NTNU
Norwegian University of
Science and Technology

Evaluation of snow simulations in SHyFT

Amir Nasser Katirachi

Hydropower Development
Submission date: September 2018
Supervisor: Knut Alfredsen, IBM
Co-supervisor: Oddbjørn Bruland, IBM

Norwegian University of Science and Technology
Department of Civil and Environmental Engineering

Agreement concerning MSc theses and supervision

This Agreement confirms that the topic for the MSc thesis approved, the supervisory issues are agreed and the parties to this Agreement (student, supervisor and department) understand and accept the guidelines for MSc theses. This Agreement is also subject to Norwegian law, the examination regulations at NTNU, the supplementary provisions and the regulations for the MSc Engineering Education program.

1. Personal information

Family name, first name: **Katirachi, Amir Nasser**	Date of birth **March 21, 1979**
Email address **amirnk@stud.ntnu.no**	Phone

2. Department and program of study

Faculty **Faculty of Engineering**
Department **Department of Civil and Environmental Engineering**
Program of study **Hydropower Development**

3. Duration of agreement

Starting date **January 16, 2018**	Submission deadline* **June 12, 2018**
If part-time study is approved, state percentage:	

** Including 1 week extra for Easter*

All supervision must be completed within the duration of the agreement.

4. Thesis working title

Evaluation of snow simulations in SHyFT

5. Supervision

Supervisor **Knut Alfredsen**
Co-supervisor: **Oddbjørn Bruland** (IBM)

Standardized supervision time is **25 hours** for 30 credits (siv.ing) and **50 hours** for 60 credits (MST) theses.

6. Thematic description

1 BACKGROUND

Snow is a very important component in the hydrological cycle in Norway and crucial for determining reservoir operation during the spring flood to ensure full reservoir and as little flood spill as possible. The Statkraft Hydrological Forecasting Toolbox (SHyFT) is a newly developed hydrological toolbox that is used for forecasting inflow in the Statkraft system. This is a flexible system in which model can be custom designed for various purposes. The SHyFT toolbox currently have three different methods for simulating snow accumulation and storage, and these are not yet evaluated with snow data. The purpose of this master thesis is to evaluate the SHyFT snow routines against observed snow data from satellite images and snow measurements in the field.

2 MAIN QUESTIONS FOR THE THESIS

The main questions for the thesis can be stated as follows:

1. Prepare the data needed to calibrate the SHyFT model for the Nea-Nidelva catchment. This includes climatic data from observation sites in the catchment and other climatic data derived from other stations. Collect the data needed for evaluating the snow simulations, including both satellite imagery and from measurement campaigns in the field. Decide on the periods that should be used for calibration and evaluation based on the available data.

2. Calibrate SHyFT for Nidelva for all three snow routines. Compare the calibrations and evaluate their goodness based on standard parameters measuring runoff distribution and runoff volume.

3. Compare simulated snow from the three setups from 2) against each other and against observed snow data. Perform a statistical analysis to evaluate both the temporal and spatial accuracy of the simulated snow. Measures of goodness of fit both for temporal and spatial variation should be decided and used in this task. Discrepancies should be quantified and evaluations should be done to try to identify reasons for any differences between observed and simulated snow cover and water equivalent such as autumn snow start errors or errors in snow volume over the winter.

4. Based on the findings in 3), try to improve the snow simulations in the model. In addition to parameters in the snow routines, the temperature and precipitation distribution and gradients should be evaluated. Any proposed changes should be evaluated using the calibrated model and the indices of goodness of fit from 3).

5. Evaluate the possibility of including snow data in the calibration of the model, and recalibrate SHyFT using available snow data.

3 SUPERVISION, DATA AND INFORMATION INPUT

Professor Knut Alfredsen and Professor Oddbjørn Bruland, NTNU will be advisors on the project. Dr. Yisak Abdella and Dr. Knut Sand at Statkraft will contribute to the project based on their experience with SHyFT, snow measurements and snow simulations. Professor Knut Alfredsen will handle the formalities related to the supervision.

Discussion with and input from colleagues and other research or engineering staff at NTNU, SINTEF, power companies or consultants are recommended. Significant inputs from others shall, however, be referenced in a convenient manner.

The research and engineering work carried out by the candidate in connection with this thesis shall remain within an educational context. The candidate and the supervisors are therefore free to introduce assumptions and limitations, which may be considered unrealistic or inappropriate in a contract research or a professional engineering context.

4 REPORT FORMAT AND REFERENCE STATEMENT

The thesis report shall be in the format A4. It shall be written as a manuscript ready for submission to the journal Hydrology Research. The manuscript must therefore adhere to the length requirements, formatting rules and standards for figures and tables given by the journal. Extra material can be submitted in the thesis as appendixes that each should include a short introduction to the material.

The report shall have a professional structure, assuming professional senior engineers (not in teaching or research) and decision makers as the main target group.

7. Other Agreements

Supplementary agreement	**Not applicable**
Approval required (REK, NSD)	**Not applicable**
Risk assessment (HES) done	**Not applicable**

Appendix (list)

8. Signatures

Conditions	Date	Signatures
I have read and accept the guidelines for MSc theses		Student
I take the responsibility for the supervision of the student in accordance with the guidelines or MSc theses		Supervisor
I take the responsibility for the co-supervision of the student in accordance with the guidelines for MSc theses		Co-supervisor
Department/Faculty approves the plan for the MSc thesis		Department/Faculty

Acknowledgment

I would first like to thank my thesis advisor professor Knut Alfredsen. I always had his prompt email responses whenever I ran into trouble spot or had a question about my research or writing. He supervised this project during last summer and dedicated his time to this project even while he was on vacation.

I would like to acknowledge professor Oddbjørn Bruland as the second advisor of this thesis, and I am grateful for his valuable comments on this thesis.

I would also like to thank Dr. Yisak Abdella of Statkraft for providing this project with requisite input data.

Also, my sincere gratitude goes to Andrew Mabula, MSc. Hydro power development student at NTNU for his great support and help and Felix Nikolaus Matt, department of geoscience university Oslo and Sigbjørn Helset from Statkraft for their vital assistance in SHyFT.

Finally, I express my very profound gratitude to my wife and my parents who have supported me throughout entire process. This accomplishment might not have been possible without them.

Amir Nasser Katirachi
September 2018

Table of content

List of Appendices

List of figures

List of tables

List of equations

Abbreviations

API	Application Programming Interface
AROME	Application of Research to Operations Mesoscale
BOBYQA	Bound Optimization By Quadratic Approximation
BTK	Bayesian Temperature Kriging
CDF	Cumulative probability Distribution Function
CV	Coefficient of Variation
DDD	Distance Dynamics Model
ECMWF	European Center for Medium Range Weather Forecast
ENKI	Dynamic Environmental Model Framework
GPR	Ground Probing Radar
GPS	Global Positioning System
HBV	Hydrologiska Byrans Vattenbalansavdelning
IDW	Inverse Distance Weighting
KGE	Kling Gupta Efficeincy
m.a.s.l	meters above sea level
MAE	Mean Absolute Error
MAESWE	Mean Absolute Error of Snow Water Equivalent
MODIS	Moderate Resolution Imaging Spectroradiometer
NetCDF	Network Common Data Frame
NSE	Nash Sutcliffe Efficiency
NWP	Numerical Weather Prediction
PCC	Pearson Correlation Coefficient
PDF	Probability Density Function
PTGSK	Priestley Taylor Gamma Snow Kirchner
PTHSK	Priestley Taylor HBV Snow Kirchner
PTSSK	Priestley Taylor Skaugen Snow Kirchner
SCA	Snow Cover Area
SCE-UA	Shuffled Complex Evolution method developed at the University of Arizona
SD_LN	Snow Distribution Log-normal
SDC	Snow Depletion Curve
SHyFT	Statkrafts Hydrological Forcasting Tool Box
SWE	Snow Water Equivalent
SWEAE	Absolute Error of Snow Water Equivalent
UTM	Universal Transverse Mercator
YAML	Yet Another Markup Language

1. Introduction

Water is considered as a vital source in all countries as it has significant economic, environmental and social values which are growing rapidly. Seasonal snow covers mountains in high-altitude regions of the Earth and supply valuable water resources for various activities such as; winter entertainment, irrigation, drinking water and hydropower production.

This study was done on Nea-Nidelva catchment in the center of Norway. The catchment is mainly forested and covered with snow for more than five months of the year. The accumulation season usually starts in November and snowmelt starts in April or May. The snow melt season is rather long and less intense. Grid lines of 1 km interval where used with the catchment being gridded into 3606 cells. The area of cells near the edge where less than 1 km² while the inner ones gave 1 km². The total area of the catchment is 2876 km².

There are six snow courses in the high elevation of the catchment. The snowpack data is collected by Statkraft each year. SHyFT (Statkraft Hydrological Forecasting Toolbox) is the main program in this study. There are three methods for snow simulation in this program. Each routine has a different number of parameters that share specific parameters. The model is a conceptual model which is able to use free variables for modeling. Evapotranspiration and soil response routines are the same in three methods but different snow routines. The Priestley Taylor equation is used for evapotranspiration which is more simple and straight forward compare with Penman equation and Kirchner formulas which are used for soil response.

SHyFT utilizes a C++ core which was designed for ENKI program previously. ENKI stands for Dynamic Environmental Model Framework. The model was developed by Statkraft and later developed and enhanced by SINTEF center. Although ENKI is a powerful hydrological tool, it is not fast enough in operation. Due to the need for a faster application for calibration in practical activities, SHyFT model was introduced. The SHyFT core is written in C++ but the API (Application Program Interface) is created in Python language and many useful Python classes are accessible in SHyFT. Due to heavy computations in SHyFT, the C++ language is used to have more control on memory leakage since C++ is more of a low-level language compared to Python and provides more flexible for memory management. The program can be used to simulate distributed model as well as lumped ones. In distributed models, the data is assigned to every single cell but in SHyFT there is no connection between cells instead there is connection between cells and their associated outlet. Every hydrological simulation model includes; evaporation routine, snow routine and soil response routine. SHyFT use Priestley Taylor equations for Actual Evaporation ('ae' is used in SHyFT yaml file). There are three snow routines in SHyFT; Gamma distribution Snow, HBV Snow and Skaugen Snow along with another routine-HBV Stack which is similar to HBV Snow. The Kirchner routine is the response routine used in SHyFT while either the Nash–Sutcliffe efficiency (NSE) or Kling–Gupta efficiency (KGE) can be used for calibration judgment. A correction value to modify precipitation biasedly is also provided in SHyFT.

Comparing Observed and simulated Hydrographs is the main key point to validate a simulation. Calibration means adjusting the free parameters of a catchment to simulate a synthetic hydrograph not exactly resembling but having a good fit within acceptable margins against the observed values. Three methods PTGSK (Priestley Taylor Gamma Snow Kirchner), PTHSK (Priestley Taylor HBV Snow Kirchner) and PTSSK (Priestley Taylor Skaugen Snow Kirchner) were studied in this project.

AROME meteorological data was used for this study. Instead of using the observed stations data and then distributing it by a given equation, the distribution AROME data was used. AROME is a numerical convective-scale forecast operational model. This numerical weather prediction took six years to be developed and validated before it became operational in 2008. High grid resolution enhances regional prediction of mesoscale phenomena to 2.5 KM resolution grid was selected for AROME. Arome data are forecast distributed data and has good spatial coverage. Other projects with the same AROME concept and various grid resolution are used in some European countries. AROME produces prognostic variables such as; Temperature, moist content, wind speed and etc. French radar network systems are used to improve the spatial and quantitative values of precipitation forecast. The model also uses ECMWF (European Center for Medium Range Weather Forecast) for radiation data. Arome was evaluated with ALADIN-France forecast in 2008. (Seity, Brousseau et al. 2011). Although these are not observed but forecasted data which have some tolerances with real values, they are not distorted by distribution equations as they are high grid resolution distributed AROME data. One of the main triggers in the use of AROME was the devastating flash flood in the south of France and needs for a high grid resolution NWP (Numerical Weather Prediction). Another benefit of Arome is that it doesn`t need to distribute data based on data point stations as it is already distributed.

Priestley Taylor was used for Evaporation routine. Due to the need for a number of input data for the Penman equation, Priestley Taylor was introduced in Australia in 1972 which is simple and uses the dimensionless empirical approximation value, Priestley-Taylor coefficient, for input data other than radiation data. This makes it suitable for places that do not have either one or both of relative humidity and wind speed data.

There are three snow routines. First HBV snow routine, HBV is hydrological model which was developed in Sweden. Snowmelt is computed by a degree-day method in HBV snow routine. Figure (1) shows the HBV snow routines formulas.

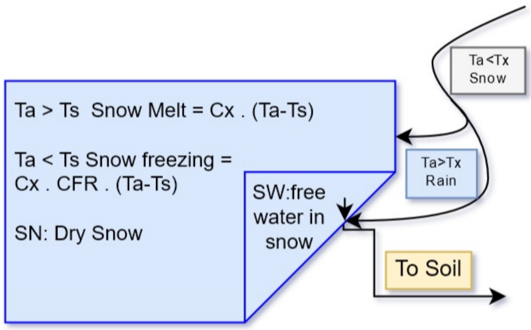


Figure intro.1 HBV Snow routine model

The second snow routine is Gamma snow distribution. (Kolberg and Gottschalk 2010) assumes that Snow Cover Area (SCA) is homogenous in all sub grid cells in Gamma snow routine. While the boundaries of SCA is from 1 to 0, the (1-SCA) is a function of accumulated snow melt depth. Gamma Snow routine illustrates with a Snow Depletion Curve (SDC). The relationship between mass balance of a heterogeneous snow cover and the fractional snow cover area is represented by SDC. The equation (1) shows SDC model for a single cell, (A) for a short term of SCA, P () the probability density function (PDF) and F () for the Cumulative probability Distribution Function (CDF)

Equation intro.1 SDC model for a single cell

$$A(t) = A_0 \cdot \{1 - F[\lambda(t)]\} \qquad F[\lambda(t)] = \int_0^{\lambda(t)} p(x; m, cv)\mathrm{d}x = \gamma\left(\frac{1}{cv^2}, \frac{1}{cv^2} \cdot \frac{\lambda}{m}\right)$$

A(t) is snow cover area of a cell in time t. Four variable define the snow pack state in every cells

1. m the average Snow Water Equivalent of the cell at the melt season start (mm)
2. Coefficient of variation (CV) of SWE which shows the heterogeneity of that cell
3. A(0) The Snow Cover Area of the cell at the start of season melt. (0 to 1)
4. λ(t) the accumulated snow melt since snow melt start (mm)

The first three variables are SDC parameters. It assumes that the SDC doesn't change during snow melt season so that these three variables remain static during the period but only λ changes during snow depletion. The most important variable of a SDC is the CV as it governs the gradual reduction of SCA. Previous researches show that a CV value between 0.4 to 0.9 for based ground terrain and 0.2 and smaller for forest and low wind speed lands are a good estimate. The value of A equivalent to 0 means the snow-covered area fraction was not covered with snow even in the mid-winter season as a result of either wind erosion or avalanche activities. (Kolberg and Gottschalk 2010).

