaporation and all other processes into account and therefore alters the time series. But we do expect to get a similar time series. We make the assumption, that the time series that is most similar to the observations, that this would be the

interpolation technique that best represent the precipitation in this area and could lead to better results if the model would be re-calibrated. I will discuss this assumption in the section 8.

In this section, I will first describe the data used and the areas of interest. Then I will further describe the 3 different steps that I mentioned above and specify the coefficients used to judge the agreement between the computed time-series and the observations.

6.1 Data

As mentioned in the introduction, I will work with World Wide HYPE (WW-HYPE), which has sub-basins with an average size of 1000 km^2. I will work with the HydroGDF2.0 data set for the precipitation, since this is the data that is used for WW-HYPE, causing the model to be calibrated to this data set. This gridded data set has a resolution of 50 km. This means that we have less data points than the average size of the sub-basins ($\sqrt{1000} \approx 31$ km), resulting in 1 or less data points in every sub-basin. Because the data points are so course, I do expect a difference by the use of different interpolation methods, since the NN can be very far away, and therefore less representative than an estimation from for example IDW.

The HydroGDF2.0 dataset is a merged dataset of historical precipitation from meteorological ERA-Interim reanalysis and global observations. It is bias adjusted for every month and is therefore a good approximation of the observed state at the monthly time scale, but not at daily. It covers a time-span of 1961 till present ("What is hydrogfd2.0?", 2021).

Note that the GDF2.0 dataset contains large negative precipitation values. These values are set to 0, since this made it easier for the computations. Do note that one could argue that it maybe would have been better to set the values to None, so that they do not get used in the computations, instead of filling it with a value that might be completely wrong. However, in that case it might occur that you take the value from really far away, which is, similarly as the 0 value, also not representative for that grid-point. Therefore both options are not ideal and it could be investigated in future research what the effect is of both.

For the discharge observations, the same observations that WW-HYPE already uses are used. These are time series from gauge stations from different open data sources globally. For Europe and therefore for the PO river, these observations come from the GRDC-EURO-FRIEND-water dataset. (Arheimer et al., 2020)

6.2 Area of interest

For now I looked at the PO river. The PO river is the longest river in Italy. The PO basin covers an area of approximately 70 000 to 71 000 km^2 and has an average discharge of 1500 m^3/s. The basin consists of the Alps, the Apennines and the PO valley. In the last one, most of the agricultural activity takes place and this area is densely populated. The agricultural land covers approximately 45% of the area and this area is artificially drained by ditches and irrigated during the summer (Coppola et al., 2014; de Wit & Bendoricchio, 2001). This human activity probably has a large influence on the water flow in this area and can cause a smalle percentage of the precipitation to end up as discharge.

In Figure 3 the PO basin is depicted with its elevation. This elevation will not be used in any of the interpolation methods, but is good to keep in mind when analysing the results of the interpolation, to see whether more differences occur in mountainous compared to flat areas, since this does have an influence on the spatial variability of the precipitation.

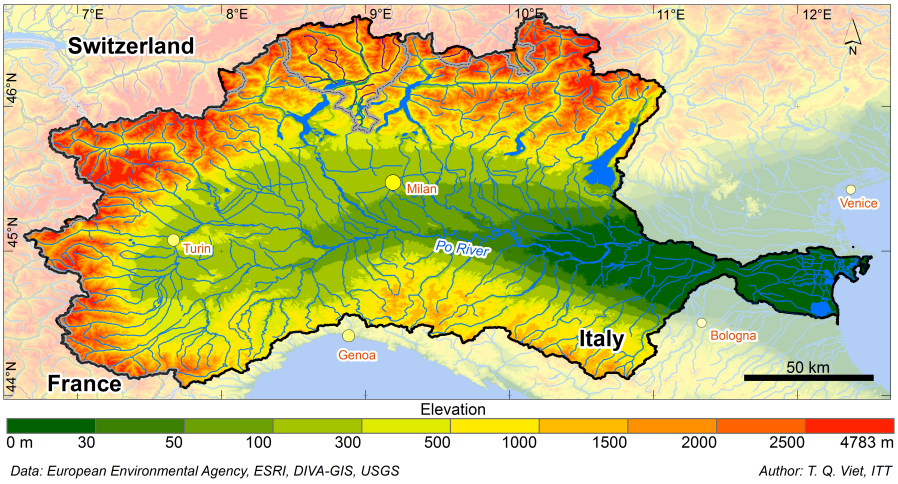


Figure 3: The elevation map of the PO basin. (Viet, n.d.)

For the area of interest, I used the shapefile and information of this area that is also given to HYPE as input in the *geodata.txt* file. This includes the division in sub-basins, there ID names, the upstream and downstream sub-basins, the coordinates of their center point and more.

The code that performs the interpolation techniques and produces the input files for HYPE, automatically selects the data for the area of interest based on the shapefile of this area. The GDF2 data is cropped to the extend of the sub-basin centers + 1 degree (longitude and lattitude).

Both the division in sub-basins from the *Geodata.txt* file and the cropping of the data can be observed in Figure 5 in section 7.

6.3 The 4 different interpolation methods

I have set up a python script, that can automatically generate the *Pobs.txt* and *ForceKey.txt* text files that the HYPE model takes in as input data. This can be done for every area of interest and by using one of the 4 interpolation methods. This script can be found in the Appendix.

The 4 interpolation methods I looked at are: Nearest Neighbour (NN), Inverse Distance Weighting (IDW), Bilinear Interpolation (BIL) and Ordinairy Krigging (OK). I did not look at any interpolation methods that include the elevation as a covariate, since we are firstly interested in whether any differences at all are observable after the use of different interpolation methods.

NN is a method that simply takes the value of the closest measured point. The distance is measured in kilometers and this is derived from the coordinates with the use of the haversine function.

IDW is a method that takes a weighted average of all stations, but gives more weight to stations that are nearby, compared to the stations that are far away. This can be expressed as:

$$\hat{X}(v_j) = \begin{cases} X(u_i) & \text{if } v_j = u_i \\ \sum_{i=1}^{n} w_{ij} \cdot X(u_i) & \text{else} \end{cases} \tag{1}$$

where

$$w_{ij} = \frac{d(u_i, u_j)^{-p}}{\sum_{i=1}^{n} d(u_i, u_j)^{-p}} \tag{2}$$

Here d is the distance between 2 points, which is again a distance in km obtained with the haversine function. I used a distance decay parameter p of 2, which means that the weights are inversely proportional to the squared distance. This is known as the inverse distance squared weighted interpolation. Other values of p are also possible. If p would be 0, it would be the mean of the whole dataset. If p is infinity, you get the nearest neighbour. To find the optimal value of p, you should try out multiple and verify which one is the best. This is shown in the the first part of the results section.

BIL computes a weighted average of the 4 closest neighbours by first interpolating along both axis. In this case the closest neighbours are simply found by comparing the coordinates, so the distances are kept in degrees. The estimate with BIL can be written in a single expression:

$$\hat{Z}(P) = (1-s)(1-t)\cdot Z(P_1) + s(1-t)\cdot Z(P_2) + t(1-s)\cdot Z(P_3) + s\cdot t\cdot Z(P_4) + \tag{3}$$

where Z is the measurements at the 4 closest locations $P_{1,2,3,4}$ and $s = \frac{x-x_1}{x_2-x_1}$ and $t = \frac{y-y_1}{y_2-y_1}$. Here x and y represent the longitude and latitude respectively.

OK computes the estimated value by a weighted sum of the available point observations. The weights for the different point observations are chosen so that the interpolation is unbiased and the variance is minimized (Ly et al., 2013). It does this by using BLUE. This method takes the spatial dependence structure of the data into account. For ordinary krigging it is assumed that the mean is constant but unknown.

OK is performed separately for every day. The data of a day is first detrended after which a variogram is computed. It was chosen to devide the variogram cloud in 17 bins, since more bins resulted in the first bins being empty. For the PO river, this variogram is cut off at 450 km, since that is where the variogram started to flatten for most days. Note that the variogram is not checked for every day, so for some it could obtain a better fit with a different cutoff. This is something that can be further investigated in future research. Through this variogram, the 4 most common models are fitted:
Exponential:

$$\gamma(d) = \begin{cases} 0 & \text{if } d = 0 \\ n + (s-n)\left(1 - \exp\left(\frac{-3d}{r}\right)\right) & \text{else} \end{cases}$$

Spherical:

$$\gamma(d) = \begin{cases} 0 & \text{if } d = 0 \\ n + (s-n)\left(\frac{3d}{2r} - \frac{d^3}{2r^3}\right) & \text{if } 0 < d < r \\ s & \text{if } d \geq r \end{cases}$$

Gaussian:

$$\gamma(d) = \begin{cases} 0 & \text{if } d = 0 \\ n + (s-n)\left(1 - \exp\left(\frac{-3d^2}{r^2}\right)\right) & \text{else} \end{cases}$$

Linear:

$$\gamma(d) = \begin{cases} 0 & \text{if } d = 0 \\ n + (s-n)\left(\frac{d}{r}\right) & \text{if } 0 < d < r \\ s & \text{if } d \geq r \end{cases}$$

In these formulas, n denotes the nugget, s the sill, r the range and d the distance in km.

The model with the lowest Mean Squared Error (MSE) is selected as the best and used for estimation at the sub-basin centers for that day. After the estimation, the trend is added again. After analysing these values, it appeared that this method resulted in some negative precipitation values. This can be caused by the 'screening effect', which is a common effect in krigging. It can be prevented by only using nearby points instead of the whole data set, or by using a correction for these negative weights as proposed by Deutsch, 1996. However, for now it was chosen to interpret these slight negative values as no precipitation, so as 0. After that correction, the values are stored in the Pobs.

Note that performing OK separately for every day is very computation expensive. However, it does have as advantage that it is optimised for the precipitation pattern for that day and gives potentially the best value. But we do need to keep in mind that the success of krigging depends on how well the variogram can represent the spatial structure of the data, and we simply can't check this for every day since we are working with a too large dataset (Lu & Wong, 2008). Therefore, and due to the way of handling the negative precipiation values, it might be that this way of krigging does not result in any better estimates.

6.4 Analyse the effect on precipitation at the sub-basin centers

Before analysing the discharge time series that we obtain with the different interpolation techniques, it was checked if the different interpolation techniques produced any differences in precipitation at all, by simply plotting the expected value at the sub-basin centers in a scatter plot. This was done, both for daily and monthly values. The results and the implications and expectations those results bring can be found in section 7.

6.5 Analyse the discharge time series

Since we are working with the HydroGDF2.0 dataset, which are bias corrected per month, only the monthly values are actually reliable to work with. Therefore I will look at the monthly timeseries. The daily observations that are now saved in the *Pobs.txt* per interpolation method, are resampled to a monthly timeseries. Then the precipitation value is transfered to a volume by multiplying it by the area of the sub-basin, which is given in the *Geodata.txt* file. This is multiplied by $\cdot 10^{-3}/(24 \cdot 60 \cdot 60)$, to convert the units $[mm/day] \cdot [m^2]$ to m^3/sec. Thereafter, all sub-basins are grouped to the sub-basin that has a discharge observation, by adding all the precipitation values of the upstream sub-basins. Figure 4 shows which sub-basins are grouped together, which sub-basins have a discharge measurement and where the water flows after the sub-basin with the discharge measurement. It can be observed that we obtain some groups that consist of only 1 sub-basin: 312294, 309614 and 310350. Some of only a few: 306719 and 311235. And others that are very large and have multiple upstream areas.

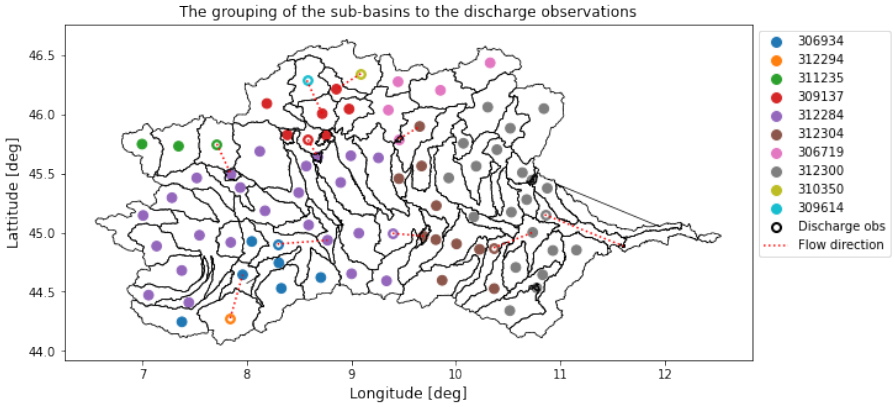


Figure 4: In this figure it is depicted which sub-basins stream to each discharge observation and how the water continues to flow after this observation point.

Before analysing the obtained timeseries, it is checked if the computation of the volume and the grouping were performed correctly, by comparing the estimated discharge of the NN interpolation, with some of the HYPE model output. As output you can request the precipitation in a sub-basin (parameter *prec*) and the precipitation in the whole upstream area (parameter *upcprc*). To obtain the discharge volume, the *prec* is multiplied with the area of the sub-basin with the observation station and the *upcprc* is multiplied with the upstream area, given by UPAREA in *Geodata.txt*. These values are added and give the discharge from the precipitation inputs as received by the model. So without any effects of the actual model.

After this check, the added precipitation values for these sub-basin groups are plotted as a time-series and compared with the discharge observations. How well the timeseries agree with the observations is determined based on 4 standard coefficients. Firstly, the correlation between the two time series. If they would perfectly follow each other, there would be a correlation of 1 and if they do not follow one another at all, the correlation would be 0. It is both analysed what the correlation is, and if a time shift would obtain a higher correlation or not. This is checked, since it could be that the precipitation needs some time to be transported all the way down to the discharge observation point.

Secondly, the Relative error (RE) (Arheimer et al., 2012), which measures the ability of the model to reproduce a long-term mean value and is expressed as a percentage of the observed mean value (with 0% as a perfect fit). See Equation 4. Here the M stands for the modelled value and the O for the observations.

$$RE = \frac{\overline{M} - \overline{O}}{\overline{O} \cdot 100[\%]} \tag{4}$$

As a third coefficient, we have the Nash and Sutcliff (NSE) coefficient (Nash & Sutcliffe, 1970). This one measures the daily error of the model compared to the variance in the observations. A value of 1 represents the perfect fit. A value of 0 means that the model does equally good as taking the mean value. A value above 0 means that the model mimics the variance of the observations better than the mean of the observations. See Equation 5. Again the M stands for the modelled value and the O for the observations.

$$NSE = 1 - \frac{\sum_{i=1}^{n}(O_i - M_i)^2}{\sum_{i=1}^{n}(O_i - \overline{O})^2} \tag{5}$$

Lastly, we have the Kling-Gupta efficiency (KGE), which combines the 3 components of the NSE model errors in a more balanced way: the correlation, bias and ratio of variances (Liu, 2020). It is the Euclidian distance to an ideal value in the 3D space of the 3 components. Negative values are seen as bad performance, and only positive as good (Knoben et al., 2019). A KGE of -41 indicates that the model performs equally good as taking the mean value (which is at an NSE of 0). A KGE of 1 indicates the perfect fit. The equation the calculated the KGE can be found in Equation 6. Here the r is the correlation, the σ the standard deviation for the simulations or observations, and the μ the mean.

$$\mathrm{KGE} = 1 - \sqrt{(r-1)^2 + \left(\frac{\sigma_{\mathrm{sim}}}{\sigma_{\mathrm{obs}}} - 1\right)^2 + \left(\frac{\mu_{\mathrm{sim}}}{\mu_{\mathrm{obs}}} - 1\right)^2} \tag{6}$$

There is no unique relationship between NSE and KGE and they should both be analysed separately. These 4 coefficients together can give an indication whether there is a difference between the resulted discharge from the interpolation methods. If all coefficients are almost similar for every interpolation method, we would come to the conclusion that for this basin with this data set and sub-basin size, a different interpolation method would not lead to any improvements. If all coefficients would clearly indicate one interpolation method to perform better, this could be suggested as a possible better method and can be tested with recalibration.

7 Results

In this section we will analyse the results in 2 steps. First we will analyse the effect of the precipitation at the sub-basin centers. Secondly we will take a look at the discharge time series computed from the interpolated precipitation, compared with the discharge observations.

7.1 Analyse the effect of precipitation at the sub-basin centers

After the *Pobs.txt* are produced with the 4 different interpolation methods, their effect is analysed by scattering these results on top of the shapefile of the area of interest. In Figure 5 you can first see the original gridded dataset on top of this shapefile. This is the dataset for a single day: January 3th 1961. This day was chosen to depict, since it shows a quite heterogeneous rainfall field and therefore shows the effect of the interpolation methods quite clearly. In Figure 6 it is depicted how the precipiation on this day gets interpolated to the sub-basin centers by the 4 different interpolation techniques. It can be seen that NN, BIL and OK keep more extreme values, while IDW with p=2 obtains more average values as estimates. Which effect would be more desired will become clear when we compute the discharge.

The results that are depicted in Figure 6 shows that the different interpolation methods can lead to some differences in some sub-basins. However, when we look at other days where the precipitation is more homogeneous or contain smoother gradients, those differences disappear almost completely.