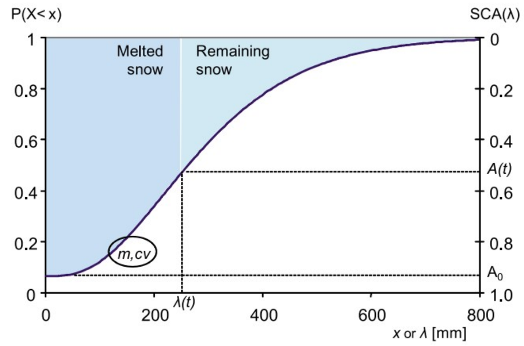


Figure intro.2 SDC in Gamma snow routine

The snow depletion curve (SDC) based on the Gamma distribution snow routine is illustrated in figure (2). Snow melt sensitivity to wind and snow-rain threshold temperature are free parameters in Gamma distribution snow routine.

The third method is Skaugen method. The main objective of developing Skaugen snow model, also known as the Distance Dynamics Model (DDD), is to reduce the number of calibration against observed runoff as much as possible. Reducing the number of free variables can decrease uncertainty of the model and make it easier to diagnose it while maintaining its accuracy as a modern hydrology model. The main parameters in this model are precipitation and temperature. The spatial Probability Density Function (PDF) of Snow Water Equivalent (SWE) is Snow Distribution Log-normal (SD_LN) for this model. The sum of uniform and log-normal distribution snowfall events make the PDF and the coefficient of variation (CV) is constant. The spatial distribution of every snowfall has a fixed shape with calibrated CV regardless of its intensity.

In this approach nine quantiles are estimated for every snowfall and in all snowfall events the highest SWE quantile always gets the most SWE with the second highest SWE quantile getting the second highest and the sequence continues accordingly while the coefficient of correlation of the sum of events remains constant. The spatial distribution of snow melting is constant and the value of Snow Cover Area (SCA) decreases when the SWE of a quantile drops to zero. The sum of zero quantile makes the free snow area fraction. (Skaugen and Weltzien 2016)

Concerning the response routine, the spatial heterogeneity of material properties and complicated physical processes control the subsurface flow. The hydrology of catchments is complicated and requires simplification to make an ideal model. In some hydrological models, microscale physics properties are generalized to an entire catchment whilst lacking clarity on correctness. Kirchner is a simple response routine formula which SHyFT makes use of.

The SCE-UA (Shuffled Complex Evolution method developed at the University of Arizona) is an efficient and effective global optimization which has been used in many watershed model calibrations. This optimization method has some unique specifications which converge globally even in the presence of multiple regions of attraction, and is not trapped by small bumps and pits on the objective surface. (Duan, Sorooshian et al. 1994). MIN-BOBYQA (Bound Optimization By Quadratic Approximation) is a local deterministic algorithm. Although it is possible to calibrate many parameters, it is not clear if it is a global response or not. In this study MIN-BOBYQA was used.

After Completion of the calibrations based on the three methods, the results obtained did not correlate to the observed. The PTHSK and PTSSK simulated just around 1% of the observed snow pack necessitating check on the scripts which was conducted thoroughly. However, the check did not identify any errors but prompted a discussion with the supervisor, Professor Knut, and another expert researcher in Oslo, Felix Nikolaus Matt. This resulted in the identification of a significant bug in the SHyFT program. The bug was reported to the developer of SHyFT and it was fixed. Although this event held the project for a few weeks, it was worth it as the program received an update.

2. Paper

Evaluation of snow simulations in SHyFT

Evaluation of snow simulations in SHyFT

Abstract:

The main objective of this paper is to evaluate the SHyFT snow routine performance against observed snow data obtained from snow measurements in the field. SHyFT (Statkraft Hydrological Forecasting Toolbox) is an open source hydrological model recently developed by Statkraft for forecasting inflow. The system of the model has been designed in a flexible manner to allow ease of customization for various purposes. The SHyFT toolbox currently has three different methods for simulating snow accumulation and snow depletion, which have not previously been evaluated with snow data. These methods are Gamma snow distribution routine, HBV snow routine and Skaugen snow routines. Each method has its own parameters and unique approach to model the snow accumulation and snow melt.

The case study is based on 2876 km² area of the Nea-Nidelva catchment which is located in the center of Norway (63° N, 10° E). A grid scale of 1 km is used, dividing the catchment into 3606 cells with some of the cells, located around the edges, being less than 1km².

Simulated results of SWE (Snow Water Equivalent) based on the three previously mentioned methods using data from 2012 to 2017 were compared against observed SWE from snow course data. The snow course data, provided by Statkraft, has 6 snow courses in a period of 5 years. The SWE data were recorded using a snow radar which collected it per meter, on average. Each snow course transect passes some cells so that these cells contain observed data points.

200 calibrations were conducted for each method, out of which the best 36 results were selected. An average of these was computed and used as a representative of the associated method. Simulations with the calibrated data were conducted and the results analyzed and discussed. The analysis was based on correlation of specific parameters; slope, elevation, terrain roughness and number of observed points in a cell against the absolute error of SWE, of which a discussion is provided. Further, in addition the Pearson correlation coefficient were presented for each scatter plot and the three methods. The hydrographs, SCA (Snow Cover Area) and SWE graphs were examined to observe similarities and dissimilarities of the different methods. The results of PTGSK (Priestley Taylor Gamma Snow Kirchner) show a variation in SCA

as well as in SWE when compared with PTSSK (Priestley Taylor Skaugen Snow Kirchner) and PTHSK (Priestley Taylor HBV Snow Kirchner). Further, the comparison of satellite images with SCA images provides a good basis for identifying the method that generates a better simulation.

KEYWORDS: SHyFT, Gamma snow distribution routine, HBV Snow routine, Skaugen snow routine, SWE, SCA, snow pack, snow course, spatial resolution satellite images, MODIS

1. INTRODUCTION

Norway has significant snowfall each year accounting for 30% of annual precipitation. This qualifies snow to be one of the main components of the hydrological cycle which plays a vital role in hydropower production. It is possible to operate hydropower reservoirs with minimal flood spill during early spring provided sufficient knowledge of spatial and temporal distribution of snow storages is available. (Skaugen and Weltzien 2016) (Kolberg and Gottschalk 2010). In this study, the Nea-Nidelva catchment in the center of Norway was modeled with SHyFT (Statkraft Hydrological Forecasting Toolbox) based on three snow routines; Gamma snow distribution, HBV snow and Skaugen snow.

SHyFT is a conceptual model. There are large number of variables in conceptual models which are not possible to measure directly but must instead be estimated using a calibration process of fitting the simulated outputs of the model to the observed outputs.(Duan, Sorooshian et al. 1994). Two important parameters in snow routines are SWE (Snow Water Equivalent) and SCA (Snow Cover Area).

The flexibility and robustness of the HBV model structure as well as the simplicity of input data make it reliable for hydrological modelling. (Al-Safi and Sarukkalige 2017). Gamma snow distribution use SCA to model the SWE with more variables than HBV. The Skaugen snow is the latest routine and uses log normal distribution for Probability Density Function (PDF) In SWE and the SCA for the entire precipitation area reaches to a value of one after every snowfall event, which means it assumes that snow covers all the ground. (Skaugen and Weltzien 2016).

It is not easy to make judge regarding their parameters as well as to find which variable(s) are more significant than the other. Many parameters and variables control snow melting and refreezing., It is reasonable to consider the temperature only but a study shows that albedo is more important than air temperature in mountainous areas. (Kolberg and Gottschalk 2006).

Good estimate of snow reservoir and reasonable forecast of precipitation and temperature guarantee a reliable knowledge on the discharge. Some drivers change the spatial distribution of snow after falling. These include wind and topography which intensify on high elevations and steep terrains. (Kolberg and Gottschalk 2010). Spatial distribution of SWE has a great effect on snow melting pattern. A large spatial heterogeneity in snow storage accompanied with large snowfall measurement errors make operational management difficult. (Kolberg and Gottschalk 2010). In this study, the AROME input data is used as meteorological data. In most of the conceptual models, including this study, calibration and validation are narrowed to comparing simulated results with observed runoffs. (Seibert 2000).

2. MATERIALS AND METHODS

2.1 Software application used:

The model is coded in SHyFT which is an integrated, sophisticated open source hydrological model and being developed and supported by Statkraft. The core of the model is written in C++ language which was previously used for another simulation model-ENKI. When modeling multiple catchments for runoff calibration with NSE method, ENKI is the better option in comparison to SHyFT (Shrestha and Aryal 2011). Distributed models such as SHyFT compensates the heavy computational and preparatory tasks by increasing the simulation precision using more explicit areal distributed input and processes (Rinde 1996).

2.2 Meteorological data (AROME):

AROME is a numerical convective-scale forecast operational model which stands for Application of Research to Operations Mesoscale, developed in France. Numerical weather prediction (NWP) were introduced more than a half century ago and have progressed due to high processing capacity of super computers. (Seity, Brousseau et al. 2011). Hourly forecast of precipitation, Temperature, wind-speed, relative-humidity and radiation in AROME, were available for the period (2012-2017) by Statkraft.

2.3 Precipitation correction scale factor:

This is used to correct the accumulation precipitation for a specific region during the calibration. The initial value is one but can be set between upper and lower boundaries for calibration.

2.4 Evaporation routine:

The Priestley Taylor is used as the evaporation routine in SHyFT.

$$PET = \alpha \frac{s(T_a)}{s(T_a)+\gamma}(K_n + L_n)\frac{1}{\rho_w \lambda v}$$ *Equation (1) Priestley Taylor formula*

Equation (1) shows the Priestley Taylor formula, where, PET is the potential evaporation, K_n the short wave radiation, L_n the long wave radiation, $s(T_a)$ the slope of satuaration vapor pressure vesus the temperature curve, γ is the Psychometric constant, ρ_w is the mass density of water and λv is the latent heat of vaporization and the value used is 2.45 MJ kg-1. α is the Priestley-Taylor`s constant. However, it fluctuates throughout seasons and days, and average value of 1.26 is used. (Priestley and Taylor 1972)

2.5 Snow routine

2.5.1 HBV Snow routine:

The HBV (Hydrologiska Byrans Vattenbalansavdelning) is a hydrological model that was originally developed at the Swedish Meteorological and Hydrological Institute (SMHI) by Dr. Sten Bergström for runoff simulation and hydrological forecasting. The model is a semi-distributed conceptual model which has been subjected to a number of modifications over time although the philosophy of the model has not changed. (Bergstr6m 1997).

2.5.2 Gamma Snow routine:

(Kolberg and Gottschalk 2010) assumes that Snow Cover Area (SCA) is homogenous in all sub grid cells. While the boundaries of SCA is from 1 to 0, the (1-SCA) is a function of accumulated snow melt depth and gamma Snow routine illustrates with a Snow Depletion Curve (SDC). The relationship between mass balance of a heterogeneous snow cover and the fractional snow cover area is represented by SDC. The equation (2) shows SDC model for a single cell and (A) is a short term of SCA and P () is probability density function while F () is cumulative probability distribution function (CDF)

Equation (2) SDC model for a single cell

$$A(t) = A_0 \cdot \{1 - F[\lambda(t)]\} \qquad F[\lambda(t)] = \int_0^{\lambda(t)} p(x; m, cv)\mathrm{d}x = \gamma\left(\frac{1}{cv^2}, \frac{1}{cv^2} \cdot \frac{\lambda}{m}\right)$$

A(t) is snow cover area of a cell in time t. Four variables define the snow pack state in every cells

1. m the average Snow Water Equivalent of the cell at the melt season start (mm)
2. Coefficient of variation (CV) of SWE which shows the heterogeneity of that cell
3. A (0) The Snow Cover Area of the cell at the start of season melt. (0 to 1)
4. λ(t) the accumulated snow melt since snow melt start (mm)

2.5.3 Skaugen Snow routine:

The input data are temperature and precipitation while the major parameters are estimated from observed data directly in Skaugen model which is also called Distance Dynamics Model (DDD) (Skaugen and Onof 2014, Skaugen and Weltzien 2016). The model uses unique distribution method for each catchment respect to the Geographical Information System(GIS) and drives the dynamics of runoff from distribution of distance from points in the water basin to the closest stream. This model is a semi-distributed model and distributes the precipitation in to ten equal areas with different elevation. Degree-day model is used for snow melting in this model. (Skaugen and Onof 2014)

2.6 Response routine:

SHyFT uses the Kirchner formulas to model soil response. The model assumes that Q is a function of catchment storage (S).

Q = f (S) *Equation (3) Q is a function of catchment storage*

In some catchments large fractions of precipitation flow directly to the stream so the premise is not correct in such a case. The model also assumes that the net ground water flow across the catchment boundary is zero and saturated and unsaturated body masses are hydraulically connected.

$$\frac{ds}{dt} = P - E - Q$$ *Equation (4) conservation mass*

P, E and Q are the rates of precipitation, evaporation and discharge respectively.
The sensitivity function shows the sensitivity of the discharge to storage changes: g(Q),

$$g(Q) = \frac{dQ}{dS} = \frac{dQ/dt}{dS/dt} = \frac{dQ/dt}{P-E-Q}$$ *Equation (5) sensitivity function*

If E & P << Q, then

$$g(Q) = \frac{dQ}{dS} = -\frac{dQ/dt}{Q} \Rightarrow \ln(g(Q)) = \ln\left(\frac{-\frac{dQ}{dt}}{Q}\middle| P \ll Q \,\&\, E \ll Q\right) \# \, c1 + c2.\ln(Q) + c3.(\ln(Q))^2$$

0.0001 mm of water level is set as the minimum Q value for each step and if the supplied Q smaller than this value, the new Q is set as the new water level to keep the stability of algorithm. C1, C2 and C3 the Kirchner parameters model. (Kirchner 2009)

2.7 Agreement between Observed and simulated values:

Minimizing the differences between observed and simulated values is the purpose of calibration (Iskra and Droste 2008). There are a number of methods used to compare simulated and observed values. Nash-Shutcliffe efficiency (NSE) is highly popular in the hydrological modeling. (Al-Safi and Sarukkalige 2017). In this study, NSE method is used. Eq. (6) shows the Nash-Shutcliffe efficiency formula

$$R2 = 1 - \frac{\sum(Q_{obs}-Q_{sim})^2}{\sum(Q_{obs}-\bar{Q}_{obs})^2}$$ *Equation (6) Nash-Shutcliffe efficiency*

Where Q_{obs} and Q_{sim} are observed and simulated runoff respectively and $\bar{Q}_{obs}$ is the mean value of observed runoff data. The result of NSE varies from minus infinity and 1. The value of

1 indicates a perfect fit. The further the value is lower than one, the lower the accuracy of the results.(Nash and Sutcliffe 1970).

2.8 Optimization methods:

There are some optimization methods that can be set in SHyFT such as; MIN-BOBYQA, dream or SCE-UA. MIN-BOBYQA was employed as the optimization method in this study as it is faster than other in SHyFT. The Hydrological calibration consists of adjusting daily flows and water balance for the whole period (2012-2017).

2.9 Study catchment:

The catchment is Nea-Nidelva located latitude 63° and 64° N, longitude 10° and 12° with an area of 2876 Km^2. The land elevation is between sea level and 1750 m.a.s.l which is situated in the center of Norway. The elevation of hilly terrain increases toward the east and just 10% of region is higher than 1000 meters. The main river, Nea-Nidelva, drains to the fjord of Trondheim.

2.10 Snow data collection

Data of snow depth and snow density for this study were obtained from 6 permanent snow courses in the Nea-Nidelva catchment set by Statkraft. The snow courses are located at around 1000 m.a.s.l. on the east part of the catchment. All of them are located in zones with similar amount of low vegetation or bare ground. The layout of snow courses is distributed on high mountain which is completely covered by snow for more than six months. The data collection was conducted in the early weeks of April except in 2017 when it was collected in the first week of March. It is not so easy to collect correct data during the melting period due to the need of manual calibration to determine the snow depth and snow density. (Marchand, Bruland et al. 2001). The snow course transects are fixed by GPS (Global Positioning System) between 2.8 km and 5 km along snow courses. The snow depths were measured at one-meter interval on average by Ground Probing Radar (GPR) which is adopted for the analysis of thickness and density of the snow cover. (Sand and Bruland 1998)

3. RESULTS AND DISSCUSION

3.1. Calibration and validation:

The three methods used are PTGSK (Priestley Taylor Gamma Snow Kirchner), PTHSK (Priestley Taylor HBV Snow Kirchner) and PTSSK (Priestley Taylor Skaugen Snow Kirchner). 200 calibrations with different random initial values were conducted for each method in a three-year period (2012-2015). There are many combination sets of free variables that would make good fit with observed data (Duan, Sorooshian et al. 1992). The calibrated parameters were examined after each calibration to ensure that none of them had gone beyond their limits. If one parameter exceeded its limit all previous calibrations were discarded and the limits of that parameter were changed and the calibration process redone. The water balance was adjusted after each calibration by modifying the precipitation factor and rerunning the simulation with a new parameter set. The validation process was done for a two-year period (2015-2017) and the average of the 36 highest NSEs for each method selected as the representative of the respective method. The NSEs of calibrations for all the methods is 77% while the validation NSEs are 67%, 81% and 79% for PTGSK, PTHSK and PTSSK respectively. NSE is normally between 0.8 and 0.95 for high-quality input data. (Al-Safi and Sarukkalige 2017). Even though all three methods failed to simulate some peaks, the hydrograph results showed that the calibrated discharges are well in agreement with observed data series inclusive of validation hydrographs.

3.2. Snow courses SWE calculation:

A Python script codes was written to read the GPR logged snow data (X and Y coordinates and SWE of all points in snow course transect) and the center points of all cells (1 km X 1km). Each snow course line point was allocated to the cell in which it was nearest to. The snow course line points of a cell were categorized into an individual group in which the SWE averages were calculated and designated as observed SWE. If the number of observed points in one cell were less than 200 points, the data was discarded. The length of 200 points in a typical snow course transect is 200 meters in average which translates to one-fifth of a side of a cell. This procedure was repeated for six snow courses from 2012 to 2017 for three methods.

3.3. Cells terrain characteristics:

Cell No.	Orientation	Elevation	Slope	Cell No.	Orientation	Elevation	Slope
210	North-West	1175	29%	765	North-East	970	5%
211	North-West	1055	23%	778	East	1218	21%
212	North-West	853	25%	779	North-East	1118	10%
213	North-West	820	5%	786	North-East	931	3%
239	North-West	1393	25%	800	North	1068	10%
242	North-East	841	6%	800	North	1068	10%
243	North	786	5%	833	West	895	11%
279	West	816	11%	855	South-West	965	9%
308	West	928	11%	856	South-West	1059	11%
336	West	956	6%	857	West	1051	11%
337	South-West	1007	7%	858	North-West	1158	12%
359	West	1007	12%	864	East	1148	15%
378	North-West	1064	21%	879	East	1005	13%
744	North-East	1016	4%	893	North	952	5%
745	East	1044	9%	907	North-West	962	3%
756	South-East	1494	31%	2627	North-West	1393	29%

Table 1 Terrain characteristics of all measured cells

Table (1) tabulated the terrain characteristics of all measured cells and these data were used to find the relationship between various characteristics and observed data. Table (1) is sorted based on the cell No. and the orientation and slope were calculated based on one Km grid net therefore the slopes show the average elevation-gradient of each cell and the same for orientation.

3.4. Observed and simulated SWE of interested cells:

Figure (1) illustrates observed and modeled SWE values by the three methods in all the measured cells. The black dots show the average observed SWE in a cell and its heads and tails are the average observed SWE plus and minus SWE standard deviation. There are great differences between SWE observed in different points in a cell. The snow depth profile shows high variability in the snow depth (Marchand, Bruland et al. 2001). The colorful lines depict the SWE value for the three methods. In 2013, all the methods tend to show less SWE than what was measured in most of the cells whilst others show more. In all the cells and the whole period, PTGSK shows less SWE than the two other methods by 60 [mm] in average. This is discussed in detail in the Investigations more on hydrographs, SCA and SWE graphs section. The PTHSK and PTSSK SWE values are almost equal. Some cells show big ranges whereas some have smaller ranges. Big ranges show big differences between points observed and shows bigger roughness on that cell. Some phenomena are controlling the variability of the observed points such as; slope, aspects, and wind blowing. In figure (1.a) almost all simulated SWE for the three methods are in the range except inside the cells which their ranges are too small.

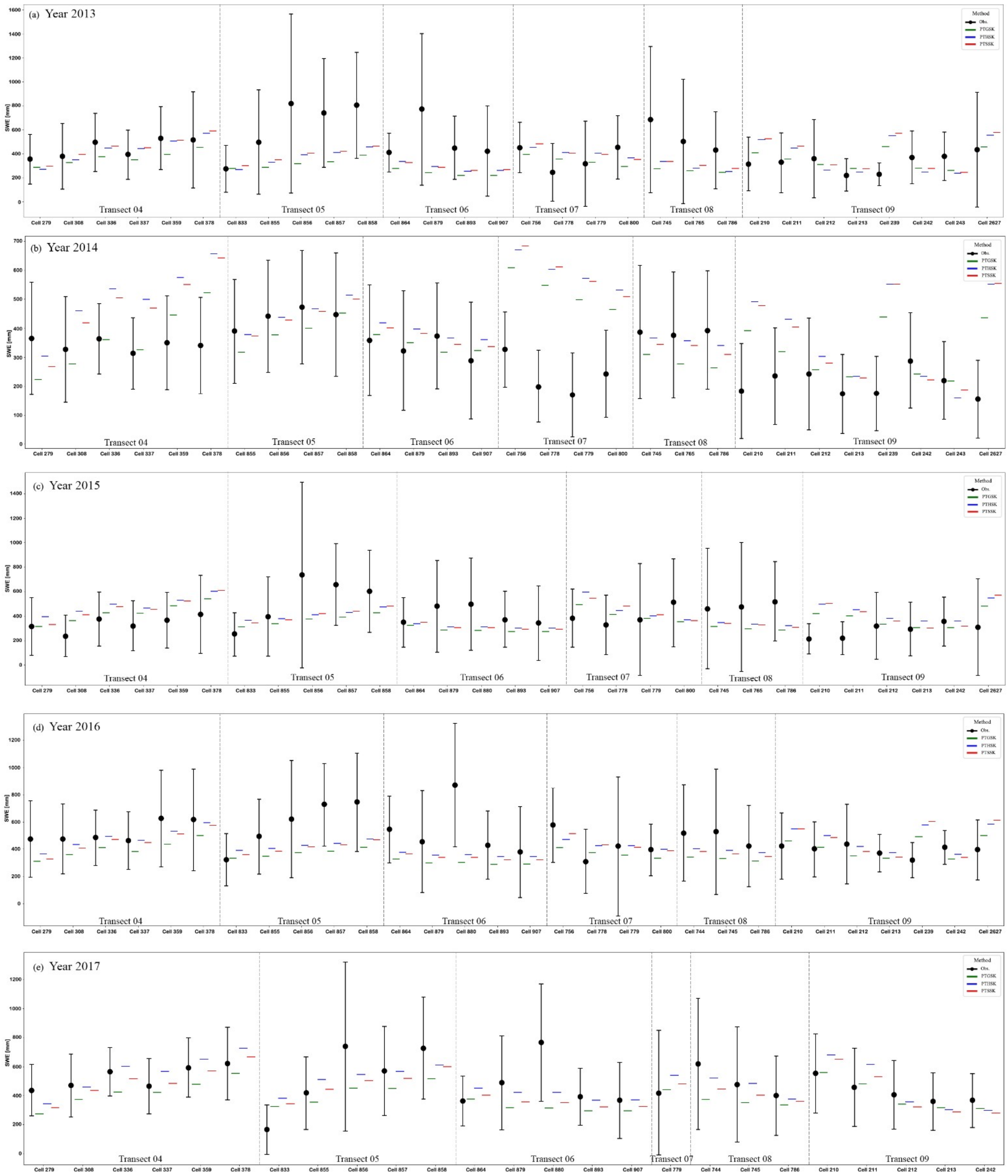


Figure 1 Observed and simulated SWE in all measured cells in five consecutive years

figure (1.b) in 2014 illustrates the worst case when compared to the others. The ranges of all observed SWE are smaller than other years while the terrain characteristics did not change. The CV (Coefficient of Variation) for all cells are smaller than other years. The CV of SWE is relatively high in the first days of accumulation and decreases during accumulation season (Skaugen 2007). The wind redistribution driving force also changes the character of mountainous seasonal snow pack in point SWE observation by a large variation. (Kolberg and Gottschalk 2006). So wind redistributes the snow by filling up the pits and ditch and stripping from regional highland. This is probably caused by less wind blowing in that year resulting in less snow redistribution and smaller observed ranges being the outcome but it is not possible to verify this hypothesis due to the lack of wind data.

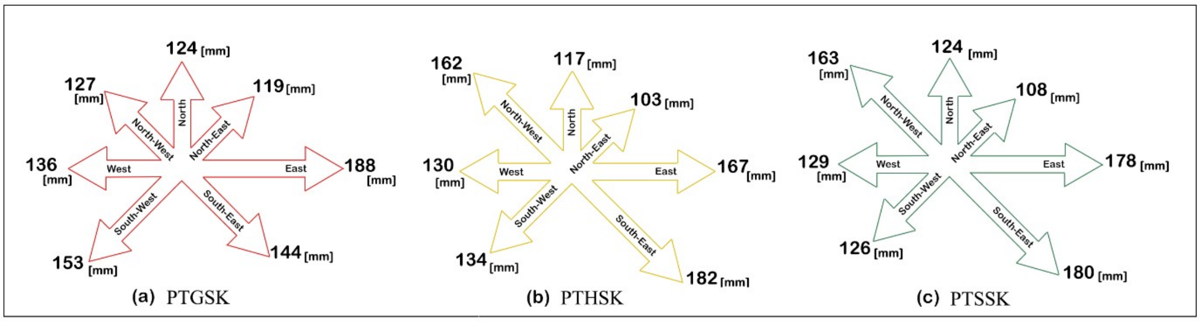


Figure 2 MEA of SWE on different aspects

3.5. MAE of SWE in different aspects

Figure (2) illustrates the Mean Absolute Error (MAE) of SWE which is the average absolute values of the difference between the observed and simulated SWE in different orientations and various methods in the whole period (2012-2015). There is no south orientation in the observed data. A similar pattern from figure (1) is observed in figure 2. PTGSK is different from PTHSK and PTSSK and the latter two methods produce similar results. The PTHSK and PTGSK show large MAE values in the line north-west and south-east and less MAE values on north-east and north compared to the other orientations while PTGSK has no significant difference between various aspects except on the east orientation. The East orientation shows, in overall, the largest MAE value in all the methods. It obvious that the PTGSK is less sensitive on different aspects in comparison with the two other methods.