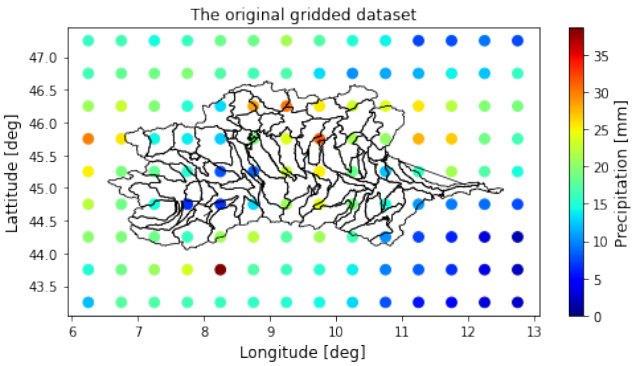


Figure 5: The original gridded HydroGDF2.0 dataset on top of the area of interest, cropped as described in the methods section: coordinates of the area of interest +/- 1 degree. The data depicted is for one day: January 3th 1961.

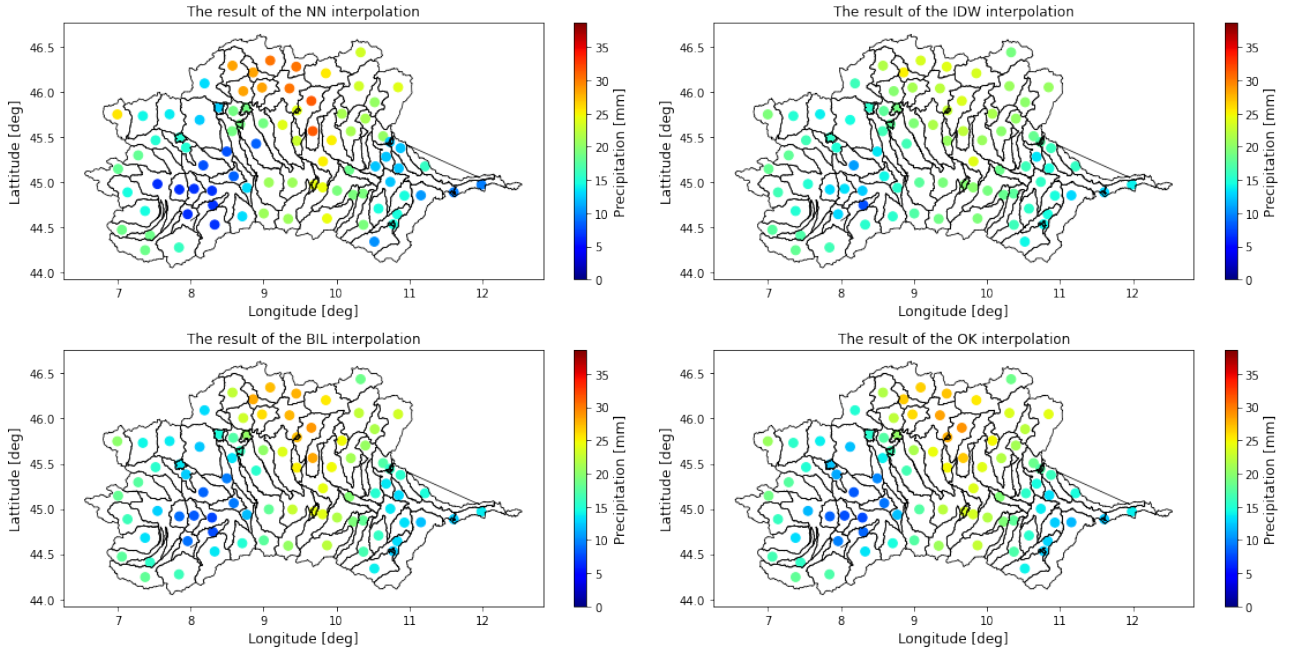


Figure 6: The result of the 4 different interpolation methods for one specific day: January 3th 1961.

Before we further analyse the effect of the interpolation methods, I want to come back to the chosen value of 2 for the distance decay parameter p in the IDW method. In Figure 7, the result of IDW with different values of p can be observed. For $p = 1$ we already have the situation of the estimates representing the mean of the whole area, and for $p = 3$ we already have estimates that are almost equal to the NN estimates (see Figure 6). Therefore, the value of 2 is chosen, since this is a quite standard value and does give different estimates. If you would want to further optimise the value of p, this could be done by only using part of the gridded data and estimating the rest, so that we can select the p that produces the best estimates. However, due to the results discussed in the rest of this paper, this optimisation is not performed.

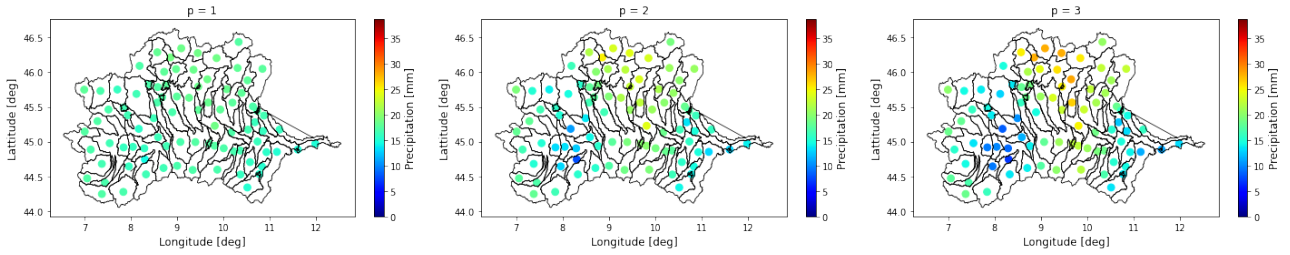


Figure 7: The result of IDW interpolation with different values of the distance decay parameter p. Depicted are the results for one specific day: January 3th 1961.

Back to analysing the effect of the different interpolation methods. As mentioned before, as soon as the precipitation is more homogeneous or contain smoother gradients, the difference between interpolation methods disappear. This would indicate, that the interpolation methods only matter if we have multiple days with smaller local precipitation events.

In the method section it was mentioned that the time-series will be a monthly time-series, since the HydroGDF2.0 is only bias corrected for every month. Therefore it is important look at the difference that we can observe on this monthly time-scale. An example is depicted in Figure 8 and 9. Here it can be observed, that some differences remain in some of the sub-basins when we sum everything up for a whole month. But there are also a lot of sub-basins where those differences completely disappear. And for other months than the one depicted here, those differences disappear for even more sub-basins. Since we have a long time-series (more than 50 years), it is hard to observe by eye how big the differences are in each month for every sub-basin. This will become clear when we analyse the time-series in the next section.

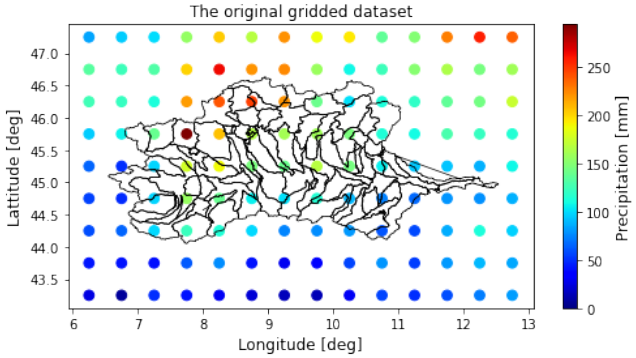


Figure 8: The summed precipitation for the month July in 2001 in the original dataset.

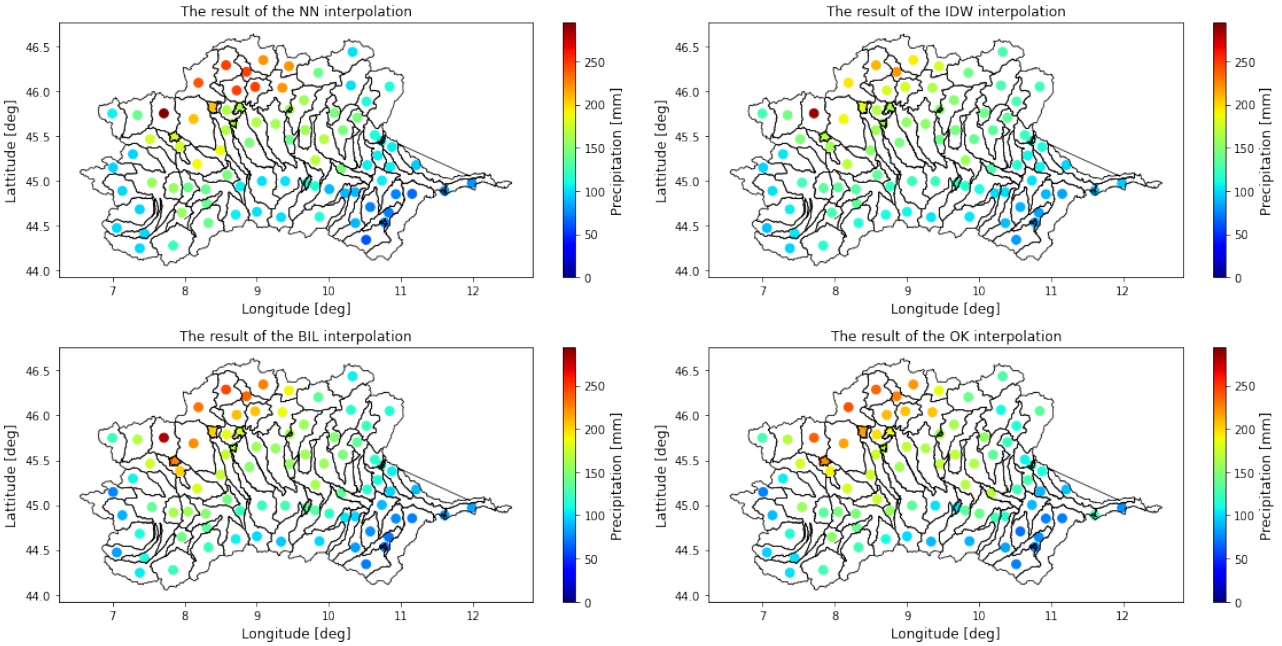


Figure 9: The summed precipitation at the sub-basin centers computed with the 4 interpolation methods. For the month July in 2001.

Looking back at Figure 6 and Figure 9, it can be observed that in both cases, the differences between the interpolation methods are present in the north part of the PO river basin. However, by observing individual months or days, it is hard to get an indication if this is always the location with the highest differences. But if it is this area where most differences occur and we compare it with Figure 4, we come to the hypothesis that differences should be mostly visible in the time-series of sub-basins: 309137, 309614 and 310350.

7.2 Analyse the discharge times series

7.2.1 Check the discharge and grouping computations

Before we start the analysis of the time-series compared with the discharge observations, it was first checked if the computations resulted in the desired result, as described in the method section. However, while comparing this, 2 remarkable things occured.

Firstly, when we compare the NN Pobs that I have computed for a sub-basin, with the *prec* output that the model gives, all values are the same if you ignore different rounding of the values (model output *prec* is rounded on 2 decimals and *upcprc* to 3 decimals, my values are not rounded), except for the last 3 dates, these precipitation values are divided by 2 in the model output. Since the rest is the same, we can assume that we did take the same nearest neighbour, but that the model just did something strange for the last 3 dates.

Secondly, when we plot the computed discharge next to each other, it can be observed that for some sub-basins, the time series are exactly the same (small differences of the order of 0.1 m^3/sec on a scale of 10 000

m^3/sec), but that for others, there is a shift (up to 1000 m^3/sec on a scale of 200 000 m^3/sec). The sub-basins where it is exactly the same are the 3 sub-basin groups that consist of only 1 sub-basin. For all the others, the same pattern is followed, my estimation is just slightly higher than what you would expect from the model.

While investigating several reasons, it was found that the sum of the area of all the sub-basins did not match with the sum of the upstream area of the last sub-basin (307329) plus its own area. Both these area and uparea values are given in the *Geodata.txt* file that the HYPE model takes as input. This results in different total areas, as can be seen in Table 1. Note that both of these areas do not agree with the area you can find on the internet when you google PO river basin, since that is around 70000 to 71000 km^2, as was mentioned in section 6. Moreover, when the same test was performed for one of the sub-basin groups: 306719 (pink in Figure 4), different areas were obtained. This is depicted in the same table.

As can be seen in Table 1, for the whole PO basin, the UPAREA should be higher than it actually is. However, for the sub-group 306719, the UPAREA should be lower. For this sub-group, this difference cannot explain why we get lower discharge values with the HYPE model output. Since if HYPE has used the larger area, the UPAREA, as area to determine the mean uparea precipitation *upcprc*, and we now multiply it again by the UPAREA, the estimate should be correct. And if HYPE used the lower area value, the sum of all the areas, then we would have gotten a too high discharge estimate instead of one that is too low. Also, when it is checked for sub-basin group 305719 what the effect is of the different areas, the effect is only $\approx 0.002 m^3/sec$ on a scale of around $6000 m^3/sec$. Therefore we can conclude that this difference cannot have caused the shift.

Sub-basin group	UPAREA + AREA sub-basin (m^2)	sum sub-basin AREA's (m^2)	Difference (m^2)
307329 (whole PO basin)	73503346880	73503351013	4133
306719	4596552194	4596551938	256

Table 1: 2 examples of that the UPAREA and AREA in the *Geodata.txt* file do not match.

Despite that it cannot be confirmed by this approach that this is the reason for the shift, it is remarkable that the areas in the *Geodata.txt* input file do not agree, and also not agree with the total area of the PO basin found on the internet. This is apparently something that went wrong when someone set up this model for the PO river. Even though this approach did not indicate this as the cause of the shift, it might still play a part in it, since it is unknown how the model exactly generates it output.

As a last remark, I want to mention that it could also be the rounding errors in the NN estimates that cause the difference. In that case it would play a role in all sub-basins. We only observe it more in larger sub-basin groups, since the area the rounding error is multiplied with is a lot larger.

However, when we analyse the difference between the 2 estimates for sub-basin group 306719, as depicted in Figure 10, we can observe that this difference increases when the amount of precipitation increases and that the difference decreases when the amount decreases. This would be contradicting the hypothesis of it being caused by the rounding errors, since the rounding errors do not increase when the precipitation increases.

Since the same pattern is followed, the shift is a maximum of 0.005% of the discharge, and it is unknown how the model exactly computed its output, I will assume that the computations are correct and continue with these values.

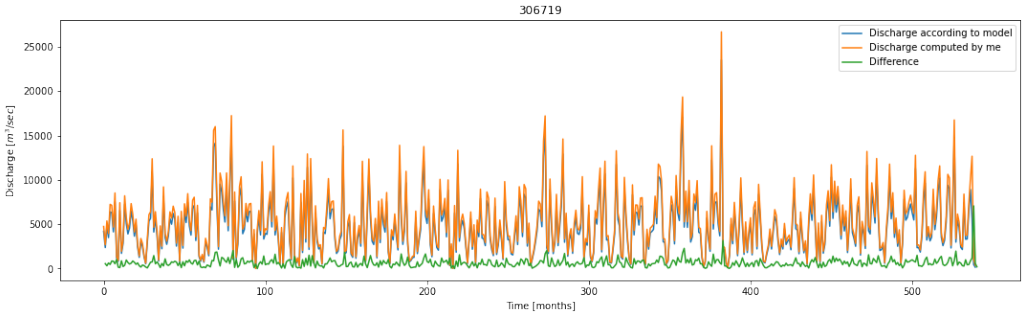


Figure 10: The difference in computed discharge between the NN output from the HYPE model and the NN discharge that I computed.

7.2.2 Compare the time-series with the discharge observations

Based on the analysis of the effect of the precipitation at the sub-basin centers, we expect to see the largest difference in sub-basins 309137, 309614 and 310350. Figure 11 seems to show a larger difference for these sub-

basins. However, note that the scale on the y-axis is different, so that the percentual difference is maybe larger, but the absolute difference might be the same or even smaller as in for example sub-basin 312300.