3.6. Relationship of various parameters and SWEAE:

Figure 3 gives a graphical illustration for the relationship between Absolute Error of SWE (SWEAE) and various parameters for the methods. The relationship between terrain characteristics and snow cover distribution values are studied in many papers. (Marchand and Killingtveit 2001). Values for the Pearson Correlation Coefficient (PCC) are included to describe the linear relationship between two variables in all figures. The first three scatter plots (a, b, c) show positive slope of SWEAE against elevation-gradient for three methods. It also conforms to the supposition that snow mass drift in slopes and the SWE changes with elevation. The PCCs show a clear increase of SWEAE in graphs b) and c) but a slight increase for PTGSK method. These three graphs show less sensitivity of PTGSK to the slope and present better results on different slopes. Despite this study's observations it is reported that elevation gradient accounts for a large uncertainty in the Gamma snow routine. (Kolberg and Gottschalk 2010). In another study, the relationship between slope and snow depth was investigated and a low correlation was reported (Marchand and Killingtveit 2001). The second row of figure 3 illustrates three scatter graphs of SWEAE and elevation. It is much similar to the elevation-gradient and depicts less sensitivity of PTGSK to elevation in comparison to the other methods. Even though the sensitivity of SWEAE to elevation is more when compared to the elevation-gradient of the other methods, it is more clear for PTGSK. The third row-graphs shows the relationship between CV (terrain roughness) and SWEAE for three methods. The CV is a suitable mean for comparing of uncertainty of different parameters. (Iskra and Droste 2008). It can be observed that as the wind changes the snow mass by making its top surface flat surface, the coarser terrain shows a high SWE in different locations, therefore the CV represents the terrain roughness. In all of the three graphs (g, h, i), the relation may be characterized as close to linear and explain the small values of PCC for all three methods. The last graphs-row demonstrates the relationship of SWEAE to the number of observed points in a cell. The negative slope shows less SWEAE and high accuracy for more observed points. Once again the PTHSK and PTSSK are more sensitive to this variable while the PTGSK method displays more stability of SWEAE.

3.7. Investigations more on hydrographs, SCA and SWE graphs:

Figure 4a) display the hydrographs of three methods and differences between PTGSK and PTSSK methods during 2014 to 2016. The hydrographs of PTSSK and PTHSK are almost the same but PTGSK shows a different pattern. The black curve shows the differences between PTGSK and PTSSK. This curve shows the less and more simulated discharge in this period which must be seen in SCA and SWE in different days. Figure 4b) shows the SWE graph for three methods and it depicts different patterns of PTGSK in comparison with two other methods. To investigate whether the graphs (a, b) match and consonant, the graph c) was made. The blue curve shows the cumulative differences of PTGSK and PTSSK hydrograph and shows the differences in SWE. The differences in SWE influenced differences in the hydrograph. Figure 4d) shows the SCA of different methods. It shows more stability and smooth change for PTGSK method compared to other methods during snow accumulation and melting seasons. The PTGSK method models more the SCA value, especially in the melt season when the curves decline to zero. All the graphs are consonant with each other and indicate the PTGSK method having more snow in the melting season. The calculation of observed SWE of snow courses and SWE of models show a marginal difference between the three methods. The Mean Absolute Error of Snow Water Equivalent (MAESWE) in the whole period where found to be 133,131 and 129 mm for PTGSK, PTSSK and PTHSK respectively. The graphs show more disagreement between these methods.

3.8. Comparing the results with satellite images:

Figure 5 illustrates SWE and SCA of all methods and satellite images on different days. Figures (5b, 5d, 5f) show less SWE for PTHSK and PTSSK compared to PTGSK on the left part of catchment (near outlet of catchment, lowland). The figures (5a, 5c, 5e) show that PTGSK models more snow on the left part of catchment while the two other models show this part free of snow. This snow pack differences on the lowland are in consonant with figure 4 graphs. All the methods model almost the same SWE on the high mountain (where the snow courses are located) while do not generate the same values near the outlet. In order to establish more on which model generates results closer to the real world situation, the SCA results were compared with spatial resolution satellite images (MODIS-Moderate Resolution Imaging

Spectroradiometer) accessible from optical sensors. This was done to evaluate the goodness of SCA outputs of model. Remotely sensed SCA information may be valuable in a verification context. (Pan, Sheffield et al. 2003). It is obvious from the figures that the PTGSK method completely failed to generate a credible SCA figure while the two other methods show a better SCA simulation. The satellite image confirms that there is no snow cover near the outlet on specific dates. The images use a gray scale that generates more values between 0 to 1 to provide more color variation except yellow (value is one) and black (value is zero). The SCA of PTSSK method shows more color besides yellow and black and resembles more the satellite images which suggests that the PTSSK method generates a better SCA simulation compared to the other two methods with PTGSK showing the least accuracy. Many models for the spatial PDF of SWE such as gamma, normal, log normal and mixed log normal are presented in a number of literature. One of the mentioned PDF seems to be more suitable for the catchment in consideration. Physical process (variability of precipitation, wind before and after snowfall and topographic features) causes the diversity of distribution of SWE.(Skaugen and Weltzien 2016). It is reported that the Gamma snow routine simulates SCA better than Snow Distribution Long Normal (SD-LN) when it comes to MODIS image comparison while the latter model simulates SWE better and avoids the ‘snow tower’ effect as well as unrealistic positive SWE trend. (Skaugen and Weltzien 2016). The snow course observation in many fields has shown that the Gamma distribution shape changes continuously during accumulation and melting season.(Skaugen 2007).

It would be worthwhile to investigate whether PTGSK can be modified using different PDF or a time variable PDF for the accumulation and melting season to simulate the SWE and if this can generate better results than PTSSK and PTHSK.

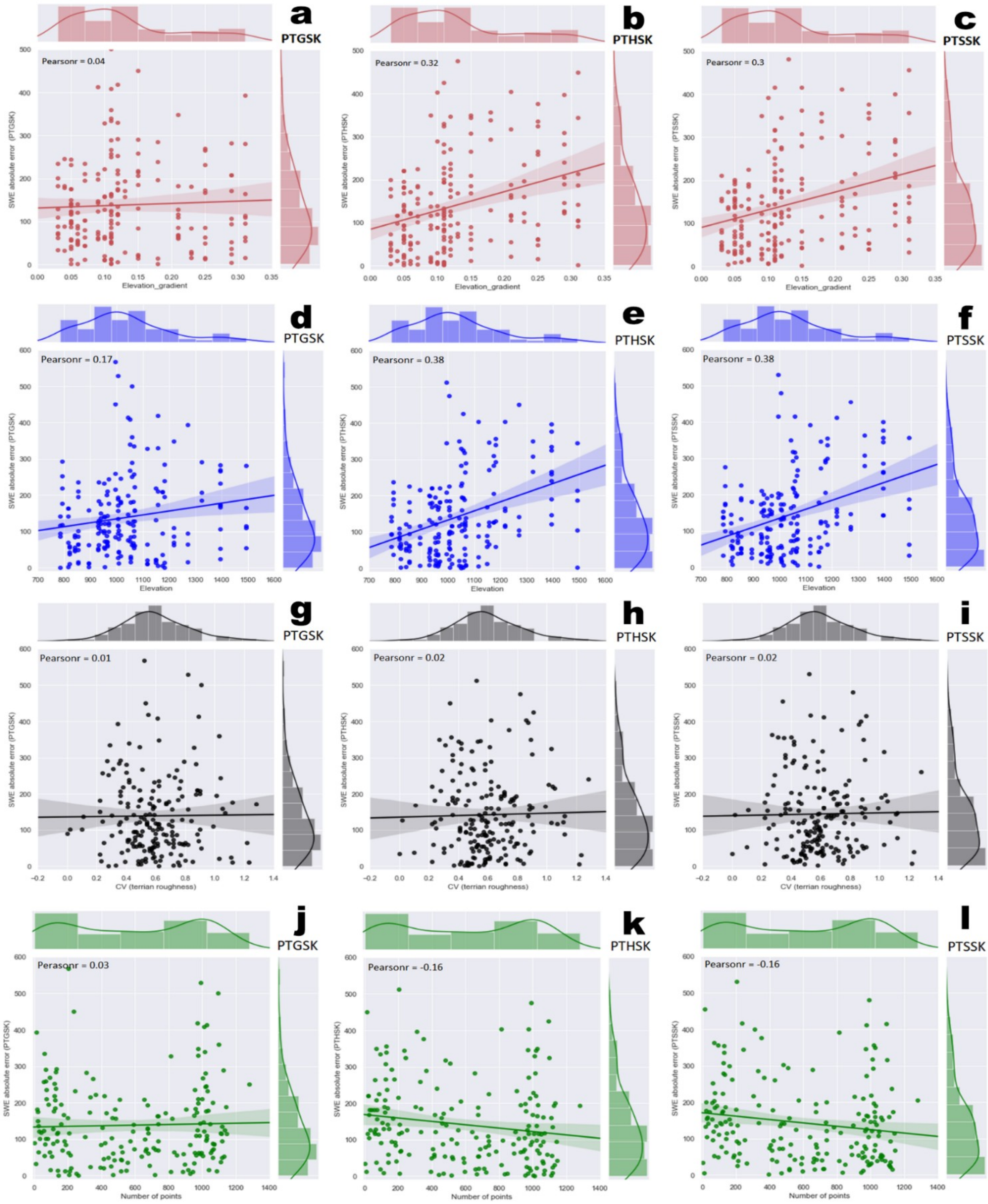


Figure 3 Relation of SWE to various parameters a,b,c) Elevation-gradient d,e,f) Elevation g,h,i) Terrain roughness j,k,l) Number of points

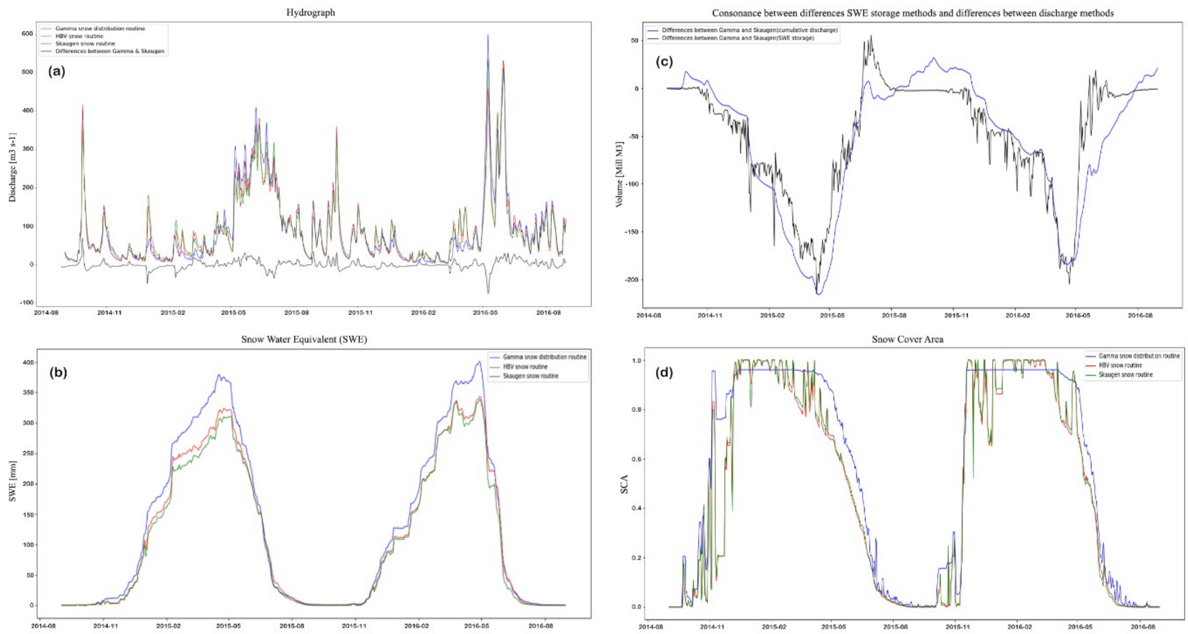


Figure 4 comparing of PTGSK method with two other methods

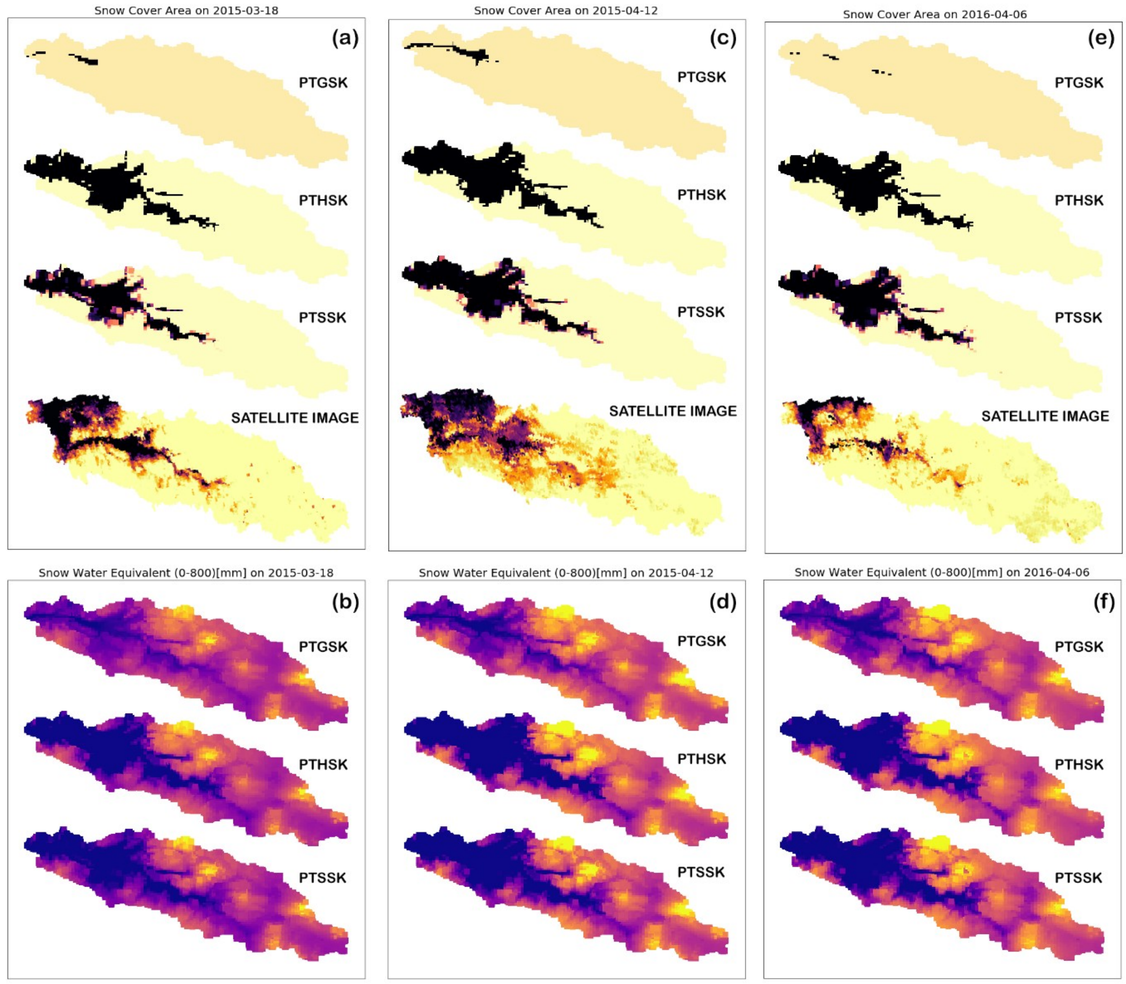


Figure 5 Snow Cover Area (SCA) and Snow Water Equivalent (SWE) and Satellite images

4. SUMMARY AND CONCLUSION

In this study, it was found that the PTHSK and PTSSK methods produced similar results. The simulated SWE images illustrate that PTGSK tends to model more snow on lowland while the two other methods do not. On the other hand, the total outflow was calibrated for all three methods, in order to compensate for the more snow on lowland, PTGSK method simulates less snow on highland where the snow courses are situated. The studied cell (The cell which has observed points) shows PTGSK method to have less SWE when compared to the two methods which are consonant with SWE images. Wind blowing is the main driving force to redistribute snow masses and the SWE observed shows low variability of SWE in comparison to other methods in the year 2014. It seems that there was less wind during the winter of that time. Calculations conducted on different orientations and MAE (Mean Absolute Error) show high and low errors on the east and west aspect in the all methods. The PTGSK method shows less sensitivity to different orientations in comparison with others while it presents less sensitivity to terrain characteristics such as; slope and elevation.

It is interesting that there is no clear relationship between terrain roughness and SWEAE (Absolute Error of Snow Water Equivalent) for all three methods. Despite the poor Pearson correlation coefficient of CV and snow absolute error the results are still significant for all the three methods. The more the number of observed SWE points in a cell the less the SWEAE and the better the results for PTSSK and PTHSK while there is less change in the SWEAE for the PTGSK method. The differences between total average of MAE in the three methods is less than 5 [mm] and the results are from snow courses on high elevations, however the methods did not generate similar SCA on the low lands. The MODIS show better similarity between PTHSK and PTSSK with satellite images. In order to simulate better with the PTGSK method, a unique PDF is needed due to the differences in terrain characteristics of each cell as well as the unique hydro-meteorological properties of catchments therefore some catchments match better with normal or log normal or even other PDF. Maybe possible to modify the PDF of Gamma snow distribution routine to simulate snow pack better than it is now.

5. ACKNOWLEDGMENTS

My sincere gratitude goes to Andrew Mabula, MSc. Hydro power development student at NTNU for his great support and help and Felix Nikolaus Matt, department of geoscience university Oslo and Sigbjørn Helset from Statkraft for their vital assistance in SHyFT.

6. REFRENCES

Al-Safi, H. I. J. and P. R. Sarukkalige (2017). "Potential climate change impacts on the hydrological system of the Harvey River catchment." World Academy of Science, Engineering and Technology, International Journal of Environmental, Chemical, Ecological, Geological and Geophysical Engineering **11**(4): 296-306.

Bergstr6m, S. (1997). "Development and test of the distributed HBV-96 hydrological model." Journal of hydrology **201**: 272-288.

Duan, Q., et al. (1992). "Effective and efficient global optimization for conceptual rainfall-runoff models." Water Resources Research **28**(4): 1015-1031.

Duan, Q., et al. (1994). "Optimal use of the SCE-UA global optimization method for calibrating watershed models." Journal of hydrology **158**(3-4): 265-284.

Iskra, I. and R. Droste (2008). "Parameter uncertainty of a watershed model." Canadian Water Resources Journal **33**(1): 5-22.

Kirchner, J. W. (2009). "Catchments as simple dynamical systems: Catchment characterization, rainfall-runoff modeling, and doing hydrology backward." Water Resources Research **45**(2).

Kolberg, S. and L. Gottschalk (2010). "Interannual stability of grid cell snow depletion curves as estimated from MODIS images." Water Resources Research **46**(11).

Kolberg, S. A. and L. Gottschalk (2006). "Updating of snow depletion curve with remote sensing data." Hydrological Processes **20**(11): 2363-2380.

Marchand, W.-D., et al. (2001). "Improved Measurements and Analysis of Spatial Snow Cover by Combining a Ground Based Radar System With a Differential Global Positioning System ReceiverPaper presented at the Nordic Hydrological Conference (Uppsala, Sweden–June, 2000)." Hydrology Research **32**(3): 181-194.

Marchand, W.-D. and A. Killingtveit (2001). Analyses of the relation between spatial snow distribution and terrain characteristics. Proceedings of the 58th Eastern snow conference, Citeseer.

Nash, J. E. and J. V. Sutcliffe (1970). "River flow forecasting through conceptual models part I—A discussion of principles." Journal of hydrology **10**(3): 282-290.

Pan, M., et al. (2003). "Snow process modeling in the North American Land Data Assimilation System (NLDAS): 2. Evaluation of model simulated snow water equivalent." Journal of Geophysical Research: Atmospheres **108**(D22).

Priestley, C. and R. Taylor (1972). "On the assessment of surface heat flux and evaporation using large-scale parameters." Monthly weather review **100**(2): 81-92.

Rinde, T. (1996). PINE–a hydrological model with flexible model structure. Proceedings from the Nordic Hydrological Conference.

Sand, K. and O. Bruland (1998). "Application of Georadar for Snow Cover SurveyingPaper presented at the 11th Northern Res. Basins Symposium/Workshop Prudhoe Bay to Fairbanks, Alaska, USA–Aug. 18-22, 1997." Hydrology Research **29**(4-5): 361-370.

Seibert, J. (2000). "Multi-criteria calibration of a conceptual runoff model using a genetic algorithm." Hydrology and Earth System Sciences Discussions **4**(2): 215-224.

Seity, Y., et al. (2011). "The AROME-France convective-scale operational model." Monthly weather review **139**(3): 976-991.

Shrestha, A. B. and R. Aryal (2011). "Climate change in Nepal and its impact on Himalayan glaciers." Regional Environmental Change **11**(1): 65-77.

Skaugen, T. (2007). "Modelling the spatial variability of snow water equivalent at the catchment scale." Hydrology and Earth System Sciences Discussions **11**(5): 1543-1550.

Skaugen, T. and C. Onof (2014). "A rainfall-runoff model parameterized from GIS and runoff data." Hydrological Processes **28**(15): 4529-4542.

Skaugen, T. and I. H. Weltzien (2016). "A model for the spatial distribution of snow water equivalent parameterized from the spatial variability of precipitation." The Cryosphere **10**(5): 1947-1963.

Appendices

Appendix 1 - Python script for Snow course calculation

Appendix 2 – How does SHyFT work?

Appendix 3 – YAML files

Appendix 4 – Calibration codes

Appendix 5 – Simulation codes

Appendix 6 – Miscellaneous codes

Appendix 7 – Calibration results

Appendix 8 – Summary of SWE calculations

Appendix 9 – Calibrated and validated Hydrographs

Appendix 10 – Graphs code in Seaborn (Python)

Appendix 11 – miscellaneous graphs

Appendix 12 – YouTube movie

Appendix 13 – Satellite images

All codes can be accessed on GitHub:

https://github.com/amirnk/master_thesis

Appendix 1

Python script for Snow course calculation

Python script for Snow course calculation

Description: In this Python Script, a number of modules were first imported which included, Pandas for DataFarme, Numpy for calculation, Matplotlib to make graphs, as well as OS and deepcopy. X, Y coordinates and SWE of all the snow course lines from 2013 to 2017 were read from some CSV files followed by the catchment's X, Y and Z.

The distances between all snow course line points and the center points of all cells were determined. Each snow course line point was allocated to the cell in which it was nearest to. The snow course line points of a cell were then categorized into an individual group in which the SWE average, minimum, maximum and standard deviation calculated. The date of doing the snow course was read off and the SWE of that cell was computed in order to compare with the SWE average from snow course points in that cell.

The elevations of neighboring cells were read with subsequent calculation of the elevation gradient of specific cells that had at least one snow course point. Three graphs where plotted with the first showing the catchment shape and the snow course line position and its length.

The second graph shows the snow course lines and the center of the cells with dots. It should be noted that those that do not have snow course line points are marked with a red dot while for those that have, a black dot was used and showing their boundaries. All latter cells include the cell No., the number of snow course line points, average, maximum, minimum, standard deviation, the elevation gradient and the orientation slope with an arrow and the SWE of that.