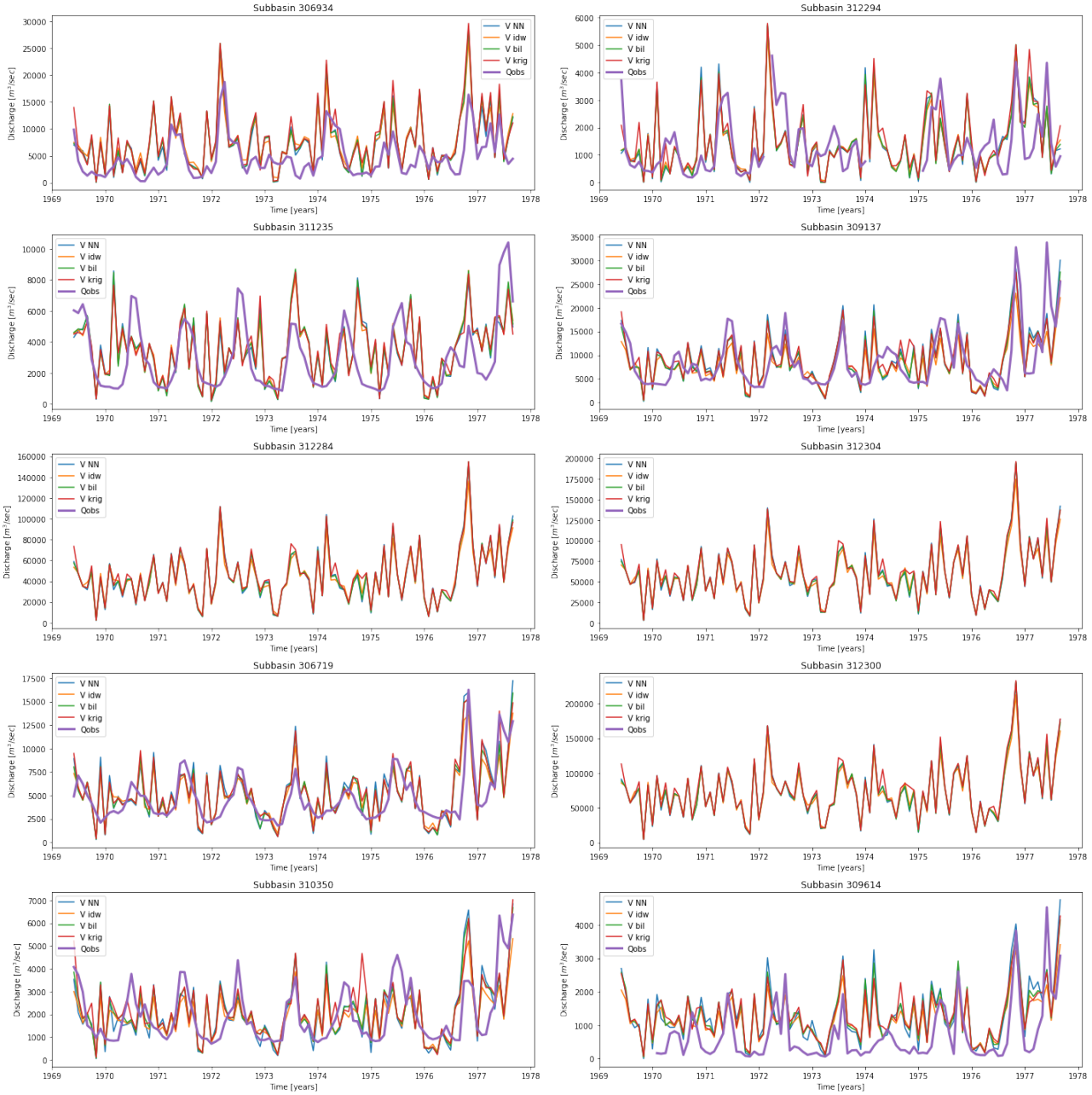


Figure 11: The computed discharge timeseries for the 4 interpolation methods and the observed discharge timeseries. It is only depicted for a few years (randomly chosen), to make any differences visible.

Before we further investigate the differences, we can observe 2 more phenomena in Figure 11. Firstly, observations are not present for every sub-basin at all times, that is why some sub-basins do not have any Qobs for this time-span. More importantly, Table 2 displays the amount of months with observations. This needs to be kept in mind when analysing the coefficients.

Sub-basin	306934	312294	311235	309137	312284	312304	306719	312300	310350	309614
# months	444	365	413	408	72	72	504	72	648	456

Table 2: The amount of months with discharge observations, per sub-basin.

Secondly, for the sub-basins where we do have the observations, the estimates seem to partly align with them.

We can see that mostly when there are peaks in the observations, these peaks are present in the estimates as well. However, the estimates contain a lot more peaks than the observations do. One could argue that this is be as expected, since in reality, water gets stored in snow or lakes fore some time and therefore this peaky behaviour as can be seen in the estimated time-series, gets flattened out.

For these time-series it was checked if they would better align when shifted in time, by checking if the correlation increased after a time-shift of the observations. However, it showed that the highest correlation was obtained when there was no time-shift applied, which indicates that the precipitation streams down to the discharge observation point within a month.

Since the visual analysis does not lead to any clear conclusions, lets look at the coefficients that were described in Section 6. These are depicted in Table 3, 4, 5, 6, where the coefficients for every interpolation method and sub-basin are depicted, and the largest differences between the interpolation method within a sub-basin are highlighted. These coefficients are computed while taking all the months with observations into account (see Table 2).

Sub-basin	306934	312294	311235	309137	312284	312304	306719	312300	310350	309614
NN	0.661	0.439	0.283	0.580	0.674	0.633	0.607	0.604	0.596	0.689
IDW	0.632	0.448	0.295	0.576	0.657	0.629	0.606	0.602	0.607	0.691
BIL	0.659	0.457	0.299	0.585	0.675	0.639	0.621	0.609	0.613	0.696
OK	0.662	0.461	0.302	0.580	0.674	0.641	0.626	0.610	0.611	0.684
Max diff	0.029	0.018	0.016	0.010	0.018	0.010	0.016	0.007	0.017	0.007

Table 3: The correlation

Sub-basin	306934	312294	311235	309137	312284	312304	306719	312300	310350	309614
NN	74.264	16.553	29.510	0.450	49.765	50.133	8.440	50.494	-1.355	89.575
IDW	85.290	17.220	28.295	-9.380	51.479	50.547	3.559	52.752	-6.640	71.361
BIL	79.753	15.512	29.449	-2.545	52.665	51.713	4.008	52.529	3.571	90.560
OK	83.283	17.818	29.005	-4.238	53.328	52.694	4.639	54.001	1.426	83.936
Max diff	11.026	1.708	1.216	9.830	2.900	1.580	4.881	2.258	10.211	19.199

Table 4: The RE

Sub-basin	306934	312294	311235	**309137**	312284	312304	306719	312300	**310350**	**309614**
NN	-0.876	-0.401	-0.721	0.221	-1.652	-2.132	-0.233	-2.463	0.127	-0.092
IDW	-1.040	-0.240	-0.613	0.300	-1.492	-1.806	0.133	-2.267	0.306	0.180
BIL	-1.081	-0.302	-0.735	0.270	-1.791	-2.172	-0.027	-2.560	0.180	-0.042
OK	-1.170	-0.304	-0.639	0.274	-1.812	-2.216	-0.041	-2.659	0.217	0.032
Max diff	0.204	0.161	0.123	0.080	0.299	0.365	0.366	0.292	0.179	0.272

Table 5: The NSE

Sub-basin	306934	312294	311235	309137	312284	312304	306719	312300	310350	309614
NN	0.165	0.402	0.207	0.571	0.381	0.338	0.521	0.310	0.586	-0.057
IDW	0.030	0.416	0.216	0.506	0.378	0.361	0.601	0.322	0.587	0.040
BIL	0.116	0.434	0.227	0.566	0.361	0.334	0.565	0.302	0.611	-0.085
OK	0.083	0.432	0.223	0.556	0.357	0.329	0.563	0.293	0.610	-0.045
Max diff	0.135	0.031	0.019	0.066	0.020	0.027	0.080	0.020	0.025	0.124

Table 6: The KGE

In Table 3 we can observe that there is almost no difference in correlation between the 4 interpolation methods, and that this difference is not the largest for the 3 basins we had identified before. Furthermore we can observe that the correlation is around the 0.6 for most of the sub-basins. Except for sub-basin 312294 and 311235. These are a sub-basin group consisting of 1 sub-basin at the south west part of the PO basin, and a group consisting of 3 sub-basins in the north west part. Both are located at higher elevations in the mountains.

This could explain the lower correlation, since precipitation can fall more in the form of snow and therefore the discharge would be more dependent on the freezing and melting and not only on the amount of precipitation. However, this same effect would then be expected for sub-basins 309614, 310350 and 306719, since they are also only located in the mountains in the north, but this is not the case. (See Figure 4 and 3).

The RE is depicted in Table 4 and it can be observed that the maximum difference between the interpolation methods is quite a lot higher for the 3 previously identified regions: 309137, 309614 and 310350. However, also sub-basin 306934, which is a mountainous region in the south, shows an equally large RE. The reason of this is unclear, since there are more other small mountainous regions that do not show a larger difference in RE.

Table 5 shows a negative NSE for most sub-basins, except for the 3 previously indicated. A negative NSE means that the mean of the observation time series is a better predictor than the computed discharge, which means that the discharge that is computed by the assumption that all precipitation ends up as discharge immediately, is not a good estimator. Since this assumption is known to not be true, this negative NSE is not a surprise. However, it is remarkable that the NSE is positive for exactly the 3 sub-basins of interest. And since 309137 (red in Figure 4) does cover a larger area where you would expect more forms of delay, it is unlikely that the assumption does hold for this area. Therefore it might just be coincidence, which seems indeed more likely in combination with the analysis of the KGE.

When we look at the max difference in the NSE between the different interpolation methods, we do not obtain the largest difference for the 3 areas of interest. The areas with the largest differences are a small area in the north and a larger area in the middle.

Lastly, Table 6 shows that all sub-basins except for 309614 have positive KGE values, which indicate a 'good' fit and therefore contradict the negative values of the NSE. This is strange, since both coefficients are based on the correlation, bias and ratio of variances, and only balanced in a different way. Therefore I would have expected more similar indications. When we analyse the maximum difference, we can see that this is very low for most sub-basins.

Analysing the highlights in all 4 tables, it appears that the 3 previously identified sub-basin groups were a result of just visually analysing a small subset of the data. Analysing the RE does indeed indicate that the differences that we can visually observe in Figure 11 can be an effect of the scale on the y-axis. Since large visual differences does not match completely with a large RE.

From the highlights, it can also be observed that sub-basin group 306934 has most often a larger difference for the different interpolation methods. Based on elevation this cannot be explained, since this sub-basin group contains both higher and lower elevations. If there is another factor to which it can be linked, or if it is just coincidence, should be further investigated, for example by applying the same analysis to different regions than the PO river.

Looking at all the differences, we can observe that the differences in correlation are very small. The differences in RE is a bit bigger for the 4 mentioned sub-basin groups, but still very small for the others, especially when you compare them with the RE value itself. The NSE values are slightly strange since they are all negative, indicating a bad fit, while the KGE values are positive, indicating a good fit. Therefore the NSE differences are hard to interpret. The KGE differences are again very small. So because the correlation, most RE and the KGE differences are all very small, we can most likely conclude that the interpolation methods do not cause that much difference. At least not differences that the model cannot correct of with its parameters.

8 Conclusion and Discussion

As discussed in section 7, no clear differences were obtained in the discharge time series after the use of different interpolation methods. This would indicate that for the PO basin, with a 1000 km^2 average sub-basin size and a gridded data set with a resolution of 50 km, the interpolation method does not matter and cannot improve the model.

As a next step in this research process, I would advice to investigate whether applying the same methodology to the PO basin, but with a different sub-basin size (for example by using E-HYPE), results in different results. Or alternatively, by using the same sub-basin size, but with a different data set. Since we are not running the model yet, it is not yet of importance that we use the GDF2.0 data set. If this indeed results in larger differences for the discharge computed with different interpolation methods, this would be in agreement with the results of Haberlandt and Kite, 1998, Segond et al., 2007 and Smith et al., 2004.

In that case, it would also be interesting to further investigate this in different regions, try other interpolation methods that include elevation and other covariates, and in the end test the findings of the 'best' interpolation method by re-calibrating and running the model. However, if this also does not show any differences, we can come to the conclusion that HYPE can keep using NN, since this does not result in a worse performance and is the least computational expensive.

In this methodology, one main assumption was made: If the computed discharge time-series follows the discharge observations better, we can expect to also obtain a better result after running the model. In the end, this assumption was not used, since all interpolation methods estimated such similar results. Nonetheless I want to discuss this assumption further. One could argue that by selecting a time-series that is closest to the observations, we prevent certain processes to be simulated in the model when we would actually run it. For example, if we would have several time series, just slightly shifted in amount, we would not aim to pick the one that is closest to the observations. However, processes as evaporation, that lower the amount of water, do get limited by this decision. Therefore one could argue that this assumption does not hold. But one could agree, but argue that the closer we already are to the observations, the more likely it will be that we obtain a good result. Therefore, this assumption can be used as a start, but the only way to find if the interpolation method really improves HYPE is to recalibrate and run the model.

References

Arheimer, B., Dahné, J., Donnelly, C., Lindström, G., & Strömqvist, J. (2012). Water and nutrient simulations using the hype model for sweden vs. the baltic sea basin–influence of input-data quality and scale. *Hydrology research*, *43*(4), 315–329.

Arheimer, B., Pimentel, R., Isberg, K., Crochemore, L., Andersson, J., Hasan, A., & Pineda, L. (2020). Global catchment modelling using world-wide hype (wwh), open data, and stepwise parameter estimation. *Hydrology and Earth System Sciences*, *24*(2), 535–559.

Carrera-Hernández, J., & Gaskin, S. (2007). Spatio temporal analysis of daily precipitation and temperature in the basin of mexico. *Journal of Hydrology*, *336*(3-4), 231–249.

Chaubey, I., Haan, C., Grunwald, S., & Salisbury, J. (1999). Uncertainty in the model parameters due to spatial variability of rainfall. *Journal of Hydrology*, *220*(1-2), 48–61.

Chen, T., Ren, L., Yuan, F., Yang, X., Jiang, S., Tang, T., Liu, Y., Zhao, C., & Zhang, L. (2017). Comparison of spatial interpolation schemes for rainfall data and application in hydrological modeling. *Water*, *9*(5). https://doi.org/10.3390/w9050342

Cheng, M., Wang, Y., Engel, B., Zhang, W., Peng, H., Chen, X., & Xia, H. (2017). Performance assessment of spatial interpolation of precipitation for hydrological process simulation in the three gorges basin. *Water*, *9*(11). https://doi.org/10.3390/w9110838

Coppola, E., Verdecchia, M., Giorgi, F., Colaiuda, V., Tomassetti, B., & Lombardi, A. (2014). Changing hydrological conditions in the po basin under global warming. *Science of the total environment*, *493*, 1183–1196.

Deutsch, C. V. (1996). Correcting for negative weights in ordinary kriging. *Computers & Geosciences*, *22*(7), 765–773.

de Wit, M., & Bendoricchio, G. (2001). Nutrient fluxes in the po basin. *Science of the Total Environment*, *273*(1-3), 147–161.

Haberlandt, U., & Kite, G. (1998). Estimation of daily space–time precipitation series for macroscale hydrological modelling. *Hydrological Processes*, *12*(9), 1419–1432.

Hundecha, Y., Arheimer, B., Donnelly, C., & Pechlivanidis, I. (2016). A regional parameter estimation scheme for a pan-european multi-basin model. *Journal of Hydrology: Regional Studies*, *6*, 90–111.

Hype model documentation. (n.d.). http://www.smhi.net/hype/wiki/doku.php?id=start%5C%3Ahype_model_description%5C%3Aprocesses_above_ground#precipitation_adjustments

Johansson, B., & Chen, D. (2005). Estimation of areal precipitation for runoff modelling using wind data: A case study in sweden. *Climate Research*, *29*(1), 53–61.

Knoben, W. J., Freer, J. E., & Woods, R. A. (2019). Inherent benchmark or not? comparing nash–sutcliffe and kling–gupta efficiency scores. *Hydrology and Earth System Sciences*, *23*(10), 4323–4331.

Lindström, G., Pers, C., Rosberg, J., Strömqvist, J., & Arheimer, B. (2010). Development and testing of the HYPE (Hydrological Predictions for the Environment) water quality model for different spatial scales. *Hydrology Research*, *41*(3-4), 295–319. https://doi.org/10.2166/nh.2010.007

Liu, D. (2020). A rational performance criterion for hydrological model. *Journal of Hydrology*, *590*, 125488.

Lu, G. Y., & Wong, D. W. (2008). An adaptive inverse-distance weighting spatial interpolation technique. *Computers & geosciences*, *34*(9), 1044–1055.

Ly, S., Charles, C., & Degré, A. (2013). Different methods for spatial interpolation of rainfall data for operational hydrology and hydrological modeling at watershed scale: A review. *Biotechnologie, Agronomie, Société et Environnement*, *17*(2), 392–406.

Masih, I., Maskey, S., Uhlenbrook, S., & Smakhtin, V. (2011). Assessing the impact of areal precipitation input on streamflow simulations using the swat model 1. *JAWRA Journal of the American Water Resources Association*, *47*(1), 179–195.

Massazza, G., Tarchiani, V., Andersson, J., Ali, A., Ibrahim, M. H., Pezzoli, A., De Filippis, T., Rocchi, L., Minoungou, B., Gustafsson, D., et al. (2020). Downscaling regional hydrological forecast for operational use in local early warning: Hype models in the sirba river. *Water*, *12*(12), 3504.

Nash, J., & Sutcliffe, J. (1970). River flow forecasting through conceptual models part i — a discussion of principles. *Journal of Hydrology*, *10*(3), 282–290. https://doi.org/https://doi.org/10.1016/0022-1694(70)90255-6

Ruelland, D., Ardoin-Bardin, S., Billen, G., & Servat, E. (2008). Sensitivity of a lumped and semi-distributed hydrological model to several methods of rainfall interpolation on a large basin in west africa. *Journal of Hydrology*, *361*(1-2), 96–117.

Schuurmans, J., & Bierkens, M. (2007). Effect of spatial distribution of daily rainfall on interior catchment response of a distributed hydrological model. *Hydrology and Earth System Sciences*, *11*(2), 677–693.

Segond, M.-L., Wheater, H. S., & Onof, C. (2007). The significance of spatial rainfall representation for flood runoff estimation: A numerical evaluation based on the lee catchment, uk. *Journal of Hydrology*, *347*(1-2), 116–131.