The third graph shows the SWE histogram of the snow course line and the SWE of passed cells. Finally, write all this values in a CSV file. The Python code and the three graphs described are presented as follows.

```
import pandas as pd
import numpy as np
import os
from matplotlib import pyplot as plt
from copy import deepcopy

for snowcourse1 in range(4,10):
    for snowcourse2 in range(2013,2018): # years 2013, 2014, 2015, 2016,
2017
        snow_course_pd = pd.DataFrame()
        all_swe_pd = pd.DataFrame()
        snow_course_file = r"D:\Dropbox\Thesis\Nea snowradar
transects\NE0" + str(snowcourse1) + "_" + str(snowcourse2) + ".csv"

        # reading the simulated SWE data in all cells which is given out
by simulation
        all_swe_pd = pd.read_csv(r'D:\Dropbox\Thesis\Nea snowradar
transects\SWE_pd_18_G.csv')

        # get the file name without extension
        file_name = snow_course_file.split('\\')[-1].split(".")[-2]

        # set the current directory to the where read the 'snow course
file'
        os.chdir(os.path.dirname(snow_course_file))

        # make DataFrame for snow course
        snow_course_pd = pd.read_csv(snow_course_file)

        # the date of doing snow course
        if file_name == 'NE01_2013' or file_name == 'NE02_2013' or
file_name == 'NE03_2013': date_get = '11-Apr-13'
        elif file_name == 'NE04_2013' or file_name == 'NE05_2013' or
file_name == 'NE06_2013' or file_name == 'NE07_2013' or file_name ==
'NE08_2013' or file_name == 'NE09_2013': date_get = '10-Apr-13'

        elif file_name == 'NE01_2014' or file_name == 'NE02_2014' or
file_name == 'NE04_2014': date_get = '2-Apr-14'
        elif file_name == 'NE03_2014': date_get = '9-Apr-14'
        elif file_name == 'NE05_2014' or file_name == 'NE06_2014' or
file_name == 'NE07_2014' or file_name == 'NE08_2014' or file_name ==
'NE09_2014': date_get = '1-Apr-14'

        elif file_name == 'NE01_2015' or file_name == 'NE03_2015' or
file_name == 'NE04_2015' or file_name == 'NE09_2015': date_get = '9-Apr-
15'
        elif file_name == 'NE02_2015' or file_name == 'NE05_2015' or
file_name == 'NE06_2015' or file_name == 'NE07_2015' or file_name ==
'NE08_2015': date_get = '10-Apr-15'

        elif file_name == 'NE01_2016' or file_name == 'NE03_2016' or
file_name == 'NE04_2016' or file_name == 'NE09_2016': date_get = '4-Apr-
16'
        elif file_name == 'NE02_2016' or file_name == 'NE06_2016' or
file_name == 'NE07_2016': date_get = '11-Apr-16'
        elif file_name == 'NE05_2016' or file_name == 'NE08_2016':
date_get = '5-Apr-16'
```

```
        elif file_name == 'NE01_2017' or file_name == 'NE02_2017' or
file_name == 'NE03_2017' or file_name == 'NE04_2017': date_get = '7-Mar-
17'
        elif file_name == 'NE05_2017' or file_name == 'NE06_2017' or
file_name == 'NE07_2017' or file_name == 'NE08_2017' or file_name ==
'NE09_2017': date_get = '8-Mar-17'

        # to read the simulated SWE form a CSV file
        if date_get == '11-Apr-13': cell_swe_no = 228
        elif date_get == '10-Apr-13': cell_swe_no = 227

        elif date_get == '1-Apr-14': cell_swe_no = 583
        elif date_get == '2-Apr-14': cell_swe_no = 584
        elif date_get == '9-Apr-14': cell_swe_no = 591

        elif date_get == '9-Apr-15': cell_swe_no = 956
        elif date_get == '10-Apr-15': cell_swe_no = 957

        elif date_get == '4-Apr-16': cell_swe_no = 1317
        elif date_get == '5-Apr-16': cell_swe_no = 1318
        elif date_get == '11-Apr-16': cell_swe_no = 1324

        elif date_get == '7-Mar-17': cell_swe_no = 1654
        elif date_get == '8-Mar-17': cell_swe_no = 1655

        cells_x_np22 = np.array(all_swe_pd.loc[1:]['2'])
        cells_xs = []
        for item in cells_x_np22:
            cells_xs.append(float(item))
        cells_x_np = np.array(cells_xs)

        cells_y_np22 = np.array(all_swe_pd.loc[1:]['3'])
        cells_ys = []
        for item in cells_y_np22:
            cells_ys.append(float(item))
        cells_y_np = np.array(cells_ys)

        path_x_np = np.array(snow_course_pd[:]['X'])
        path_y_np = np.array(snow_course_pd[:]['Y'])
        swe_np = np.array(snow_course_pd[:]['SWE'])
        swe = list(swe_np)

        cells, path=[], []

        for i in range(len(cells_x_np)):
            cells.append((cells_x_np[i],cells_y_np[i]))

        for i in range(len(path_x_np)):
            path.append((path_x_np[i],path_y_np[i]))

        snow_course_length = 0
        for i in range(len(path_x_np)-1):
            dd = ((path_x_np[i]-path_x_np[i+1])**2 + (path_y_np[i]-
path_y_np[i+1])**2)**0.5
            snow_course_length += dd
        snow_course_length = round(snow_course_length,1)
        print('Lenght of snow course:\t\t ', snow_course_length, "meters")
        print('Number of cells in the catchment:', cells_x_np.size)
```

```
        list1, distance =[], []

        for i in range(len(path)):
            for j in range(len(cells)):
                list1.append((((path[i][0])-
(cells[j][0]))**2+((path[i][1])-(cells[j][1]))**2)**0.5)
            distance.append(list1)
            list1 = []

        distance_pd = pd.DataFrame(distance)
        distance2_pd = distance_pd.transpose()
        first = distance_pd[:][0]
        cell_close_no = []
        for i in range(first.size):

cell_close_no.append(list(distance2_pd[i][:]).index(distance2_pd[i][:].min
()))

        list_close_cell = list(set(cell_close_no))
        list_close_cell.sort()

        all_cat, averageif, std_swe, min_swe, max_swe, cv_swe,
no_swe,forprint = [],[],[],[],[],[],[], []

        for j in range(len(set(cell_close_no))):
            sum, counter = 0,0
            all_cat.append([])
            for i in range(len(cell_close_no)):
                if list_close_cell[j] == cell_close_no[i]:
                    sum += swe[i]
                    counter += 1
                    all_cat[j].append(swe[i])
            averageif.append(sum/counter)
            new_np = np.array(all_cat[j])
            std_swe.append(new_np.std())
            min_swe.append(new_np.min())
            max_swe.append(new_np.max())
            cv_swe.append(new_np.std()/(sum/counter))
            no_swe.append(counter)
            forprint.append([])
            forprint[j].append(list_close_cell[j])
            forprint[j].append(counter)
            average_1 = sum/counter
            forprint[j].append(round(average_1,2))
            forprint[j].append(round(new_np.std(),2))
            forprint[j].append(round(new_np.min(),2))
            forprint[j].append(round(new_np.max(),2))

        for i in range(len(cell_close_no)-len(list_close_cell)):
            list_close_cell.append(0)
            averageif.append(0)
            std_swe.append(0)
            min_swe.append(0)
            max_swe.append(0)
            cv_swe.append(0)
            no_swe.append(0)
```

```
        snow_course_pd['inside_cell'] = cell_close_no
        snow_course_pd['Cell No.'] = list_close_cell
        snow_course_pd['AverageIf'] = averageif
        snow_course_pd['Minimum'] = min_swe
        snow_course_pd['Maximum'] = max_swe
        snow_course_pd['Standard Deviation'] = std_swe
        snow_course_pd['CV'] = cv_swe
        snow_course_pd['No.'] = no_swe
        snow_course_pd.to_csv(f'{file_name}_cell_close_no.csv')

        fig, ax1 = plt.subplots(figsize=(25,9))

        for item in ([ax1.title, ax1.xaxis.label, ax1.yaxis.label] +
ax1.get_xticklabels() + ax1.get_yticklabels()):
            item.set_fontsize(12)

        close = []
        for i in range(len(cells_x_np)):
            close.append(distance_pd[:][i].min())

        cm = plt.cm.get_cmap('tab20c')
        ax1.scatter(cells_x_np, cells_y_np, c=close, marker='o', s=220,
lw=0, cmap=cm, alpha = 0.9)

        ax1.plot(path_x_np, path_y_np, lw = 1.4, color = 'black',
label=f'Snow course line ({snow_course_length} Meters)')

        file_name = snow_course_file.split('\\')[-1].split(".")[-2]

        plt.title(f"Snow course on the catchment layout ({date_get}):
{file_name}", fontsize = 14)
        plt.xlabel('X coordinade')
        plt.ylabel('Y coordinade')
        ax1.legend(loc=1, fontsize = 14)

        plt.savefig(f"{file_name}_1.png")
        plt.show()

        draw_all_cells = 'no' # 'yes' or 'no'
        range_cell = 650
        font_size = 14

        cells_x_2, cells_y_2 = [],[]

        for i in range (len(cells_x_np)):
            if cells_x_np[i] > path_x_np.min()-range_cell and
cells_x_np[i] < path_x_np.max()+range_cell:
                if cells_y_np[i] > path_y_np.min()-range_cell and
cells_y_np[i] < path_y_np.max()+range_cell:
                    cells_x_2.append(cells_x_np[i])
                    cells_y_2.append(cells_y_np[i])

        cells_x_np2 = np.array(cells_x_2)
        cells_y_np2 = np.array(cells_y_2)

        list1 = []
        for i in range(len(cells_x_np2)):
            for j in range(len(cells_x_np)):
```

```
                    if cells_x_np2[i] == cells_x_np[j]:
                        if cells_y_np2[i] == cells_y_np[j]:

list1.append([cells_x_np2[i],cells_y_np2[i],j])

        status = 0
        for i in range(len(list1)):
            for j in range(len(forprint)):
                if list1[i][2] == forprint[j][0]:
                    list1[i].append(forprint[j][1])
                    list1[i].append(forprint[j][2])
                    list1[i].append(forprint[j][3])
                    list1[i].append(forprint[j][4])
                    list1[i].append(forprint[j][5])
                    status +=1
            if status == 0:
                list1[i].append(0)
                list1[i].append(0)
                list1[i].append(0)
                list1[i].append(0)
                list1[i].append(0)
            status = 0

        # make a deep copy of list1
        list2 = deepcopy(list1)
        cell_list = []

        for i in range(cells_x_np.size):
            cell_list_temp =
[float(all_swe_pd.loc[i+1][1]),float(all_swe_pd.loc[i+1][2]),float(all_swe
_pd.loc[i+1][3])]
            cell_list.append(cell_list_temp)

        for i in range(len(list2)):
            for j in range(len(cell_list)):
                if int(list2[i][0]) == int(cell_list[j][0]):
                    if int(list2[i][1]) == int(cell_list[j][1]):
                        list2[i].append(cell_list[j][2])

        list_x_new, list_y_new, list_z_new, distan_new, gradia_new = [],
[], [], [], []

        for i in range(len(list2)):
            for j in range(len(cell_list)):
                if int(cell_list[j][0]) < int(list2[i][0]) + 1500 and
int(cell_list[j][0]) > int(list2[i][0])-1500:
                    if int(cell_list[j][1]) < int(list2[i][1]) + 1500 and
int(cell_list[j][1]) > int(list2[i][1])-1500:
                        list_x_new.append(cell_list[j][0])
                        list_y_new.append(cell_list[j][1])
                        list_z_new.append(cell_list[j][2])

            for l in range(len(list_x_new)):
                dis = ((list2[i][0]-list_x_new[l])**2 + (list2[i][1]-
list_y_new[l])**2)**0.5
                if dis == 0:
                    gradian = 0
                    gradia_new.append(gradian)
```

```
                    distan_new.append(dis)
                    continue
                gradian = (list2[i][8] - list_z_new[l]) / dis
                distan_new.append(dis)
                gradia_new.append(gradian)
            gradia_new_np = np.array(np.abs(gradia_new))

            gr = 0
            for ii in range(len(gradia_new_np)):
                if gradia_new[ii] > gr:
                    gr = gradia_new_np[ii]
                    x_compare = list_x_new[ii]
                    y_compare = list_y_new[ii]

            if list2[i][0] == x_compare:
                if list2[i][1] < y_compare:
                    o ="North"
                elif list2[i][1] > y_compare:
                    o ="South"
            elif list2[i][0] < x_compare:
                if list2[i][1] < y_compare:
                    o ='North-East'
                elif list2[i][1] == y_compare:
                    o ='Easth'
                else:
                    o ='South-East'
            elif list2[i][0] > x_compare:
                if list2[i][1] < y_compare:
                    o ='North-West'
                elif list2[i][1] == y_compare:
                    o ='West'
                else:
                    o ='South-West'

        #    list2[i].append(int(gradia_new_np.mean()*100)) # take the
average slope of the cells
            list2[i].append(int(gradia_new_np.max()*100)) # take the
maximum slope of the cells
            list2[i].append(o)
            list_x_new, list_y_new, list_z_new, distan_new, gradia_new =
[], [], [], [], []

        grad = []
        orientation = []
        for i in range(len(list2)):
            if list2[i][3]!=0:
                grad.append(list2[i][9])
                orientation.append(list2[i][10])

        for i in range(len(cell_close_no)-len(grad)):
            grad.append(0)
            orientation.append(0)

        snow_course_pd['Elevation gradient'] = grad
        snow_course_pd['Orientation'] = orientation

        snow_course_pd.to_csv(f'{file_name}_cell_close_no.csv')
```

```
        for i in range(len(list2)):
            if list2[i][3] != 0:
                
list2[i].append(all_swe_pd.loc[list2[i][2]+1][cell_swe_no])
            else:
                list2[i].append(0)

        fig, ax = plt.subplots(figsize=(int((max(cells_x_2) -
min(cells_x_2)+1000)/270),
                                        int((max(cells_y_2) -
min(cells_y_2) + 1000)/270)))

        for item in ([ax.title, ax.xaxis.label, ax.yaxis.label] +
ax.get_xticklabels() + ax.get_yticklabels()):
            item.set_fontsize(font_size)

        label_stat1, label_stat2 = True, True

        for d in range(len(list1)):
            if list1[d][3] == 0:
                if label_stat1:
                    plot = ax.scatter(list1[d][0], list1[d][1],
marker='o', s=60, lw=0, color = 'red',
                                      label = 'snow course doesn`t pass
this cell')
                    label_stat1 = False
                else:
                    plot = ax.scatter(list1[d][0], list1[d][1],
marker='o', s=60, lw=0, color = 'red')
            else:
                if label_stat2:
                    plot = ax.scatter(list1[d][0], list1[d][1],
marker='o', s=60, lw=0, color = 'Black',
                                      label = 'snow course passes this
cell')
                    label_stat2 = False
                else:
                    plot = ax.scatter(list1[d][0], list1[d][1],
marker='o', s=60, lw=0, color = 'Black')

        plt.plot(path_x_np[0], path_y_np[0], lw = 4, color = 'green',
marker='_',  alpha = 0.5,
                    label =f'Snow course ({snow_course_length} Meters)' )
        plt.scatter(path_x_np, path_y_np, marker='.', s=100, lw=0, color =
'green', alpha = 0.03)

        if draw_all_cells == 'yes':
            draw_all_cells = -1
        else:
            draw_all_cells = 0

        temp_x, temp_y = [],[]

        label_stat3 = True

        for i in range(cells_x_np2.size):
            temp_x.append(list1[i][0]+500)
            temp_x.append(list1[i][0]+500)
```

```
            temp_x.append(list1[i][0]-500)
            temp_x.append(list1[i][0]-500)
            temp_x.append(list1[i][0]+500)
            temp_y.append(list1[i][1]+500)
            temp_y.append(list1[i][1]-500)
            temp_y.append(list1[i][1]-500)
            temp_y.append(list1[i][1]+500)
            temp_y.append(list1[i][1]+500)

            if list1[i][3] > draw_all_cells:
                if label_stat3:
                    plt.plot(temp_x, temp_y, lw = 4, color = 'black',
alpha = 0.2, label = "Confine a cell")
                    label_stat3 = False
                else:
                    plt.plot(temp_x, temp_y, lw = 4, color = 'black',
alpha = 0.2, label = "")

            temp_x, temp_y = [],[]

            for j in range(len(list1)):
                if cells_x_np2[i] == list1[j][0]:
                    if cells_y_np2[i] == list1[j][1]:

                        strcell = f'Cell No.: {list1[j][2]}'
                        if list1[j][3] == 0 : strcell = ""
                        plt.annotate(strcell, xy =(cells_x_np2[i]-450,
cells_y_np2[i]+410), fontsize = 12,
                                     color = "blue")
                        if list1[j][3] == 0:
                            continue
                        else:
                            if list2[j][10] == 'North':
                                xx = 0
                                yy = 350
                            elif list2[j][10] == 'South':
                                xx = 0
                                yy = -350
                            elif list2[j][10] == 'North-East':
                                xx = 250
                                yy = 250
                            elif list2[j][10] == 'Easth':
                                xx = 350
                                yy = 0
                            elif list2[j][10] == 'South-East':
                                xx = 250
                                yy = -250
                            elif list2[j][10] == 'North-West':
                                xx = -250
                                yy = 250
                            elif list2[j][10] == 'West':
                                xx = -350
                                yy = 0

                            elif list2[j][10] == 'South-West':
                                xx = -250
                                yy = -250
```

```
                            plt.annotate("", xy =(cells_x_np2[i]+xx,
cells_y_np2[i] + yy), fontsize = 12,
                                     arrowprops = dict(facecolor =
'olive', width =4, alpha = 0.6),
                                     xytext=(cells_x_np2[i],
cells_y_np2[i]),)

                        strcell = f'EL.grad. {list2[j][9]}%'
                        if list1[j][3] == 0 : strcell = ""
                        plt.annotate(strcell, xy =(cells_x_np2[i]-450,
cells_y_np2[i]+70), fontsize = 12,
                                     color = "olive")

                        strnumber = f'points: {list1[j][3]}'
                        if list1[j][3] == 0 : strnumber = ""
                        plt.annotate(strnumber, xy =(cells_x_np2[i]+20,
cells_y_np2[i]+410), fontsize = 12,
                                     color = 'darkgreen')

                        strnumber = f'{list2[j][10]}'
                        if list1[j][3] == 0 : strnumber = ""
                        plt.annotate(strnumber, xy =(cells_x_np2[i]+30,
cells_y_np2[i]+70), fontsize = 12,
                                     color = 'olive')

                        str_average_swe = f'Average SWE:
{int(list1[j][4])} (mm)'
                        if list1[j][3] == 0 : str_average_swe = ""
                        plt.annotate(str_average_swe, xy =(cells_x_np2[i]-
450, cells_y_np2[i]-260), fontsize = 12,
                                     color = 'green')

                        str_std = f'  {int(list1[j][5])}, '
                        if list1[j][3] == 0 : str_std = ""
                        plt.annotate(str_std, xy =(cells_x_np2[i]-120,
cells_y_np2[i]+200), fontsize = 12,
                                     color = 'black')

                        str_min = f'    {int(list1[j][6])}]'
                        if list1[j][3] == 0 : str_min = ""
                        plt.annotate(str_min, xy =(cells_x_np2[i]+200,
cells_y_np2[i]+200), fontsize = 12,
                                     color = 'black')

                        str_max = f'[{int(list1[j][7])}, '
                        if list1[j][3] == 0 : str_max = ""
                        plt.annotate(str_max, xy =(cells_x_np2[i]-450,
cells_y_np2[i]+200), fontsize = 12,
                                     color = 'black')

                        str_maxminstd1 = f'[Max.         Std.
Min.]'
                        if list1[j][3] == 0 : str_maxminstd1 = ""
                        plt.annotate(str_maxminstd1, xy =(cells_x_np2[i]-
450, cells_y_np2[i]+310), fontsize = 12,
                                     color = 'black')

                        obs = float(list1[j][4])
```

```
                        sim = float(list2[j][11])
                        accuracy = int((1-abs((obs-sim)/obs))*100)
                        str_accuracy = f'Accuracy: {accuracy} %'
                        if list1[j][3] == 0 : str_accuracy = ""
                        plt.annotate(str_accuracy, xy =(cells_x_np2[i]-
450, cells_y_np2[i]-400), fontsize = 12,
                                     color = 'black')

                        model_swe = round(float(list2[j][11]),2)
                        str_max = f'Model SWE: {int(model_swe)} (mm)'
                        if list1[j][3] == 0 : str_max = ""
                        plt.annotate(str_max, xy =(cells_x_np2[i]-450,
cells_y_np2[i]-120), fontsize = 12,
                                     color = 'deeppink')

        if int((max(cells_x_2) - min(cells_x_2))/220) < 12:
            plt.title(f"Snow course over cells grid\n({date_get}) file:
{file_name}", fontsize = font_size + 6)
        else:
            plt.title(f"Snow course over cells grid ({date_get}) file:
{file_name}", fontsize = font_size + 6)
        plt.xlabel('X coordinade')
        plt.ylabel('Y coordinade')

        ax.legend(loc=0, fontsize = 14)

        plt.savefig(f"{snowcourse2}_{snowcourse1}.png")

        plt.show()

        list3 = deepcopy(list2)
        for i in range(len(list2)):
            if float(list2[i][4]) == 0:
                accuracy = 0
            else:
                obs = float(list2[i][4])
                sim = float(list2[i][11])
                accuracy = int((1-abs((obs-sim)/obs))*100)
            list3[i].append(accuracy)

        accuracy = []
        sim_model = []
        for i in range(len(list3)):
            if list2[i][3]!=0:
                accuracy.append(list3[i][12])
                sim_model.append(list3[i][11])

        for i in range(len(cell_close_no)-len(accuracy)):
            accuracy.append(0)
            sim_model.append(0)

        snow_course_pd['SWE Model'] = sim_model
        snow_course_pd['Accuracy'] = accuracy

        snow_course_pd.to_csv(f'{file_name}_cell_close_no.csv')

        list_cell_no, list_swe_model  = [], []
```

```
        for item in range(len(list1)):
            if list1[item][3]== 0:
                continue
            list_cell_no.append(list1[item][2])

        for num in range(len(list_cell_no)):

list_swe_model.append(all_swe_pd.loc[list_cell_no[num]+1][cell_swe_no])

        list_swe_model_flat = []
        for item in list_swe_model:
            item_str = str(item)
            list_swe_model_flat.append(int(item_str.split('.')[0]))

        fig, ((ax1, ax2)) = plt.subplots(nrows=1, ncols=2, figsize =
(15,6))
        ax1.hist(list_swe_model_flat, color='y', alpha=0.3)
        ax1.set_xlabel(f"Snow Water Equivalent (mm) of passed
cells\n{list_swe_model_flat}", fontsize=14)
        ax1.set_ylabel("frequency", fontsize=14)
        ax1.set_title(f"SWE Histogram of passed cells ({file_name})",
fontsize = font_size + 0)

        ax2.hist(swe, bins=50,  color='r', alpha=0.3)
        ax2.set_xlabel(f"Snow Water Equivalent (mm) of snow course\nMin:
{int(swe_np.min())}         Mean: {int(swe_np.mean())}         Max:
{int(swe_np.max())}", fontsize=14)
        ax2.set_ylabel("frequency", fontsize=14)
        ax2.set_title(f"SWE Histogram of snow course ({file_name}",
fontsize = font_size + 0)

        plt.savefig(f"{file_name}_3.png")

        plt.show()

print('_It is done_'*7)

# make a sound to notify that it is done
import winsound
for i in range(1400,3500,100):
    winsound.Beep(i, int(200*1500/i))
```