Seth, S. W. (2016). Physically based hydrological modelling. https://www.geospatialworld.net/article/physically-based-hydrological-modelling/

SMHI. (n.d.). Hype model documentation. http://www.smhi.net/hype/wiki/doku.php?id=start%3Ahype_tutorials%3Ashort_intro

Smith, M. B., Koren, V. I., Zhang, Z., Reed, S. M., Pan, J.-J., & Moreda, F. (2004). Runoff response to spatial variability in precipitation: An analysis of observed data. *Journal of hydrology*, *298*(1-4), 267–286.

Tobin, C., Nicotina, L., Parlange, M. B., Berne, A., & Rinaldo, A. (2011). Improved interpolation of meteorological forcings for hydrologic applications in a swiss alpine region. *Journal of Hydrology*, *401*(1-2), 77–89.

Viet, T. (n.d.). Po river basin (italy). http://www.basin-info.net/river-basins/po-river-basin-br-europe

Wagner, P. D., Fiener, P., Wilken, F., Kumar, S., & Schneider, K. (2012). Comparison and evaluation of spatial interpolation schemes for daily rainfall in data scarce regions. *Journal of Hydrology*, *464*, 388–400.

What is hydrogfd2.0? (2021). https://hypeweb.smhi.se/what-is-hydrogfd2-0/

Final_code_Additional_Thesis_Diewertje_Dekker_4592190

January 30, 2022

1 Code to produce Pobs.txt and ForceKey.txt for the HYPE model with different interpolation methods, and to analyse the results

In this Notebook, you can find the code that was produced during the Additional Thesis of Diewertje Dekker (4592190), which was done at TU Delft in cooperation with Lund University. Under supervision of Abdulghani Hasan (Lund University) and Markus Hrachowitz (TU Delft).

First you will find the code to produce a Pobs.txt and ForceKey.txt file with one out of 4 different interpolation methods to link the observations to the sub-basin centers of the model. The 4 methods that were tested were: Nearest Neighbour (NN), Inverse Distance Interpolation (IDW), Bilinear interpolation (BIL) and Ordinairy Krigging (OK). These Pobs.txt files can be used to run the HYPE model.

Secondly, you will find functions to resample the daily precipitation values to monthly values, and to compare these results with discharge observations. There is code to visualise this and to compute the RE, NSE, KGE and RE.

Everything in this notebook is run for the PO river in Italy, but the functions can be applied to any other river basin, as long as the geodata file contains all the needed information. The only thing that needs to be changed to apply it for another river basin, are the 3 data files that get loaded in: sf (shapefile), geodata, Qobs. (And in the analysis part it is specified where changes are needed when the river basin is changed.)

[1]:
```
# First load all the packages used
# from scipy.io import netcdf
import numpy as np
import matplotlib.pyplot as plt
import scipy as sc
import netCDF4 as nc
import os
import pandas as pd
import xarray as xr
import powerlaw
import matplotlib.colors as colors
import copy as copy
import datetime
from scipy.optimize import curve_fit
import scipy as sp
```

```
import scipy.optimize
import shapefile as shp  # Requires the pyshp package
import matplotlib.pyplot as plt
from scipy import signal
import matplotlib
from matplotlib.lines import Line2D
from permetrics.regression import Metrics
import array_to_latex as a2l # to easily transfer it to latex
```

1.1 Load and pre-process all the required data

[2]:
```
# Load the gridded observation data: HydroGDF2.0
path = 'C:/Users/Diewertje/Documents/Master_GRS/Thesis/Data/' # Exchange this␣
 ↪path to where you have saved the file on your computer
name_file = 'pr_GLB-0.5_GFD2.0_observation_r1i1p1_SMHI_v1_1961-2018.nc'
nc = nc.Dataset(path+name_file)
grid_data = xr.open_dataset(xr.backends.NetCDF4DataStore(nc))

# Open the GeoData.txt file. This is one of the input files for the HYPE model
# PO RIVER
curr_dir = os.getcwd()
geodata = pd.read_csv(curr_dir + '/PO_WWHYPE/PO_WWH_model/GeoData.txt',␣
 ↪delimiter='\t') # change the path to where the file is saved on your computer

# Load in the shapefile of the PO river, this is needed for plotting
# PO RIVER
sf = shp.Reader(curr_dir + "/PO_WWHYPE/PO_Shape_WWH/PO_River_307329.shp")

# Crop the gridded data to the extend of the sub-basin center (+1 degree)␣
 ↪(Takes 2-3 min)
min_lon = min(geodata['CENTERX']) -1
max_lon = max(geodata['CENTERX']) +1
min_lat = min(geodata['CENTERY']) -1
max_lat = max(geodata['CENTERY']) +1

mask_lon = (grid_data.lon >= min_lon) & (grid_data.lon <= max_lon)
mask_lat = (grid_data.lat >= min_lat) & (grid_data.lat <= max_lat)
cropped_grid_data = grid_data.where(mask_lon & mask_lat, drop=True)

# The HydroGDF2.0 dataset has a lot of large negative precipitation values.␣
 ↪These are chosen to be set to 0. An alternative solution would be to set␣
 ↪them to None.
# The np.where function replaces all values where the condition is true
cropped_grid_data['pr'] = cropped_grid_data.pr.where(cropped_grid_data.pr > 0,␣
 ↪0)

# Open the runoff dataset (Qobs.txt, an input file for the HYPE model)
```

```
Qobs = pd.read_csv(curr_dir + '/PO_WWHYPE/PO_WWH_model/Qobs.txt',␣
 ↪delimiter='\t') # Change the path to where it is saved on your computer
# Replace all negative values with 0
Qobs.iloc[:,1:] = Qobs.iloc[:,1:].where(Qobs.iloc[:,1:] > 0, np.NaN)
# The Qobs values are in m3/sec
```

Note that the GDF2.0 dataset does not have the same units as the Pobs.txt that the HYPE model takes in. Pobs should be in mm/day. GDF2.0 is in kg m^-2 s^-1.This is solved later by a correction after the interpolation method:

kg/m^2 –> 1 kg water is 1 L water, which is 1 dm^3. So 1 kg is 1 dm^3. If we spread this same volume out over an area of 1 m^2, we have a water height of: 10^-3/1 = 10^-3 m/sec of precipitation. Which is 1 mm/sec.

So the GDF2.0 values are in mm/sec. We need it in mm/day. So do a correction of *60* 60 *24. (This I do at the end of the create_Pobs function.)

1.2 Perform the interpolation and produce the Pobs.txt file

Firstly, the functions for the different interpolation methods are given, together with a few helper functions. These functions get called upon in the create_Pobs function and only there they are actually executed.

[3]:
```
##### All the helper functions #####

# ----------------------------------#
# A function to make an array with all the coordinates we are considering. (is␣
 ↪used in the create_Pobs function)
def cartesian(arrays, out=None):
    """
    Generate a cartesian product of input arrays.

    Parameters
    ----------
    arrays : list of array-like
        1-D arrays to form the cartesian product of.
    out : ndarray
        Array to place the cartesian product in.

    Returns
    -------
    out : ndarray
        2-D array of shape (M, len(arrays)) containing cartesian products
        formed of input arrays.

    Examples
    --------
    >>> cartesian(([1, 2, 3], [4, 5], [6, 7]))
    array([[1, 4, 6],
```

```
            [1, 4, 7],
            [1, 5, 6],
            [1, 5, 7],
            [2, 4, 6],
            [2, 4, 7],
            [2, 5, 6],
            [2, 5, 7],
            [3, 4, 6],
            [3, 4, 7],
            [3, 5, 6],
            [3, 5, 7]])

    """

    arrays = [np.asarray(x) for x in arrays]
    dtype = arrays[0].dtype

    n = np.prod([x.size for x in arrays])
    if out is None:
        out = np.zeros([n, len(arrays)], dtype=dtype)

    #m = n / arrays[0].size
    m = int(n / arrays[0].size)
    out[:,0] = np.repeat(arrays[0], m)
    if arrays[1:]:
        cartesian(arrays[1:], out=out[0:m, 1:])
        for j in range(1, arrays[0].size):
        #for j in xrange(1, arrays[0].size):
            out[j*m:(j+1)*m, 1:] = out[0:m, 1:]
    return out

grid_data_coord = cartesian((cropped_grid_data.lat.values, cropped_grid_data.
 ↪lon.values))
# grid_data_coord is used as input in the interpolation methods, it has first␣
 ↪lattitude and then longitude

# ----------------------------------#
def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(np.deg2rad, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
```

```
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2)**2
    c = 2 * np.arcsin(np.sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6378.1* c
    return km

# ----------------------------------#
# Detrend the whole dataset, this is needed for the krigging
def trend_data(coord, a1, a2, b1, b2,c):
    """
    coord[:,0] = latitude for all the coordinates in the array grid_data_coord
    coord[:,1] = longitude for all the coordinates in the array grid_data_coord
    """
    return a1*coord[:,0]**2+ b1*coord[:,0] + a2*coord[:,1]**2+ b2*coord[:,1] + c

def detrend_data(cropped_grid_data, grid_data_coord):
    """
    This function takes in an x-array dataframe and removes the daily spatial␣
 ↪trends in the precipitation.
    """
    # Make a copy of the data as the detrended xarray
    detrended_data = copy.deepcopy(cropped_grid_data)
    param_out = np.zeros((len(cropped_grid_data.time.values), 5))

    for i in np.arange(len(cropped_grid_data.time.values)):
        # Select a day
        pr_day = cropped_grid_data.sel(time=cropped_grid_data.time.values[i])

        # Remove spatial trends for that day
        param, _ = curve_fit(trend_data, grid_data_coord, pr_day.pr.values.
 ↪reshape(126))
        detrend_pr_day = (pr_day.pr.values.reshape(126) -␣
 ↪trend_data(grid_data_coord, param[0], param[1], param[2], param[3],␣
 ↪param[4])).reshape(9,14)

        # Plug in the daily detrended data into the xarray at the correct day.
        detrended_data['pr'] = detrended_data['pr'].where(detrended_data.time !
 ↪= cropped_grid_data.time[i], detrend_pr_day)

        # safe the param of each day in param_out, so that we can later add␣
 ↪this trend to the interpolated values.
        param_out[i,:] = param

    return detrended_data, param_out

cropped_grid_data_detrend, param_out = detrend_data(cropped_grid_data,␣
 ↪grid_data_coord)
```

```
# cropped_grid_data_detrend is used as input for the krigging
# with the use of param_out, the trend can be added again after the krigging

# ----------------------------------#
# A function to recover the trend at a certain coordinate, with the use of␣
 ↪param_out
def trend_data_out(coord, a1, a2, b1, b2,c):
    """
    input: coord[0] = latitude
           coord[1] = longitude
           a1, a2, b1, b2, c: the param_out[:,0] etc, which are the a1 for all␣
 ↪timepoints
    output: returns the trend at the coordinate of intrest for the full␣
 ↪timeseries.
    """
    return a1*coord[0]**2+ b1*coord[0] + a2*coord[1]**2+ b2*coord[1] + c

# ----------------------------------#
# The 4 model functions for the krigging
def spherical(d, n, s, r):
    ans =  n + (s - n) * (3 * d / (2 * r) - d ** 3 / ( 2 * r **3))
    ans[d==0] = 0
    ans[d >= r] =  s
    return ans

def gaussian(d, n, s, r):
    ans =  n + (s - n) * (1 - np.exp( -3 * d **2 / r**2 ))
    ans[d==0] = 0
    return ans

def exponential(d, n, s, r):
    ans = n + (s-n) * (1-np.exp(-3*d/r))
    ans[d==0] = 0
    return ans

def linear(d, n, s, r):
    ans = n+ (s-n)*(d/r)
    ans[d==0] = 0
    ans[d >= r] = s
    return ans

# ----------------------------------#
# A helper function for computing the distance matrix
def fill_upper_diag(a):
    """
    A function to compute the distance matrix between all the points.
```

```
    This funciton only fills in the diagonal, by then adding this + the␣
↪transpose together,
    the full distance matrix can be obtained.
    input: a = the array containing all the distances between all the stations
    output: out = the distance between all the stations put in the matrix above␣
↪the diagonal
    """
    n = int(np.sqrt(len(a)*2))+1
    mask = np.tri(n,dtype=bool, k=-1).T # or np.arange(n)[:,None] > np.arange(n)
    out = np.zeros((n,n),dtype=int)
    out[mask] = a
    return out
```

[4]:
```
##### The 4 different interpolation methods #####
def NN(grid_data_coord, y, x, cropped_grid_data, p=False):
    """
    Performs the Nearest Neighbor interpolation.
    This funciton you need to run for every subbasin.
    Input:  grid_data_coord = the coordinates of the gridded data set
            y = The y coordinate of the sub-basin center (lattitude)
            x = The x coordinate of the sub-basin center (longitude)
            cropped_grid_data = the gridded dataset

    Output: It returns the timeseries for the subbasin of interest.
            In this case of the NN in the gridded dataset
    """
    distances = haversine(grid_data_coord[:,1], grid_data_coord[:,0], x, y)
    nn = np.argmin(distances)
    coordNN = grid_data_coord[nn]
    return cropped_grid_data.sel(lat=coordNN[0],lon=coordNN[1]).pr

def BIL(grid_data_coord, y, x, cropped_grid_data, p=False):
    """
    Performs the Bilineair interpolation.
    This funciton you need to run for every subbasin.
    Input:  grid_data_coord = the coordinates of the gridded data set
            y = The y coordinate of the sub-basin center (lattitude)
            x = The x coordinate of the sub-basin center (longitude)
            cropped_grid_data = the gridded dataset

    Output: It returns the timeseries for the subbasin of interest.
            In this case of the NN in the gridded dataset
    """
    nn_y = np.sort(np.argpartition(np.abs(cropped_grid_data.lat.values-y), 2)[:
↪2])
    y1 = cropped_grid_data.lat.values[nn_y[0]]
    y2 = cropped_grid_data.lat.values[nn_y[1]]
```

```
    nn_x = np.sort(np.argpartition(np.abs(cropped_grid_data.lon.values-x), 2)[:
↪2])
    x1 = cropped_grid_data.lon.values[nn_x[0]]
    x2 = cropped_grid_data.lon.values[nn_x[1]]

    s = (x-x1)/(x2-x1)
    t = (y-y1)/(y2-y1)

    w1 = (1-s)*(1-t)
    w2 = s*(1-t)
    w3 = t*(1-s)
    w4 = s*t

    return w1 * cropped_grid_data.sel(lat=y1,lon=x1).pr + w2 *␣
↪cropped_grid_data.sel(lat=y1,lon=x2).pr + w3 * cropped_grid_data.
↪sel(lat=y2,lon=x1).pr + w4 * cropped_grid_data.sel(lat=y2,lon=x2).pr

def idw(grid_data_coord, y, x, cropped_grid_data, p=2):
    """
    Performs the Inverse Distance Weighted (IDW) interpolation. It can be used␣
↪inside the function: create_Pobs
    This funciton you need to run for every subbasin.
    Input:  grid_data_coord = the coordinates of the gridded data set
            y = The y coordinate of the sub-basin center (lattitude)
            x = The x coordinate of the sub-basin center (longitude)
            cropped_grid_data = the gridded dataset
            p = the distance decay parameter. p=0 is uniform, p=infinity is NN

    Output: estimate = The predicted value for every point on your grid
    """
    w = (haversine(grid_data_coord[:,1], grid_data_coord[:,0], x, y))**(-p)
    # np.sqrt((nprad[:,0] - grid_n)**2 + (nprad[:,1] - grid_e)**2)**(-p)
    estimate = 0
    for i in np.arange(len(w)):
        # Select the time series of a specific coordinate
        time_series = cropped_grid_data.
↪sel(lat=grid_data_coord[i,0],lon=grid_data_coord[i,1]).pr
        # Add the contribution of that coordinate to the estimate
        estimate = (time_series* w[i]) + estimate
    return estimate/np.sum(w) # Devide by the sum of w no normalise the␣
↪contributions.

def krig_prep(grid_data_coord, y, x, cropped_grid_data_detrend, p=2):
    """
    Performs the preparation for the Krigging interpolation. It can be used␣
↪inside the function: create_Pobs
```

```
   This funciton only need to be run ones, so not separately for all subbasins
   Input:  grid_data_coord = the coordinates of the gridded data set
           y = The y coordinate of the sub-basin center (lattitude)
           x = The x coordinate of the sub-basin center (longitude)
           cropped_grid_data_detrend = the detrended gridded dataset
           p = the distance decay parameter. p=0 is uniform, p=infinity is NN,␣
↪not used in krigging

   Output: It returns the variables that the krigging needs: times, invG_out,␣
↪nsr_out, best_model_out

   Note that the choice that were made for this function are:
   * For the PO river, the distance at which to cut the variogram was 450 km.
   """
   t0 = datetime.datetime.now()
   # Now start to compute the variogram
   n = grid_data_coord.shape[0] # how many coordinates we have
   dist = np.zeros(int(n * ( n-1) / 2)) # the distance between all the␣
↪gridpoints
   times = cropped_grid_data_detrend.time.shape[0] # of how many timepoints we␣
↪have data
   s2 = np.zeros((int(n * ( n-1) / 2), times))

   ii = 0
   for ind1 in np.arange(n):
       for ind2 in np.arange(ind1+1,n):
           dist[ii] = haversine(grid_data_coord[ind1,1],␣
↪grid_data_coord[ind1,0], grid_data_coord[ind2,1], grid_data_coord[ind2,0])
           val1 = cropped_grid_data_detrend.sel(lon=grid_data_coord[ind1,1],␣
↪lat=grid_data_coord[ind1,0])
           val2 = cropped_grid_data_detrend.sel(lon=grid_data_coord[ind2,1],␣
↪lat=grid_data_coord[ind2,0])
           s2[ii,:]=(val1-val2).pr.values**2/2
           ii=ii+1

   N = 17 # number of bins (chosen so that no n_bin value became 0)
   bins = np.linspace(0,np.max(dist), N+1)
   deltad = bins[1]-bins[0]

   s_mean = np.zeros((N,times))
   bin_mean = bins[:-1] + deltad/2
   n_bin = np.zeros(N)

   for jj in range(N):
       idx0 = (dist >= bins[jj])
       idx1 = (dist < bins[jj+1])
```

```
        idx = idx1 * idx0
        if np.sum(idx) == 0:
            continue
        n_bin[jj] = np.sum(idx)
        s_mean[jj,:] = s2[idx].mean(axis=0) # s_mean now contains the mean s␣
↪values in every bin (17) for all timepoints (all days or months)

    # Cut off the variogram
    idx_verio = bin_mean < 450 # The distance at which the histogram height␣
↪falls to halve the maximum for PO area!
    dist_bm = bin_mean[idx_verio]
    gamma = s_mean[idx_verio]
    n_d = n_bin[idx_verio]

    w = n_d / (dist_bm ** 2)
    w = w / np.linalg.norm(w)
    sigma = np.diag(1/w)

    dt = datetime.datetime.now() - t0
    print('It took {} seconds to do the first part of the preparation for the␣
↪krigging.'.format(dt.total_seconds()))

    print('start fitting')
    t0 = datetime.datetime.now()
    for i in np.arange(times):
        # Try all 4 types of fits and see which one is best, continue with that␣
↪one
        try:
            popt_sph = sp.optimize.curve_fit(spherical, dist_bm, gamma[:,i], p0␣
↪= [0, 450, np.max(dist_bm)], sigma = sigma)
            n_sph, s_sph, r_sph = popt_sph[0]
            ver_sph = spherical(dist_bm, n_sph, s_sph, r_sph)
            error_model = [np.mean((ver_sph-gamma[:,i])**2)]
        except:
            error_model = [99999]

        try:
            popt_gauss = sp.optimize.curve_fit(gaussian, dist_bm, gamma[:,i],␣
↪p0 = [0, 450, np.max(dist_bm)], sigma = sigma)
            n_gauss, s_gauss, r_gauss = popt_gauss[0]
            ver_gauss = gaussian(dist_bm, n_gauss, s_gauss, r_gauss)
            error_model.append(np.mean((ver_gauss-gamma[:,i])**2))
        except:
            error_model.append(9999)

        try:
```

```
            popt_exp = sp.optimize.curve_fit(exponential, dist_bm, gamma[:,i],␣
↪p0 = [0, 450, np.max(dist_bm)], sigma = sigma)
            n_exp, s_exp, r_exp = popt_exp[0]
            ver_exp = exponential(dist_bm, n_exp, s_exp, r_exp)
            error_model.append(np.mean((ver_exp-gamma[:,i])**2))
        except:
            error_model.append(9999)

        try:
            popt_lin = sp.optimize.curve_fit(linear, dist_bm, gamma[:,i], p0 =␣
↪[0, 450, np.max(dist_bm)], sigma = sigma)
            n_lin, s_lin, r_lin = popt_lin[0]
            ver_lin = linear(dist_bm, n_lin, s_lin, r_lin)
            error_model.append(np.mean((ver_lin-gamma[:,i])**2))
        except:
            error_model.append(9999)

        # From error_model, find the best model
        best_model = [spherical, gaussian, exponential, linear][np.
↪argmin(error_model)]
        best_nsr = ['sph', 'gauss', 'exp', 'lin'][np.argmin(error_model)]

        # Compute matrix G, the full OK matrix, corresponding to the␣
↪trainingdata
        lt = fill_upper_diag(dist)
        D = lt + lt.T                           # Distance matrix
        D = D.astype(np.float)                  # turn the elements of D into␣
↪floats, so that the computer can deal with the large numbers
        G = best_model(D, locals()['n_'+best_nsr], locals()['s_'+best_nsr],␣
↪locals()['r_'+best_nsr]) # compute matrix G, based on D and the variogram␣
↪function obtained in the question before

        # Add the row and column with ones, and a 0 at the last element.
        newrow = np.ones(n)
        newcolumn = np.ones((n + 1,1))
        newcolumn[-1] = 0
        G = np.vstack([G, newrow])
        G = np.hstack([G, newcolumn])

        invG=np.linalg.inv(G)

        if i == 0:
            invG_out = np.zeros((times, G.shape[0], G.shape[1]))
            nsr_out = np.zeros((3,times)) # n,s,r saved in this order.
            best_model_out = []
```

```
        invG_out[i,:,:] = invG
        nsr_out[0,i] = locals()['n_'+best_nsr]
        nsr_out[1,i] = locals()['s_'+best_nsr]
        nsr_out[2,i] = locals()['r_'+best_nsr]
        best_model_out.append(best_model)

        if i in np.arange(times, step=2500):
            print(f'Done with {i} timepoints, time passed: {datetime.datetime.
 ↪now()-t0}')

    dt = datetime.datetime.now() - t0
    print('It took {} seconds to complete the preparation for the krigging.'.
 ↪format(dt.total_seconds()))
    return times, invG_out, nsr_out, best_model_out # this returns the G, n,s,r␣
 ↪for all time points for this PO basin. this is needed as input for the next␣
 ↪function that gets ran per basin.

# Run the preparation for the krigging now, so that the input variables for␣
 ↪krig are known.
print('Preparation krigging starts')
times, invG_out, nsr_out, best_model_out =␣
 ↪krig_prep(grid_data_coord=grid_data_coord, y=0, x=0,␣
 ↪cropped_grid_data_detrend=cropped_grid_data_detrend) # x and y are not used␣
 ↪in here yet
print('Preperation krigging is done')

def krig(grid_data_coord, y, x, cropped_grid_data_detrend, p=2, times=times,␣
 ↪invG_out=invG_out, nsr_out=nsr_out, best_model_out=best_model_out):
    """
    Performs the Krigging interpolation. It can be used inside the function:␣
 ↪create_Pobs
    This funciton you need to run for every subbasin.
    Input:  grid_data_coord = the coordinates of the gridded data set
            y = The y coordinate of the sub-basin center (lattitude)
            x = The x coordinate of the sub-basin center (longitude)
            cropped_grid_data_detrend = the detrended gridded dataset
            p = the distance decay parameter. p=0 is uniform, p=infinity is NN,␣
 ↪not used in krigging
            times, dist_bm, gamma, sigma = all variables delivered by the␣
 ↪krig_prep.

    Output: est = It returns the timeseries for the subbasin of interest.

    Note the choices that were made for this function are:
    * For the PO river, the distance at which to cut the variogram was 450 km.
    * Which model is used gets determined per day.
```

```
    """
    est = cropped_grid_data_detrend.
↪sel(lat=grid_data_coord[0,0],lon=grid_data_coord[0,1]).pr # use this instead␣
↪of np.zeros, since now the output is an xarray and that is what we need.
    for i in np.arange(times):
        dist_basin = haversine(grid_data_coord[:,1], grid_data_coord[:,0], x, y)

        G0 = best_model_out[i](dist_basin, nsr_out[0,i], nsr_out[1,i],␣
↪nsr_out[2,i])
        G0 = np.hstack([G0,1])
        lapda = invG_out[i,:,:] @ G0
        est[i]= np.sum(lapda[:-1] * cropped_grid_data_detrend.
↪sel(time=cropped_grid_data_detrend.time.values[i]).pr.values.reshape(1,126))
    # Now only add the trend back to the estimates.
    est = est + trend_data_out(np.array([y,x]), param_out[:,0], param_out[:,1],␣
↪param_out[:,2], param_out[:,3], param_out[:,4])
    # Set all the negative precipitation values to 0
    est = est.where(est>=0, 0)
    est_NN = NN(grid_data_coord, y, x, cropped_grid_data, p=False)
    est = est.where(est<0.0023, est_NN)
    # Due to a (unknown) numerical issue in the krigging the values sometimes␣
↪become unreasonably large.
    # Therefore I have chosen to set all values that result in more than 200 mm␣
↪rain (max of Pobs_NN is 185 and 200/(60*60*24)=0.0023),
    # to the NN estimate. This is something that can be further investigated in␣
↪the future.
    return est
```

```
Preparation krigging starts
It took 28.71638 seconds to do the first part of the preparation for the
krigging.
start fitting
Done with 0 timepoints, time passed: 0:00:01.112752

<ipython-input-3-892d02a8531c>:139: RuntimeWarning: overflow encountered in exp
  ans = n + (s-n) * (1-np.exp(-3*d/r))
C:\Users\Diewertje\Python\envs\geo_env\lib\site-
packages\scipy\optimize\minpack.py:794: OptimizeWarning: Covariance of the
parameters could not be estimated
  warnings.warn('Covariance of the parameters could not be estimated',

Done with 2500 timepoints, time passed: 0:01:25.826220
Done with 5000 timepoints, time passed: 0:02:50.303473
Done with 7500 timepoints, time passed: 0:04:14.964603
Done with 10000 timepoints, time passed: 0:05:38.957412
Done with 12500 timepoints, time passed: 0:07:04.633844
Done with 15000 timepoints, time passed: 0:08:30.311640
Done with 17500 timepoints, time passed: 0:09:55.664164
```

```
Done with 20000 timepoints, time passed: 0:11:20.240379
It took 718.005228 seconds to complete the preparation for the krigging.
Preperation krigging is done
```

Choises that are made in the function for the krigging are: * The number of bins is chosen as 17, since this was the lowest number without any empty bins. * Analysing the histogram for the PO river, show that the max freq is reached at around 300 km and this frequency is halved at 450 km. The 450 km is chosen as the cut-off of the variogram, also since the variogram started to flatten at 450 km for most days. Note that the variogram is not checked for every day, so for some it could obtain a better fit with a different cutoff. This is something that can be further investigated in future research. * All 4 models are fitted through te variogram cloud for every day. The model with the lowest Mean Squared Error (MSE) is selected as the best and used for estimation at the sub-basin centers for that day. * Note that the krigging interpolation does result in negative precipitation values. It is chosen to set these values to 0. * Note that the krigging interpolation also results in unreasonably large values sometimes. Why this occurs is unknown. But it is chosen to set these values to the Nearest Neighbour estimation instead. This is done for all precipitation values above 200.

[5]:
```
# The funciton to create Pobs
def create_Pobs(int_meth, cropped_grid_data, geodata, p=2):
    grid_data_coord = cartesian((cropped_grid_data.lat.values,␣
 ↪cropped_grid_data.lon.values))

    for i in np.arange(geodata.shape[0]):
        # Do the interpolation for 1 sub-basin and find its timeseries
        time_series = int_meth(grid_data_coord,␣
 ↪geodata['CENTERY'][i],geodata['CENTERX'][i], cropped_grid_data, p=p)

        if i==0: # Initialise Pobs by making the column with DATE
            day = (time_series.time.dt.day.values).astype(str)
            for j in np.arange(len(day)):
                if len(day[j])==1:
                    day[j]='0'+day[j]

            month = (time_series.time.dt.month.values).astype(str)
            for j in np.arange(len(month)):
                if len(month[j])==1:
                    month[j]='0'+month[j]
            date = np.char.add(np.char.add((time_series.time.dt.year.values).
 ↪astype(str), month),day)
            Pobs = pd.DataFrame(date, columns=['DATE'])

        # Add a column for every sub-basin center, so for every subID
        Pobs.insert(1, str(geodata['SUBID'][i]), time_series.values*60*60*24) #␣
 ↪also convert to correct units
        if np.sum(np.sum(Pobs == np.inf)) != 0:
            print(f'for i ={i} we get a value of infinity.')
            break
```

```
    Pobs.to_csv('Pobs_'+int_meth.__name__+'.txt', sep='\t', index=False)
    return Pobs
```

[6]:
```
# Run this cell to actually execute above functions
Pobs_NN = create_Pobs(NN, cropped_grid_data, geodata)
Pobs_idw = create_Pobs(idw, cropped_grid_data, geodata)
Pobs_bil = create_Pobs(BIL, cropped_grid_data, geodata)

t0 = datetime.datetime.now()
Pobs_krig = create_Pobs(krig, cropped_grid_data_detrend, geodata)
dt = datetime.datetime.now() - t0
print('It took {} seconds to compute the timeseries for all subbasins with␣
 ↪krigging.'.format(dt.total_seconds()))


# To see the influence of the distance decay parameter in IDW, run the␣
 ↪following lines and plot the figure that is commented out in a few cells␣
 ↪from here
# Pobs_idw1 = create_Pobs(idw, cropped_grid_data, geodata, p=1)
# Pobs_idw3 = create_Pobs(idw, cropped_grid_data, geodata, p=3)
```

```
It took 3599.401615 seconds to compute the timeseries for all subbasins with
krigging.
```

1.3 Create the ForceKey.txt file

[ ]:
```
# First load the existing force key from the current HYPE model
path_to_force = 'C:/Users/Diewertje/Documents/Master_GRS/Thesis/
 ↪Intro_files_Abdulghani/PO/PO/PO_WWHYPE/PO_WWH_model/'
ForcKey = pd.read_csv(path_to_force+'ForcKey.txt', sep='\t')

# Now change the POBSID to the SUBID, since the Pobs we have produced has a␣
 ↪Pobs for every sub-id
ForcKey['POBSID'] = ForcKey['SUBID']

# And safe the file
ForcKey.to_csv('ForcKey.txt', sep='\t', index=False)
```

1.4 Functions to plot the Pobs from different interpolation methods, to be able to compare visually for different time periods

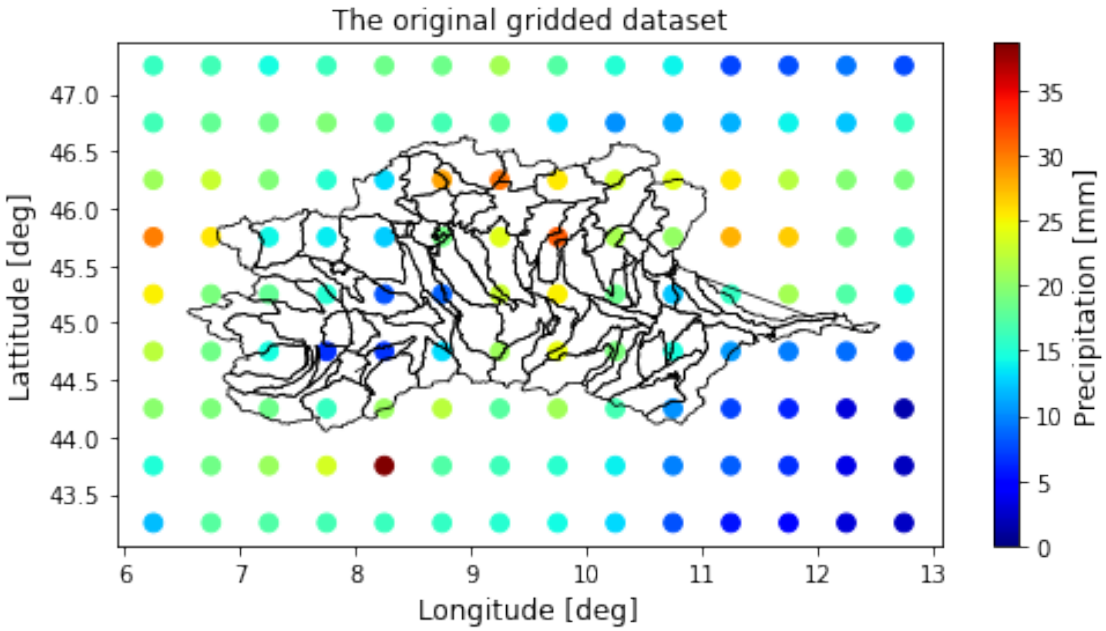
[7]:
```
def plot_grid(sf, cropped_grid_data, time_index):
    '''
    This funciton plots the original gridded data with the subbasin borders on␣
 ↪the background.

    sf = shape file
```

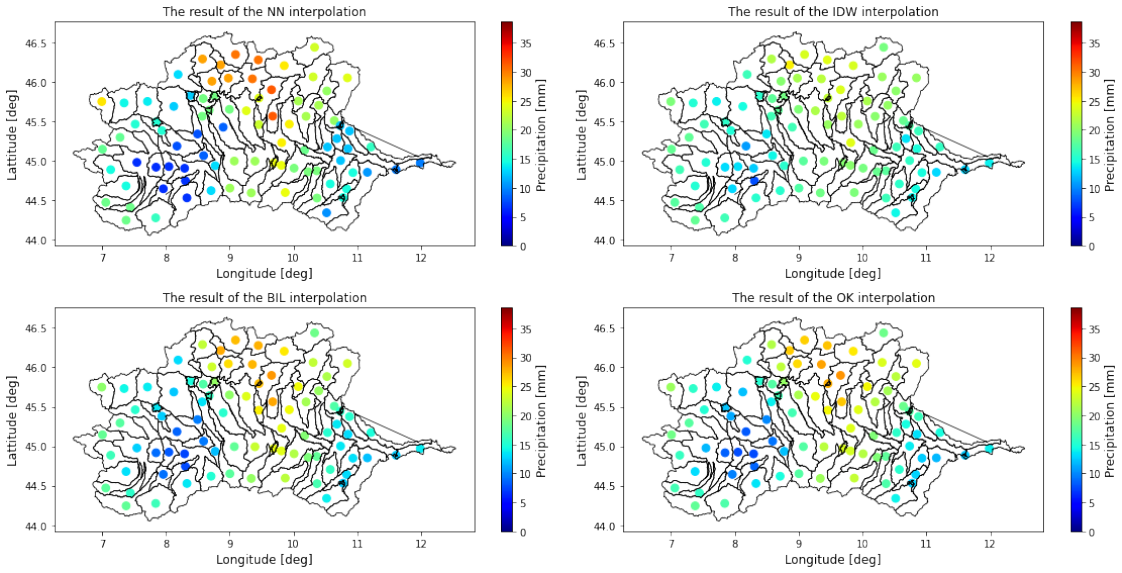
```
    time_index = A list with the rows in the Pobs that you want to depict.
                 This is a single day in the time-series,
                 or multiple days and then the average is taken.
                 Can be any value between 0 and 20938.
    '''
    # Compute the average grid over the time period
    time_sel0 = cropped_grid_data.time.values[0] # the 0 is the time index??␣
↪check this
    dim = (cropped_grid_data.sel(time=time_sel0).pr).shape # the lat+lon␣
↪dimensions
    grid = np.zeros((len(time_index),dim[0], dim[1]))
    for ii in np.arange(len(time_index)):
        time_sel = cropped_grid_data.time.values[time_index[ii]]
        grid[ii,:] = cropped_grid_data.sel(time=time_sel).pr.values
    grid = np.sum(grid, axis=0)*60*60*24 # also convert to correct units of mm/
↪day instead of mm/s

    # Plot the subbasin outlines
    for shape in sf.shapeRecords():
        x = [i[0] for i in shape.shape.points[:]]
        y = [i[1] for i in shape.shape.points[:]]
        plt.plot(x,y, color='black', linewidth=0.7)

    # Plot the differences at the subbasin centers
    plt.scatter(grid_data_coord[:,1], grid_data_coord[:,0], c=grid ,cmap='jet',␣
↪s=60, vmin=0, vmax=np.max(grid))#,norm=colors.PowerNorm(0.7,vmin=0,vmax=np.
↪max(grid)))
    plt.colorbar().set_label('Precipitation [mm]', fontsize=12)
    plt.title(f'The original gridded dataset')#, timeindex={time_index}')#,␣
↪fontsize=20)
    plt.xlabel('Longitude [deg]', fontsize=12)
    plt.ylabel('Lattitude [deg]', fontsize=12)
    return np.max(grid)

def plot_int_meth(sf, int_meth, time_index, title_method, max_grid=False):
    '''
    This funciton This funciton plots the result of one of the interpolation␣
↪methods
    with the subbasin borders on the background.

    sf = shape file
    int_meth = the result of the interpolation method, so for example Pobs_NN
    time_index = A list with the rows in the Pobs that you want to depict.
                 This is a single day in the time-series,
                 or multiple days and then the average is taken.
                 Can be any value between 0 and 20938.
```

```
    '''
    plt_int = np.zeros((len(time_index),len(int_meth.loc[time_index[0]].
↪values[1:])))
    for ii in np.arange(len(time_index)):
        plt_int[ii,:] = (int_meth.loc[time_index[ii]].values[1:])[::-1]
        # The [::-1] is to reverse the numbers, so that they are in the correct␣
↪order of the subIDs
    plt_int = np.sum(plt_int, axis=0) # THIS WAS np.mean before
    if max_grid == False:
        max_grid = np.max(plt_int)

    # Plot the subbasin outlines
    for shape in sf.shapeRecords():
        x = [i[0] for i in shape.shape.points[:]]
        y = [i[1] for i in shape.shape.points[:]]
        plt.plot(x,y, color='black', linewidth=0.7)

    # Plot the differences at the subbasin centers
    plt.scatter(geodata['CENTERX'], geodata['CENTERY'], c=plt_int ,cmap='jet',␣
↪s=60, vmin=0, vmax=max_grid)
    plt.colorbar().set_label('Precipitation [mm]', fontsize=12)
    plt.title(f'The result of the {title_method} interpolation')#,␣
↪timeindex={time_index}')#, fontsize=20)
    plt.xlabel('Longitude [deg]', fontsize=12)
    plt.ylabel('Lattitude [deg]', fontsize=12)
```

[8]:
```
"""
To produce an image for a certain month or multiple months,
this can be changed by changing the time_index,
when you take multiple months in the list,
it depicts the sum of them.
To produce image in the report: time_index 2
"""
time_index=[2]

plt.figure(figsize=(8,4))
max_grid = plot_grid(sf,cropped_grid_data,time_index)

plt.figure(figsize=(16,8))
plt.title('The precipitation in 1 month')
plt.subplot(2,2,1)
plot_int_meth(sf,Pobs_NN,time_index, 'NN', max_grid)
plt.subplot(2,2,2)
plot_int_meth(sf,Pobs_idw,time_index, 'IDW', max_grid)
plt.subplot(2,2,3)
plot_int_meth(sf,Pobs_bil,time_index, 'BIL', max_grid)
plt.subplot(2,2,4)
```

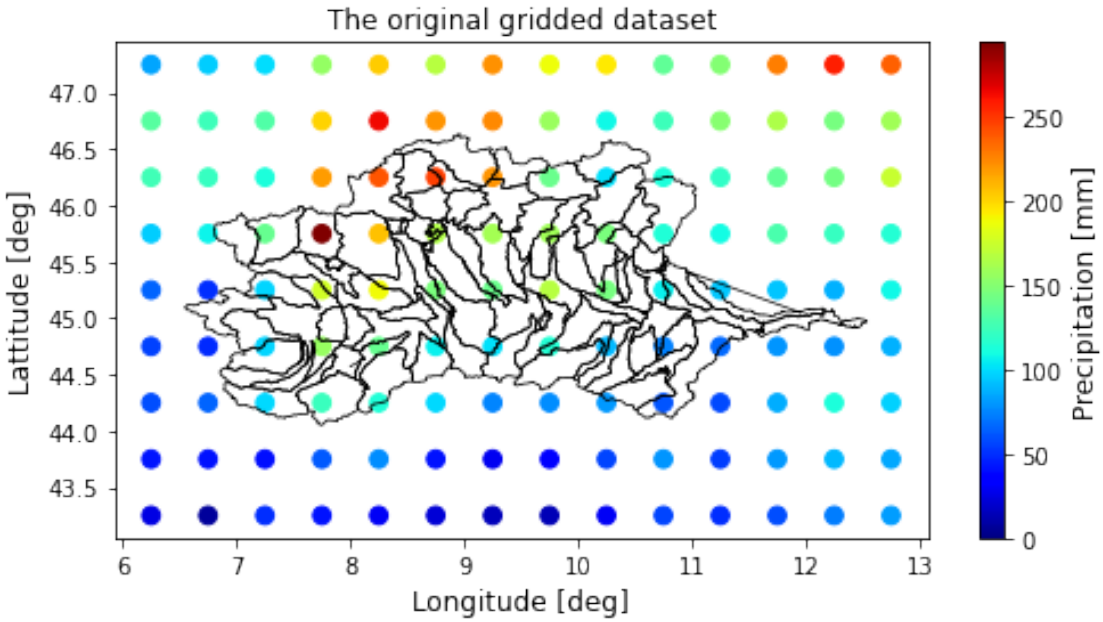
```
plot_int_meth(sf,Pobs_krig,time_index, 'OK', max_grid)

plt.tight_layout()
```

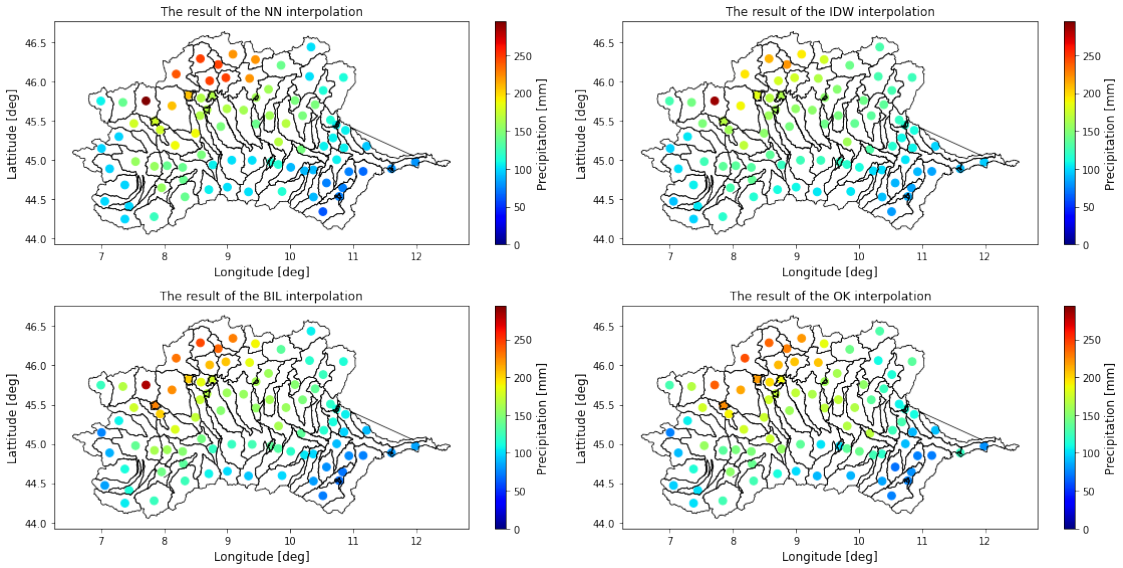




[22]:
```
time_index=np.arange(31)+31+28+31+30+31+30+365*41 # july 2001

plt.figure(figsize=(8,4))
```

```
max_grid = plot_grid(sf,cropped_grid_data,time_index)

plt.figure(figsize=(16,8))
plt.title('The precipitation in 1 month')
plt.subplot(2,2,1)
plot_int_meth(sf,Pobs_NN,time_index, 'NN', max_grid)
plt.subplot(2,2,2)
plot_int_meth(sf,Pobs_idw,time_index, 'IDW', max_grid)
plt.subplot(2,2,3)
plot_int_meth(sf,Pobs_bil,time_index, 'BIL', max_grid)
plt.subplot(2,2,4)
plot_int_meth(sf,Pobs_krig,time_index, 'OK', max_grid)

plt.tight_layout()
```





[ ]:
```
# # Create figure for report with different values of P in the IDW interpolation
# # NOTE: if you want to run this cell, you do need to run the different idw options in the Pobs cell.
# time_index=[2]

# plt.figure(figsize=(20,4))
# plt.subplot(1,3,1)
# plot_int_meth(sf,Pobs_idw1,time_index, 'IDW', max_grid)
# plt.title('p = 1')
# plt.subplot(1,3,2)
# plot_int_meth(sf,Pobs_idw,time_index, 'IDW', max_grid)
# plt.title('p = 2')
# plt.subplot(1,3,3)
# plot_int_meth(sf,Pobs_idw3,time_index, 'IDW', max_grid)
# plt.title('p = 3')

# plt.tight_layout()
```

1.5 To compare the results with the measured discharge

[9]:
```
#### First add some information to the geodata ####

# First add an extra row to the geodata so that we know what the upstream sub-basin is of the basins with a Qobs
geodata['MAINUP']=np.zeros(geodata.shape[0])
for Qobs_intrest in Qobs.columns[1:].astype(int):
    maindown = geodata.loc[geodata['SUBID']==Qobs_intrest]['MAINDOWN'].values[0] # look what the main down is for the Qobs of interest
```

```
    geodata.loc[geodata['SUBID']==maindown,['MAINUP']]=Qobs_intrest # assign␣
↪the value of the Qobs of interest as the main up to the subid of the main␣
↪down

# Also add an extra column to geodata to identify which sub-basins are the Qobs
geodata['NOT_Qobs']=np.ones(len(geodata))
for i in Qobs.columns[1:].to_list():
    geodata.loc[geodata['SUBID']==int(i),['NOT_Qobs']]=0

# Add a group index behind every subbasin, so they are grouped to a Qobs
geodata['GROUPED'] = np.zeros(geodata.shape[0])
group_id=1
for name in Qobs.columns[1:].to_list():
    geodata.loc[geodata['MAINDOWN']==int(name),['GROUPED']]=group_id
    upstreams = geodata.loc[geodata['MAINDOWN']==int(name)]['SUBID'].values
    while upstreams.size != 0:
        geodata.loc[(geodata['MAINDOWN'] == upstreams[0]) &␣
↪(geodata['NOT_Qobs']==1), ['GROUPED']]=group_id # Set the GROUPED indicator␣
↪for thse upstream sub-basins to the same number (only the ones that are not␣
↪the next Qobs)
        upstreams = np.concatenate([upstreams, geodata[(geodata['MAINDOWN'] ==␣
↪upstreams[0]) & (geodata['NOT_Qobs']==1)]['SUBID'].values]) # if it is not a␣
↪Qobs basin, add the sub-id to the upstream list
        upstreams=upstreams[1:] # Delete the upstream you looked at now.
    group_id=group_id+1


#### Functions to create volume datasets and resample them to monthly datasets.
↪###
def create_Volume(Pobs, geodata):
    """
    This function calculates the volume in every subbasin based on the Pobs␣
↪file.

    Pobs = the earlier produced Pobs dataframe
    geodata = the geodata file in the dataframe

    The Pobs is in mm/day. The Vobs we want in m^3/sec, since that is what the␣
↪Qobs is given in.
    Area is given in m^2. , so to transfer to right units: *10^-3/(24*60*60)
    """
    Volume = copy.deepcopy(Pobs)
    subid_names = Pobs.columns.values.tolist()[1:]
    for i in subid_names:
        Volume[i] = Pobs[i]*geodata.loc[geodata['SUBID'] == np.int(i)]['AREA'].
↪values *10**-3/(24*60*60)
```

```
    return Volume

def make_monthly_dataset(Vobs):
    """
    This function resamples the daily Vobs or Pobs dataframe into a montly one,␣
 ↪which is the sum of all days.
    """
    Vobs_new = copy.deepcopy(Vobs)
    for i in np.arange(Vobs.shape[0]):
        Vobs_new['DATE'][i] = datetime.datetime.
 ↪strptime(Vobs['DATE'][i],"%Y%m%d")
    Vobs_new = Vobs_new.set_index('DATE')
    Vobs_new = Vobs_new.resample('M').sum()

    Vobs_new.loc[len(Vobs_new)] = geodata['MAINUP'].values.astype(str)[::-1]
    return Vobs_new

def make_monthly_Qobs(Vobs):
    """
    This function resamples the daily Qobs dataframe into a monthly one.
    """
    Vobs_new = copy.deepcopy(Vobs)
    for i in np.arange(Vobs.shape[0]):
        Vobs_new['DATE'][i] = datetime.datetime.
 ↪strptime(Vobs['DATE'][i],"%Y-%m-%d")
    Vobs_new = Vobs_new.set_index('DATE')
    return Vobs_new.resample('M').agg(pd.Series.sum, skipna=False)

def group_Vobs(Vobs, geodata):
    """
    This function groups the subbasins together to get the added volume in all␣
 ↪the
    subbasins for which there is one measurement in the Qobs (discharge).
    """
    Vobs_grouped=copy.deepcopy(Vobs)
    count = 0

    for i in np.arange(geodata.shape[0]):
        if not str(geodata['SUBID'][i]) in Qobs.columns.values.tolist()[1:]:
            count=count+1
            # to only do the following for if the subid is not in the Qobs
            if geodata['MAINDOWN'][i] != -9999:
                Vobs_grouped[str(geodata['MAINDOWN'][i])] =␣
 ↪Vobs_grouped[str(geodata['MAINDOWN'][i])] +␣
 ↪Vobs_grouped[str(geodata['SUBID'][i])]
                Vobs_grouped = Vobs_grouped.
 ↪drop(columns=str(geodata['SUBID'][i]))
```

```
    # Now use the last row that was added while making the monthly dataset, to␣
↪add the upstream volumes
    for i in np.arange(Vobs_grouped.shape[1]):
        list_upstream = Vobs_grouped.loc[len(Vobs_grouped)-1][i].split('.0')
        list_upstream = list(filter(None, list_upstream)) # remove all empty␣
↪entries in the list
        list_upstream = list(filter(lambda num: num != '0', list_upstream)) #␣
↪remove all 0 in list
        # save this list in the Vobs_grouped as an extra row
        Vobs_grouped.iat[len(Vobs_grouped)-1,i] = ','.join(list_upstream)

    # Now also add the areas upstream of the upstream area to that list, so␣
↪that we get the full upstream area
    for i in np.arange(Vobs_grouped.shape[1]): # Loop through all Qobs subbasins
        new_name = Vobs_grouped.iat[len(Vobs_grouped)-1,i].split(',') # Find␣
↪all the 1st order upstream areas for this Qobs subbasin
        length_original = len(new_name)
        count = 1
        while new_name != [''] and len(new_name)!=0:
            added = new_name[0]
            if count > length_original:
                Vobs_grouped.iat[len(Vobs_grouped)-1,i] = Vobs_grouped.
↪iat[len(Vobs_grouped)-1,i] +','+ added # Add the upstream name to Vobs last␣
↪row
            new_name.extend(Vobs_grouped[added].iat[len(Vobs_grouped)-1].
↪split(',')) # Add the 2nd order upstream names to the list
            new_name.pop(0) # Delete the name that is already added to the Vobs
            new_name = list(filter(None, new_name)) # remove all empty entries␣
↪in the list
            count = count+1

        # Now add all these volumes to this Qobs subbasin
        if Vobs_grouped.iat[len(Vobs_grouped)-1,i].split(',') != ['']:
            for vol_to_add in Vobs_grouped.iat[len(Vobs_grouped)-1,i].
↪split(','):
                Vobs_grouped.iloc[:len(Vobs_grouped)-1,i] = Vobs_grouped.iloc[:
↪len(Vobs_grouped)-1,i]+ Vobs_grouped[vol_to_add]


    return Vobs_grouped
    # Here we have one group more than in Qobs, since there is no observation␣
↪for the entry to the sea, so 307329 we have extra.
```

[10]:
```
# Run the functions defined above for NN Pobs
V_NN = create_Volume(Pobs_NN, geodata)
```

```
VNN_month = make_monthly_dataset(V_NN)
VNN_month_group = group_Vobs(VNN_month, geodata)

# Run the functions defined above for idw Pobs
V_idw = create_Volume(Pobs_idw, geodata)
Vidw_month = make_monthly_dataset(V_idw)
Vidw_month_group = group_Vobs(Vidw_month, geodata)

# Run the functions defined above for BIL Pobs
V_bil = create_Volume(Pobs_bil, geodata)
Vbil_month = make_monthly_dataset(V_bil)
Vbil_month_group = group_Vobs(Vbil_month, geodata)

# Run the functions defined above for KRIG Pobs
V_krig = create_Volume(Pobs_krig, geodata)
Vkrig_month = make_monthly_dataset(V_krig)
Vkrig_month_group = group_Vobs(Vkrig_month, geodata)

# Resample the Discharge oversvations to monthly ones
Qobs_mon = make_monthly_Qobs(Qobs)
```

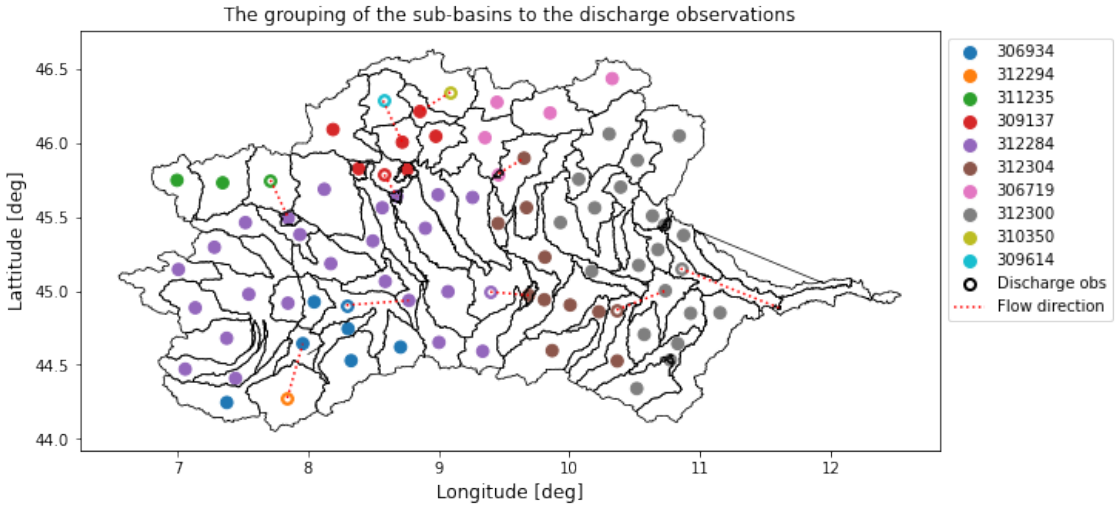
```
<ipython-input-9-9d52bab25e74>:50: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  Vobs_new['DATE'][i] = datetime.datetime.strptime(Vobs['DATE'][i],"%Y%m%d")
C:\Users\Diewertje\Python\envs\geo_env\lib\site-
packages\pandas\core\indexes\base.py:3331: RuntimeWarning: '<' not supported
between instances of 'int' and 'Timestamp', sort order is undefined for
incomparable objects
  join_index = self.union(other)
<ipython-input-9-9d52bab25e74>:63: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  Vobs_new['DATE'][i] = datetime.datetime.strptime(Vobs['DATE'][i],"%Y-%m-%d")
```

First visualise the grouping of the sub-basins to the discharge observation, this image is also used in the additional thesis

[11]:
```
plt.figure(figsize=(10,5))

# Plot the subbasin outlines
for shape in sf.shapeRecords():
    x = [i[0] for i in shape.shape.points[:]]
    y = [i[1] for i in shape.shape.points[:]]
```

```
    plt.plot(x,y, color='black', linewidth=0.7)

# Plot the differences at the subbasin centers
for i in np.arange(8)+1:
    sel_sub = geodata[geodata['GROUPED']==i]
    plt.scatter(sel_sub['CENTERX'], sel_sub['CENTERY'], c=np.
 ↪ones(len(sel_sub))*(i-1) ,cmap='tab10', s=60, vmin=0, vmax=10)

i=0
for name in Qobs.columns[1:].to_list():
    sel_sub = geodata[geodata['SUBID']==int(name)]
    plt.scatter(sel_sub['CENTERX'], sel_sub['CENTERY'], c=i ,cmap='tab10',␣
 ↪s=60, vmin=0, vmax=10)#, label=name)
    plt.scatter(sel_sub['CENTERX'], sel_sub['CENTERY'], c=0 ,cmap='binary',␣
 ↪s=10, vmin=0, vmax=10)
    main_down = geodata.loc[geodata['SUBID']==int(sel_sub['MAINDOWN'].
 ↪values[0])]
    plt.plot(np.array([sel_sub['CENTERX'].values, main_down['CENTERX'].values]),
             np.array([sel_sub['CENTERY'].values, main_down['CENTERY'].
 ↪values]), color='red', linestyle=':')
    i=i+1

# plt.colorbar().set_label('Precipitation [mm]', fontsize=12)
plt.title(f'The grouping of the sub-basins to the discharge observations')#,␣
 ↪timeindex={time_index}')#, fontsize=20)
plt.xlabel('Longitude [deg]', fontsize=12)
plt.ylabel('Lattitude [deg]', fontsize=12)

# Compute the legend manually (since automatically it gives wrong colors)
colors_plot = matplotlib.cm.get_cmap('tab10')
legend_elements = []

i=0
for name in Qobs.columns[1:].to_list():
    legend_elements.append(Line2D([0], [0], marker='o', color='w', label=name,
                          markerfacecolor=colors_plot(i/len(Qobs.columns[1:].
 ↪to_list())), markersize=10))
    i=i+1
legend_elements.append(Line2D([0], [0], marker='o', color='w', label=␣
 ↪'Discharge obs', markerfacecolor='w', mec='black', mew=2, markersize=7))
legend_elements.append(Line2D([0], [0], linestyle=':', color='red', label='Flow␣
 ↪direction'))
plt.legend(handles=legend_elements, bbox_to_anchor=(1,1), loc="upper left")
```

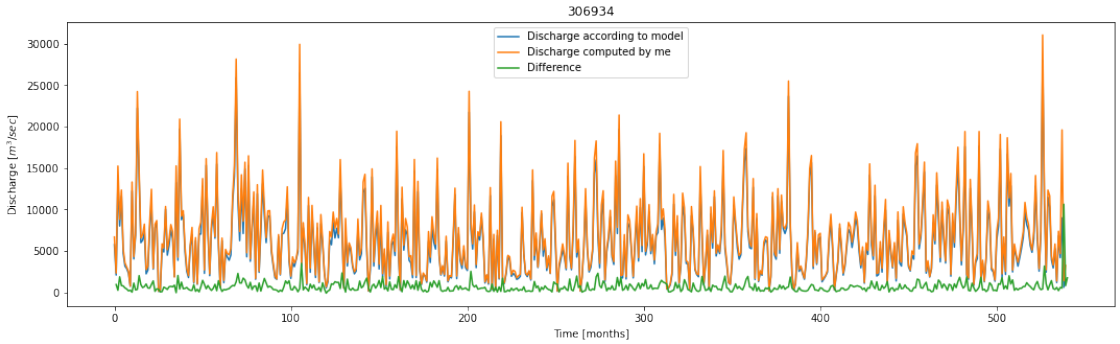
[11]: <matplotlib.legend.Legend at 0x2393db7d2e0>

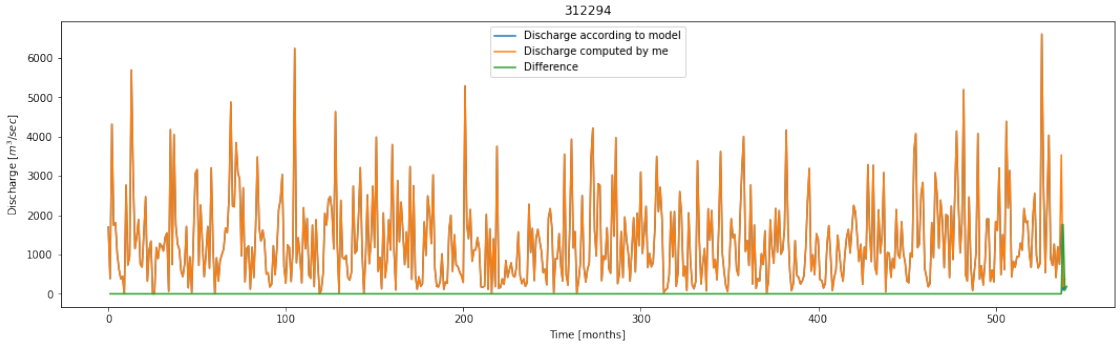


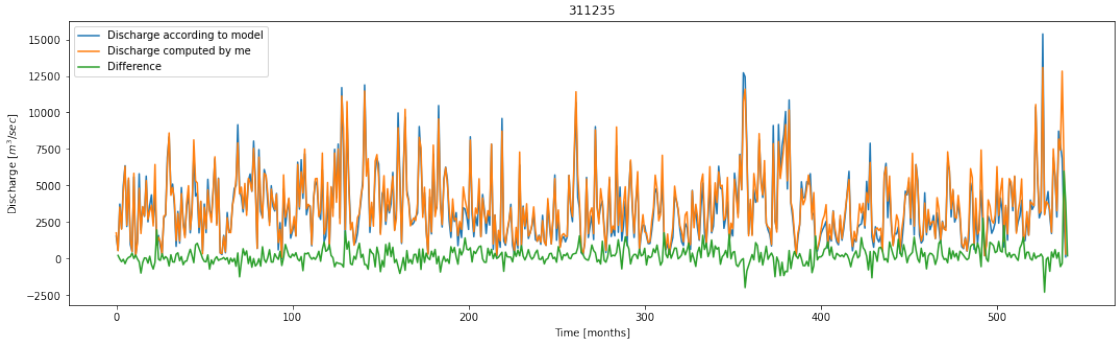
Compare the computed discharge from my calculations for NN with the output from the HYPE model, this is done as a check to see if the calculations are performed correctly To do this, a new file gets loaded into the notebook (in line 10). See the report of my Additional thesis for reflection upon the results.

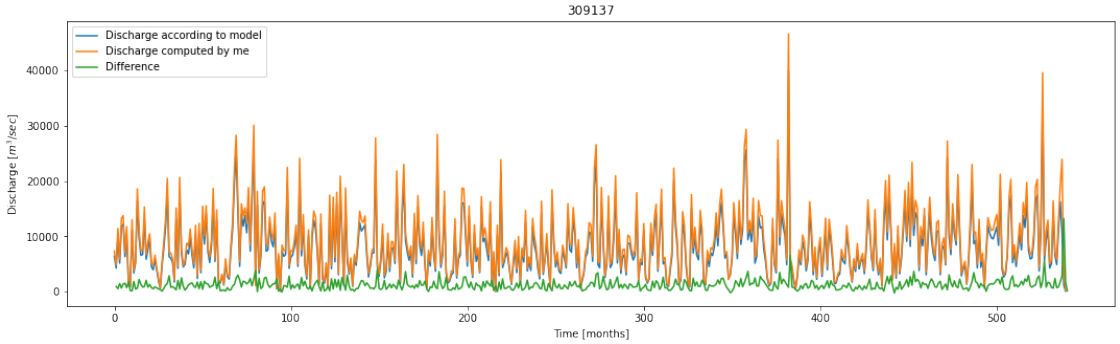
[12]:
```
"""
Area is given in m^2
Uparea is also given in m^2
prec and upcprc I assume are given in mm/day.
So need a correction of: *10^-3/(24*60*60) to transfer to m^3/sec
"""
for i in Qobs.columns[1:].astype(int):
    Qobs_intrest = i # 306719
    curr_dir = os.getcwd()
    model_output = pd.read_csv(curr_dir + '/PO_Results_dd/
↪0'+str(Qobs_intrest)+'.txt', delimiter='\t')

    # amount of precipitation in upstream area
    uparea = geodata.loc[geodata['SUBID']==Qobs_intrest]['UPAREA'].values
    up_prec = model_output['upcprc'][1:].astype(float)*uparea*10**-3/(24*60*60)

    # amount of precipitation in subbasin itself
    sub_area = geodata.loc[geodata['SUBID']==Qobs_intrest]['AREA'].values
    sub_prec = model_output['prec'][1:].astype(float)*sub_area*10**-3/(24*60*60)

    # total precipitation
    tot_prec = up_prec+sub_prec

    plt.figure(figsize=(18,5))
```

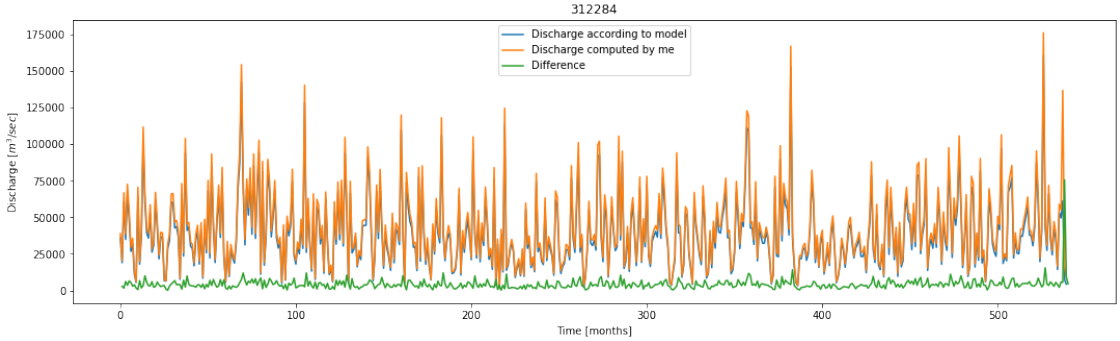
```
   plt.plot(tot_prec.values, label='Discharge according to model')
   plt.plot(VNN_month_group[str(Qobs_intrest)][120:-27].values,␣
↪label='Discharge computed by me') # the [120:] is to make it both start at␣
↪jan 1971.
   plt.title(str(Qobs_intrest))
   plt.xlabel('Time [months]')
   plt.ylabel('Discharge [$m^3/sec$]')

   plt.plot((VNN_month_group[str(Qobs_intrest)][120:-27].values - tot_prec),␣
↪label='Difference')
   plt.legend()
```

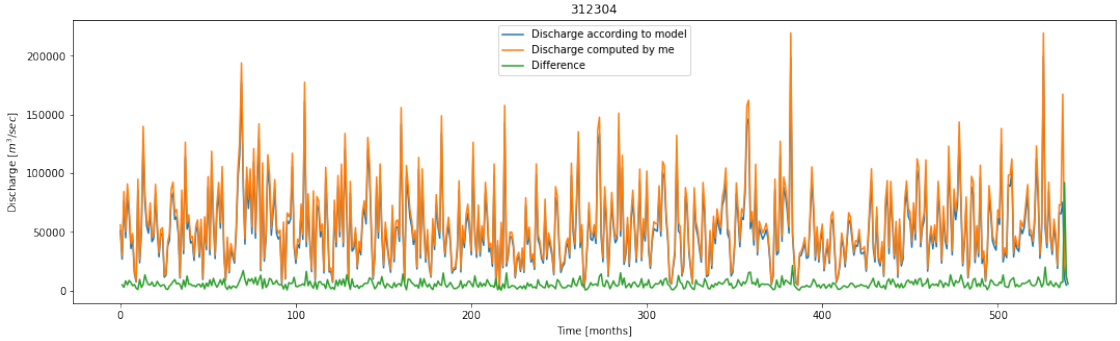


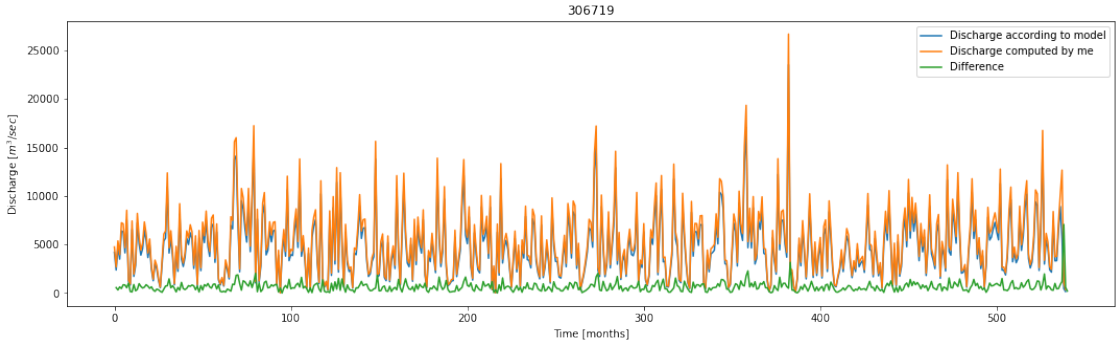


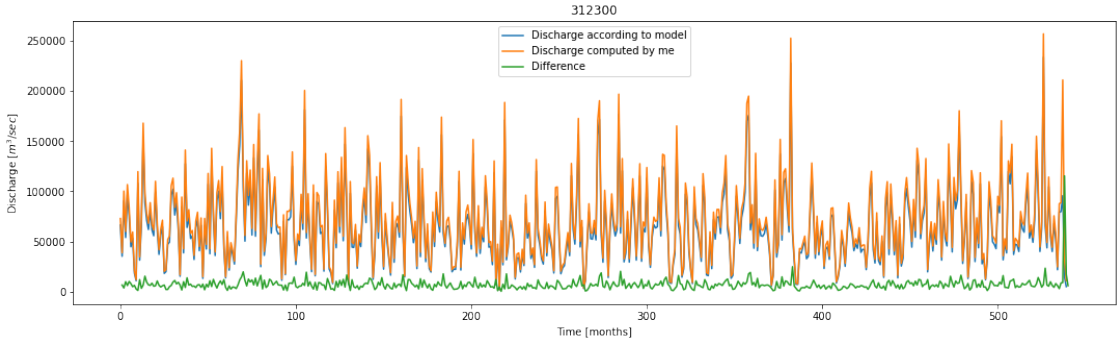


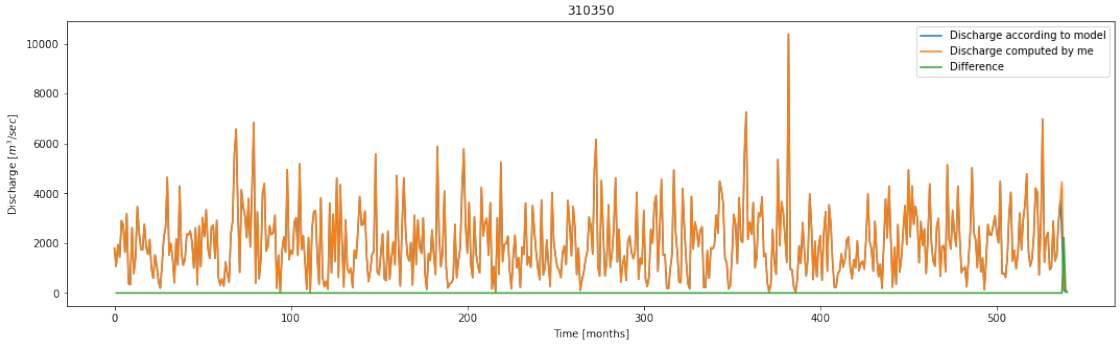

309137
Discharge according to model
Discharge computed by me
Difference
Discharge [m^3/sec]
Time [months]

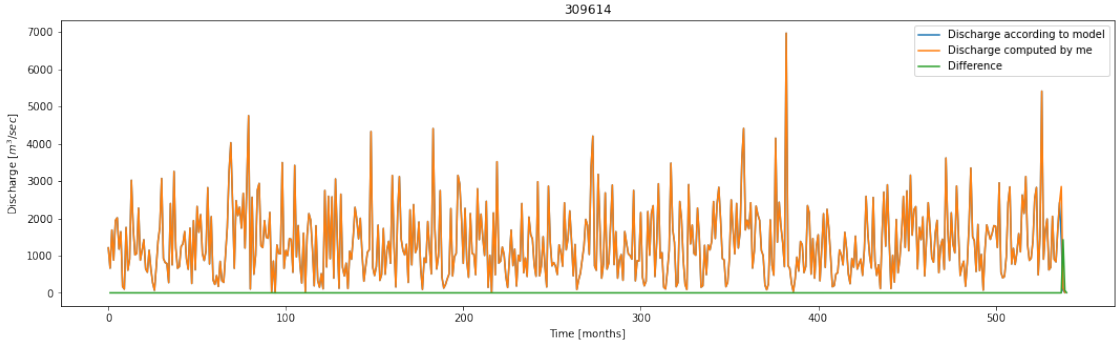

312284
Discharge according to model
Discharge computed by me
Difference
Discharge [m^3/sec]
Time [months]


312304
Discharge according to model
Discharge computed by me
Difference
Discharge [m^3/sec]
Time [months]


306719
Discharge according to model
Discharge computed by me
Difference
Discharge [m^3/sec]
Time [months]

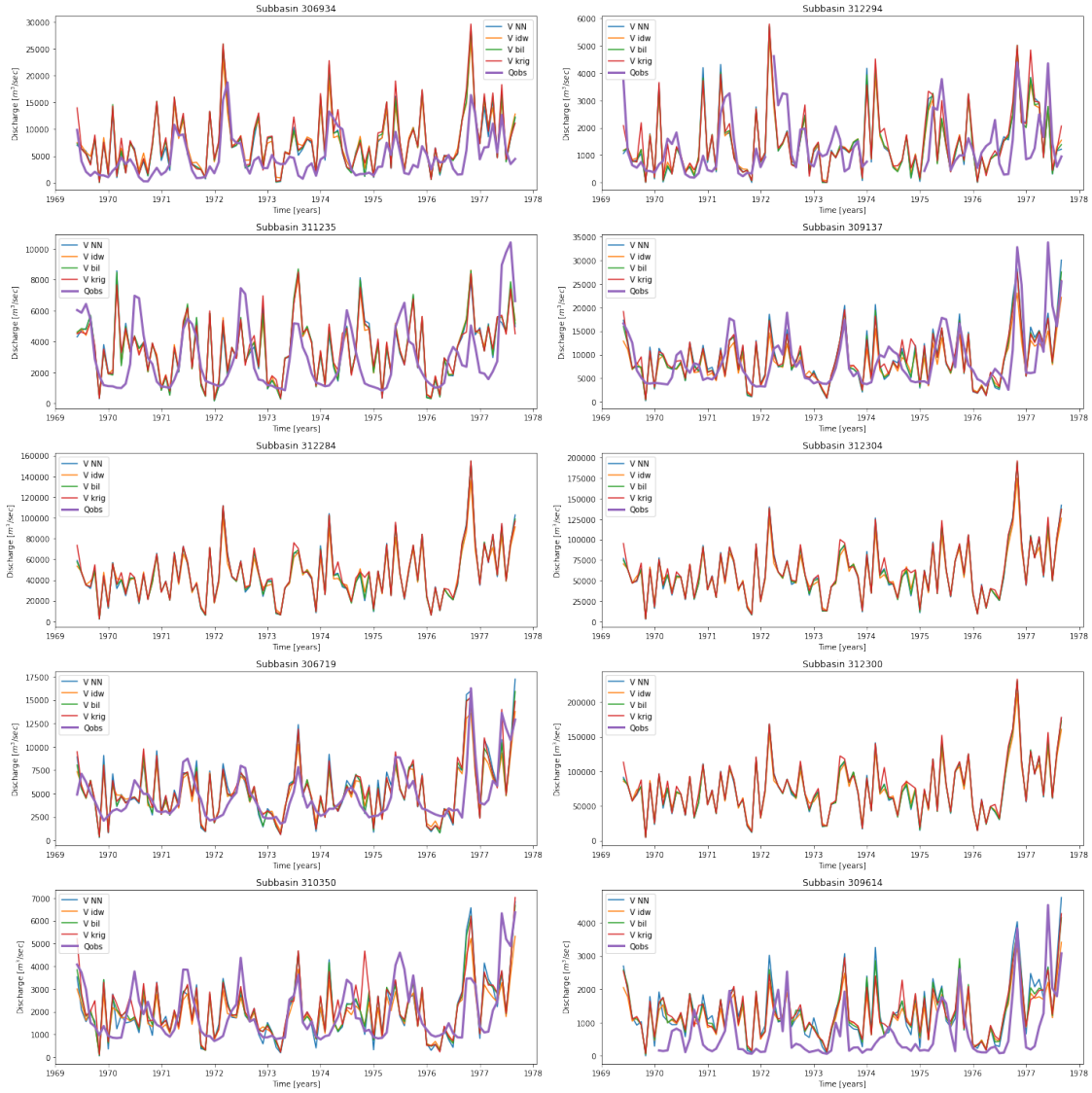

312300
Discharge according to model
Discharge computed by me
Difference
Discharge [m^3/sec]
Time [months]


310350
Discharge according to model
Discharge computed by me
Difference
Discharge [m^3/sec]
Time [months]


309614
Discharge according to model
Discharge computed by me
Difference
Discharge [m^3/sec]
Time [months]

Compare the computed discharge from the different interpolation methods with each other

[13]:
```
# To plot all of then underneath each other for longer time
# for i in Qobs_mon.columns.values.tolist():
#     plt.figure(figsize=(20,4))
#     plt.plot(VNN_month_group[i][0:650], label='V NN')
#     plt.plot(Vidw_month_group[i][0:650], label='V idw')
#     plt.plot(Vbil_month_group[i][0:650], label='V bil')
#     plt.plot(Vkrig_month_group[i][0:650], label='V krig')
#     plt.plot(Qobs_mon[i][0:650], label='Qobs', linewidth=3)

#     plt.title(f'Subbasin {i}')
#     plt.xlabel('Time [years]')
#     plt.ylabel('Discharge [$m^3/sec$]')
#     plt.legend()

# MAKE IMAGE FOR IN REPORT
plt.figure(figsize=(20,20))
ii=1
for i in Qobs_mon.columns.values.tolist():
    plt.subplot(5,2,ii)
    plt.plot(VNN_month_group[i][100:200], label='V NN')
    plt.plot(Vidw_month_group[i][100:200], label='V idw')
    plt.plot(Vbil_month_group[i][100:200], label='V bil')
    plt.plot(Vkrig_month_group[i][100:200], label='V krig')
    plt.plot(Qobs_mon[i][100:200], label='Qobs', linewidth=3)

    plt.title(f'Subbasin {i}')
    plt.xlabel('Time [years]')
    plt.ylabel('Discharge [$m^3/sec$]')
    plt.legend()
    ii=ii+1

plt.tight_layout()
```



Calculate the correlation, NSE, KGE and RE for all the sub-basins and all interpolation methods

[14]:
```
def NSE(targets, predictions):
    return (1-(np.sum((predictions-targets)**2)/np.sum((targets-np.
 ↪mean(targets))**2)))

def RE(targets, predictions):
    return (np.mean(predictions)-np.mean(targets))/np.mean(targets)*100
```

[15]:
```
# Make an array to save all the values for all sub-basins. it has 10 columns␣
 ↪for the 10 subbasins
```

```
# and 5 columns, the first 4 for the interpolation methods, then 1 for the max␣
 ↪diff for that subbasin.

# NOTE: the dimensions of these arrays need to be changed when you switch to a␣
 ↪different river basin.

correlation = np.zeros((5,10))
NSE_coef = np.zeros((5,10))
KGE_coef = np.zeros((5,10))
RE_coef = np.zeros((5,10))

i=0
for name in Qobs.columns[1:].to_list():
    # Prepare varaibles to check the coefficients
    x1NN = VNN_month_group[name][:-1].values
    x1idw = Vidw_month_group[name][:-1].values
    x1bil = Vbil_month_group[name][:-1].values
    x1krig = Vkrig_month_group[name][:-1].values
    x2 = Qobs_mon[name].values[:-3] # take out the last 3 dates, since they␣
 ↪were halved

    # To cutt off x1 since it continues for more dates than x2, and remove all␣
 ↪the NaNs.
    x1NN = x1NN[:-(len(x1NN)-len(x2))] # to cutt off the last few dates
    x1NN = x1NN[np.isnan(x2)==False] # take out all the dates with NaN values

    x1idw = x1idw[:-(len(x1idw)-len(x2))] # to cutt off the last few dates
    x1idw = x1idw[np.isnan(x2)==False] # take out all the dates with NaN values

    x1bil = x1bil[:-(len(x1bil)-len(x2))] # to cutt off the last few dates
    x1bil = x1bil[np.isnan(x2)==False] # take out all the dates with NaN values

    x1krig = x1krig[:-(len(x1krig)-len(x2))] # to cutt off the last few dates
    x1krig = x1krig[np.isnan(x2)==False] # take out all the dates with NaN val

    x2 = x2[np.isnan(x2)==False]

    #check correlation
    correlation[0,i] = scipy.stats.pearsonr(x1NN, x2)[0]
    correlation[1,i] = scipy.stats.pearsonr(x1idw, x2)[0]
    correlation[2,i] = scipy.stats.pearsonr(x1bil, x2)[0]
    correlation[3,i] = scipy.stats.pearsonr(x1krig, x2)[0]

    # KGE
    obj1 = Metrics(x2, x1NN)
    KGE_coef[0,i] = obj1.kling_gupta_efficiency(clean=True, decimal=5)
    obj1 = Metrics(x2, x1idw)
```

```
    KGE_coef[1,i] = obj1.kling_gupta_efficiency(clean=True, decimal=5)
    obj1 = Metrics(x2, x1bil)
    KGE_coef[2,i] = obj1.kling_gupta_efficiency(clean=True, decimal=5)
    obj1 = Metrics(x2, x1krig)
    KGE_coef[3,i] = obj1.kling_gupta_efficiency(clean=True, decimal=5)

    # NSE
    NSE_coef[0,i] = NSE(x2, x1NN)
    NSE_coef[1,i] = NSE(x2, x1idw)
    NSE_coef[2,i] = NSE(x2, x1bil)
    NSE_coef[3,i] = NSE(x2, x1krig)

    # RE
    RE_coef[0,i] = RE(x2, x1NN)
    RE_coef[1,i] = RE(x2, x1idw)
    RE_coef[2,i] = RE(x2, x1bil)
    RE_coef[3,i] = RE(x2, x1krig)

    i=i+1

correlation[4,:] = correlation[0:3,:].max(axis=0)-correlation[0:3,:].min(axis=0)
RE_coef[4,:] = (RE_coef[0:3,:].max(axis=0)-RE_coef[0:3,:].min(axis=0))
NSE_coef[4,:] = NSE_coef[0:3,:].max(axis=0)-NSE_coef[0:3,:].min(axis=0)
KGE_coef[4,:] = KGE_coef[0:3,:].max(axis=0)-KGE_coef[0:3,:].min(axis=0)
```

[ ]:
```
# After running this cell, you can immediately paste it in Latex.
# change the first entry in the function to the array you want to export.
a2l.to_ltx(correlation, frmt = '{:6.3f}', arraytype = 'array') # use to_clp to␣
 ↪immediately put it to clipboard
```

[16]:
```
# To check if higher correlation is achieved with the Qobs after a time shift

i = '310350' # the sub-basin id

# Check for different interpolation techniques by commenting or uncommenting␣
 ↪the lines
x1 = VNN_month_group[i][:-1].values
# x1 = Vidw_month_group[i][:-1].values
# x1 = Vbil_month_group[i][:-1].values
x2 = Qobs_mon[i].values
x1 = x1[:-(len(x1)-len(x2))] # to cutt off the last few dates
x1 = x1[np.isnan(x2)==False] # take out all the dates with NaN values
x2 = x2[np.isnan(x2)==False]

# check pearson correlation coefficient for non-shifted signal or shift till up␣
 ↪a year (14 months)
start = 0
```

```
for time_shift in np.arange(15):
    print(f'correlation coefficient for {time_shift} months shift is: {scipy.
 ↪stats.pearsonr(x1[:len(x1)-time_shift], x2[time_shift:])[0]}')
```

```
correlation coefficient for 0 months shift is: 0.5964702471101042
correlation coefficient for 1 months shift is: 0.31242484254237274
correlation coefficient for 2 months shift is: 0.12347997980609642
correlation coefficient for 3 months shift is: 0.03896269453015229
correlation coefficient for 4 months shift is: -0.02192613614148574
correlation coefficient for 5 months shift is: -0.11524027425036755
correlation coefficient for 6 months shift is: -0.05153799432144788
correlation coefficient for 7 months shift is: -0.02846403419516394
correlation coefficient for 8 months shift is: 0.0010247920837111006
correlation coefficient for 9 months shift is: 0.06667489417248754
correlation coefficient for 10 months shift is: 0.1523537905421345
correlation coefficient for 11 months shift is: 0.15534199363995427
correlation coefficient for 12 months shift is: 0.1915882770508745
correlation coefficient for 13 months shift is: 0.07914022169213047
correlation coefficient for 14 months shift is: -0.013238714696582776
```

```
[ ]:
```