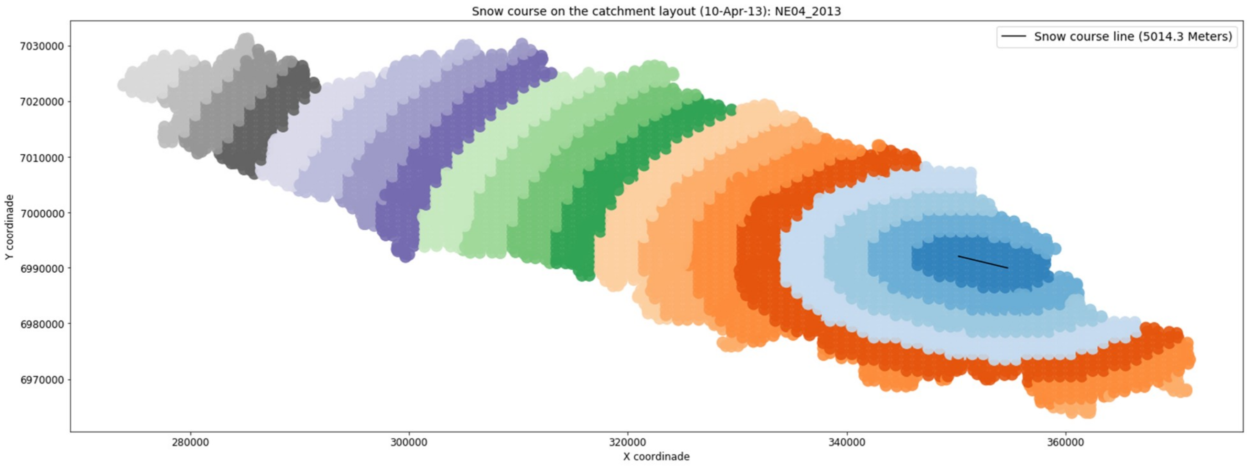


Figure Ap1.1 Snow course on the catchment layout

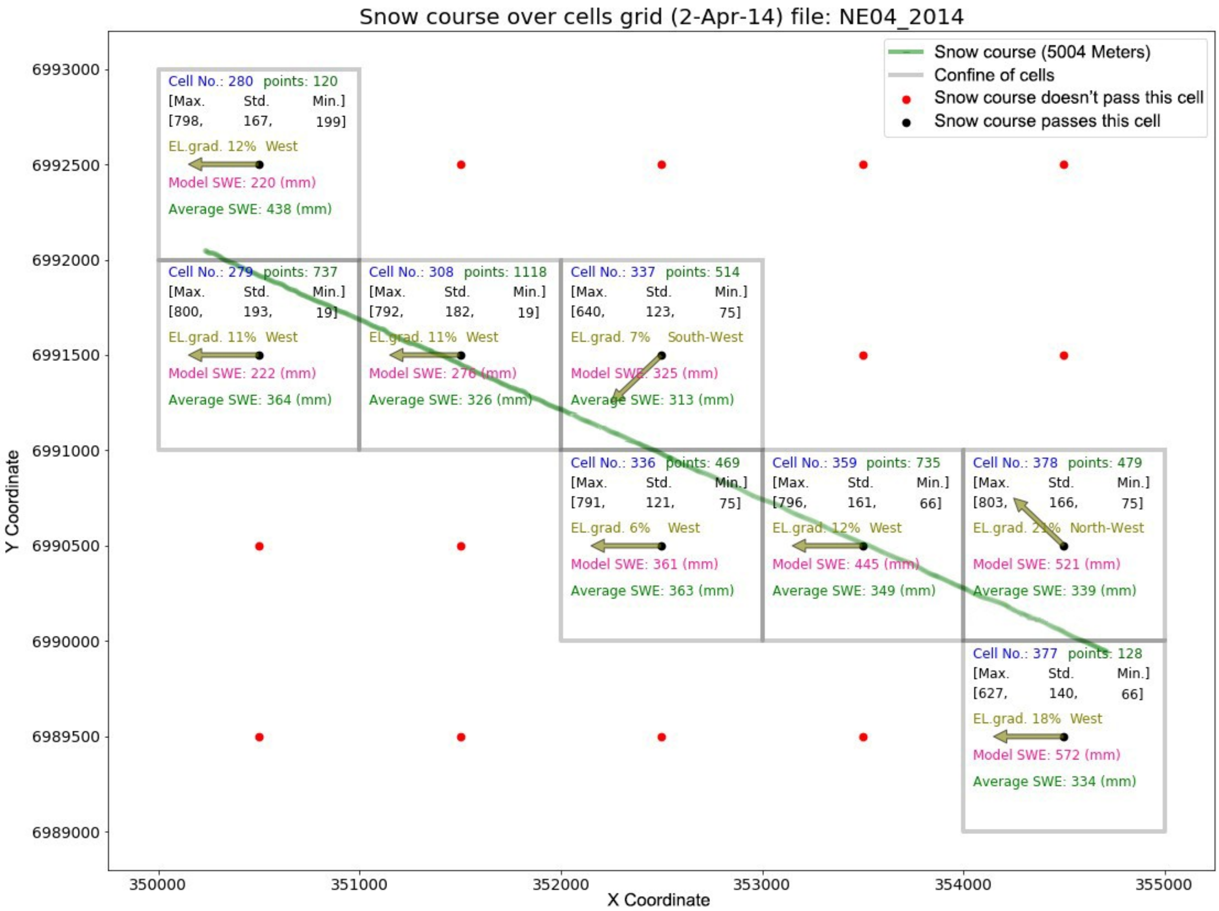


Figure Ap1.2 Snow course over cells grid 1

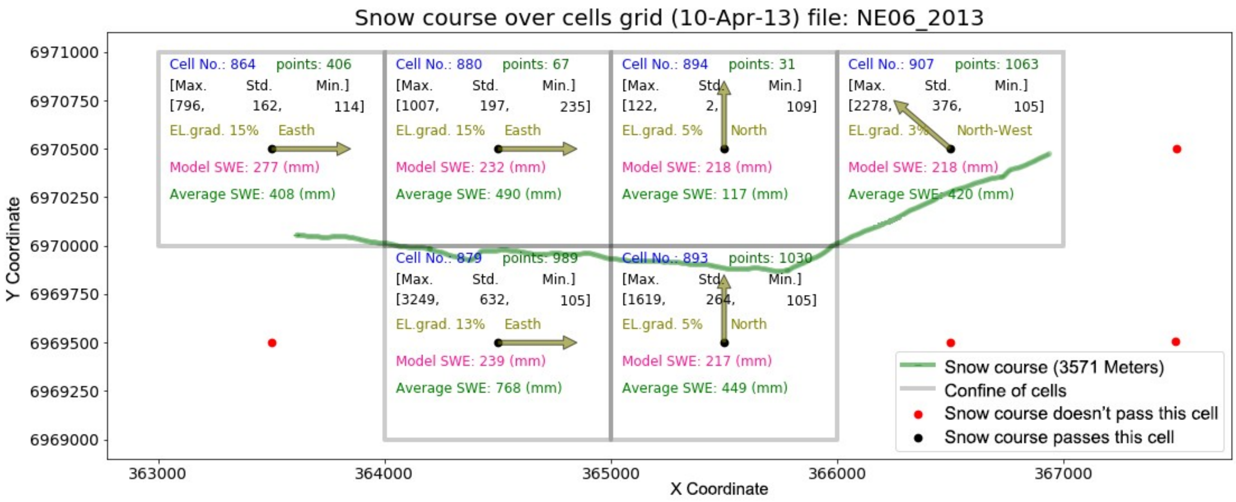


Figure Ap1.3 Snow course over cells grid 2

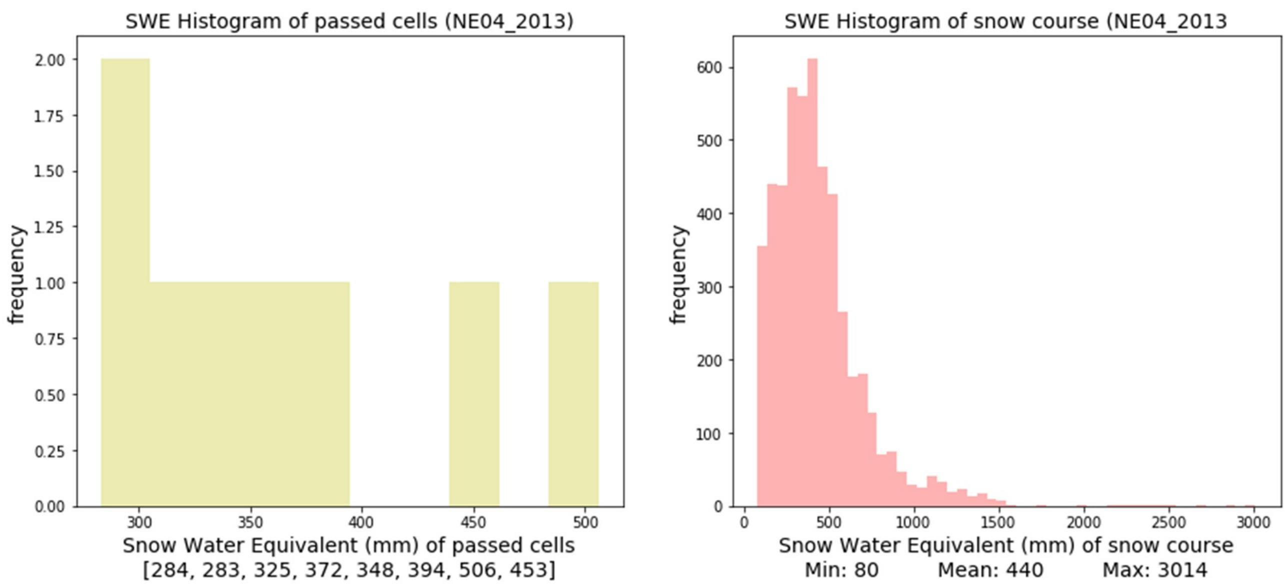


Figure Ap1.4 Snow water equivalent histogram

Appendix 2

How does SHyFT work?

How does SHyFT work?

1. SHyFT needs the region properties and hydro-meteorological data which are fed into it using NC files (NC is short term for NetCDF which stands for Network Common Data Frame). The NC files are:
    - Region properties (cell_data.nc), which is the whole catchment divided into small areas called cells, for example 1x1 kilometers (fishnet method). The file includes;
        i. EPSG (the UTM projection) that shows where the catchment is located. UTM stands for Universal Transverse Mercator.

        Which represents catchment IDs of every single cell (the main catchment is divided into small sub-catchments with unique Id and then categorized all cells to different sub-catchments), so each cell belongs just to one unique sub-catchment.

        ii. X and Y coordinates at the center of all the cells
        iii. Elevation of all cells (Z plane)
        iv. Area of all cells
        v. Reservoir fraction, lake fraction, forest fraction and glacier fraction of all cells with values between 0 to 1.

    <u>*The regional data are static and independent to the time but the following data changes with time and are represented in time series.*</u>

    - Precipitation (precipitation.nc): Contains precipitation data that is collected from a station(s), which can either be inside or outside the catchment. SHyFT uses these data to distribute the precipitation over the entire catchment by the use of distribution methods. This file includes the EPSG, how missing values are represented (e.g. -999), unit (e.g. mm/hr.), station(s) name, X and Y coordinates and elevation and precipitation of all the days in that period.
    - Temperature (temperature.nc): This is similar to the Precipitation. It includes EPSG data, how it shows missing values (e.g. -999), units (e.g. Celsius),

station(s) name, X, Y coordinates and elevation as well as temperature values in that period. (Like precipitation, the temperature values are distributed to all the catchment cells)

- Wind speed (wind_speed.nc): is similar to the temperature file but has wind speed instead of temperature values
- Global radiation (radiation.nc): Similar to temperature file but instead of temperature values it includes radiation values.
- Relative humidity (relative_humidity.nc): Similar to temperature file but instead of temperature values it includes relative humidity values.
- Discharge (discharge.nc): Similar to temperature file but has one more value which is the catchment ID, it shows that these discharge values are associate with which sub-catchment. This catchment ID shows that all cells of that sub-catchment drain to the relevant point and generate the discharge of that sub-catchment. In the cell data file cells are categorized and assigned to catchment IDs and in discharge file, the discharge of catchments is shown.
- In some cases, all precipitation, temperature, wind speed and relative humidity values are combined in one file which is called Arome stands as Application of Research to Operations at Mesoscale (AROME-France). So instead of four files just one Arome file. In this study we used AROME file too.
- To make the NC files, Using ArcGIS, physiographic, observed Hydro-meteorological all into Excel file then convert to tab delaminated text files and then with a Python script convert them to NC files.

2. SHyFT like other programs need to be introduced methods for simulation and also calibration. There is two ways to do that, first feed the SHyFT through command lines or put all methods and variables in some separate files and feed SHyFT theses files through command lines. For sure using these separate files are more convenient and straight forward and possible to use them for other simulations as many times as you want. These separate files are YAML files, YAML stands for Yet Another Markup Language. The YAML files are explained as follows:

- The YAML file is a *region. yaml* file which contains the required region data. It includes ESPG, the number of cells in X and Y coordinates, the dimensions of cells in X and Y directions and lower left of X and Y based on the UTM system. The IDs of all sub-catchments participate in the simulation and hence a limited number of cells in the mentioned sub-catchments are modeled in the simulation.
- The *model. yaml*, states which methods should be used, with the options of PTGSK (Priestley Taylor Gamma Snow Kirchner), PTHSK (Priestley Taylor HBV Snow Kirchner) or PTSSK (Priestley Taylor Skaugen Snow Kirchner). The file also gives the free variables of that model. It is however, impossible to collect values of free variables in a catchment at stations like temperature value, the mentioned models use the values of free variables for simulation. In the calibration part, the free variable values are modified to fit the simulation results with observed ones.
- The interpolation*. yaml* determines the interpolation method, BTK or IDW. BTK stands for Bayesian Temperature Kriging and IDW stands for Inverse Distance Weighting. The file shows how the input station data should be interpolated and assigns meteorological data to all the cells. In most cases it is better to use BTK for temperature interpolation and IDW for the precipitation, wind speed, radiation and relative humidity data.
- The *dataset. yaml* file shows the path and names of all NC files. NC files contain precipitation, temperature, wind speed, relative humidity and radiation data or AROME data. These files were introduced in the first section. In some new cases, the Arome NC file is used instead of the mentioned NC files.
- The *Simulation.yaml* file shows the name of simulation and the required yaml files. This file is the only file that is introduced in the simulation and SHyFT to find all relevant yaml files and to retrieve the required data. The start date and time, step times (in seconds) and number of steps are shown in this file and

also shows the method for the simulation (PTGSK, PTHSK or PTSSK). The path and name of discharge NC file is also shown, and lastly the target vectors. The target vectors are categorized sub-catchments in different groups with each group being a target vector. Each target vector is associated to a single meteorological station which means that all sub-catchments of that target vector drain to one point. SHyFT operates by distributing input data to all the cells and then draining out the rain fall and outflow of melted snow of each cells to the outlet of relevant sub-catchment. The water however does not go from cell to cell then though all cells are connected to one point (outlet of that sub-catchment). For each target value there is a station name and the discharge values of all stations are in discharge NC files.

- The yaml files are needed for simulation, but for calibration, another yaml file, *calibration. yaml* is required. The file first shows the calibration name followed by the calibrated parameters file after calibration. It also determines the optimization method (min_bobyqa, dream, sceua), all target values, start date-time, run time step, number of steps and the weight of target values as well as the parameters that should be calibrated (discharge) and the method to be used to compare the observed and simulated values. The two methods for comparing the simulated and observed values are NSE and KGE which stand for Nash-Sutcliffe Efficiency and Kling-Gupta Efficiency, respectively. It then determines the method (PTGSK, PTHSK or PTSSK) and the ranges of all parameters for calibration.

3. In summary, orchestration in SHyFT means ingestions of all observed hydro-meteorological data. In Jupyter Notebook, first run the RunShyft.py for simulation and then it gives out all distributed of precipitation, temperature, relative humidity, wind speed and radiation, also SWE (Snow Water Equivalent) SCA (Snow Cover Area) in separated CSV files. Also, execute the Calibshyft.py to calibrate the model and then gives out a file with all calibrated parameters. The simulation and calibration codes must be run for all three routines separately.

Appendix 3

YAML files

YAML files:

Every method in SHyFT requires six yaml files. Each method shares three yaml files consisting of the dataset.yaml, region.Yaml and interpolation.yaml. There are three other yaml files specific for each method which include model.yaml, simulation.yaml and calibration.yaml. An elaborate description on yaml files has been provided in appendix2.

PTGSK (Priestley Taylor Gamma Snow Kirchner)

PTHSK (Priestley Taylor HBV Snow Kirchner)

PTSSK (Priestley Taylor Skaugen Snow Kirchner).

Datasets.yaml (PTGSK, PTHSK, PTSSK)


```
---
sources:
  - repository:
!!python/name:shyft.repository.netcdf.concat_data_repository.ConcatData
Repository
    types:
      - precipitation
      - wind_speed
      - temperature
      - relative_humidity
      - radiation
    params:
      filename: netcdf/orchestration-testdata/arome_merged_all.nc
      nb_lead_intervals_to_drop: 0
      nb_lead_intervals: 1
      use_filled_values: true
...
```

Interpolation.yaml (PTGSK, PTHSK, PTSSK)


```yaml
interpolation_parameters:
  temperature:
    #method: btk
    #params:
      #temperature_gradient: -0.6
      #temperature_gradient_sd: 0.25
      #nug: 0.5
      #range: 200000.0
      #sill: 25.0
      #zscale: 20.0
    method: idw
    params:
      max_distance: 3000.0
      max_members: 5
      distance_measure_factor: 1.0
      default_temp_gradient: -0.005 # degC/m, so -0.5 degC/100m
      gradient_by_equation: false
  precipitation:
    method: idw
    params:
      max_distance: 3000.0
      max_members: 5
      distance_measure_factor: 1
      scale_factor: 1.02
  radiation:
    method: idw
    params:
      max_distance: 3000.0
      max_members: 5
      distance_measure_factor: 1.0
  wind_speed:
    method: idw
    params:
      max_distance: 3000.0
      max_members: 5
      distance_measure_factor: 1.0
  relative_humidity:
    method: idw
    params:
      max_distance: 3000.0
      max_members: 5
      distance_measure_factor: 1.0
```

Region.yaml (PTGSK, PTHSK, PTSSK)


```yaml
---
repository:
  class:
!!python/name:shyft.repository.netcdf.cf_region_model_repository.CFRegi
onModelRepository
  params:
    data_file: netcdf/orchestration-testdata/cell_data.nc

domain:
  EPSG: 32633
  nx: 109
  ny: 80
  step_x: 1000
  step_y: 1000
  lower_left_x: 266000
  lower_left_y: 6960000

catchment_indices:
  - 1228
  - 1308
  - 1394
  - 1443
  - 1726
  - 1867
  - 1996
  - 2041
  - 2129
  - 2195
  - 2198
  - 2277
  - 2402
  - 2446
  - 2465
  - 2545
  - 2640
  - 2718
  - 3002
  - 3536
  - 3630
  - 1000010
  - 1000011
```

model.yaml (PTGSK)


```yaml
model_t: !!python/name:shyft.api.pt_gs_k.PTGSKModel  # model to
construct
model_parameters:
  ae:  # actual_evapotranspiration
    ae_scale_factor: 0.7
  gs:  # gamma_snow
    calculate_iso_pot_energy: false
    fast_albedo_decay_rate: 1.194
    glacier_albedo: 0.484
    initial_bare_ground_fraction: 0.04
    max_albedo: 0.897
    max_water: 0.106
    min_albedo: 0.652
    n_winter_days: 217
    slow_albedo_decay_rate: 8.429
    snow_cv: 0.203
    snow_cv_altitude_factor: 0.0
    snow_cv_forest_factor: 0.0
    tx: -0.932
    snowfall_reset_depth: 6.401
    surface_magnitude: 29.798000000000002
    wind_const: 5.031000000000001
    wind_scale: 0.583
    winter_end_day_of_year: 114
  kirchner:
    c1: -3.984
    c2: 0.08900000000000001
    c3: -0.05599999999999994
  p_corr:  # precipitation_correction
    scale_factor: 0.727
  pt:  # priestley_taylor
    albedo: 0.2
    alpha: 1.26
  routing:
    alpha: 0.9
    beta: 3.0
    velocity: 0.0
  gm:
    direct_response: 0.475
```

model.yaml (PTHSK)


```yaml
model_t: !!python/name:shyft.api.pt_hs_k.PTHSKModel  # priestley_taylor
HBV_Snow kirchner
model_parameters:
  ae:  # actual_evapotranspiration
    ae_scale_factor: 0.603133018
  hs:  # HBV_Snow
    cfr: 0.000550204
    cx: 0.281854159
    lw: 0.051751484
    ts: 0.436940844
    tx: -0.42668965600000003
  kirchner:
    c1: -3.606984865
    c2: 0.462349299
    c3: -0.030007472
  p_corr:  # precipitation_correction
    scale_factor: 0.7766866720000001
  pt:  # priestley_taylor
    albedo: 0.2
    alpha: 1.26
  routing:
    alpha: 0.9
    beta: 3.0
    velocity: 0.0
```

model.yaml (PTSSK)


```
model_t: !!python/name:shyft.api.pt_ss_k.PTSSKModel  # priestley_taylor
Skaugen_Snow kirchner
model_parameters:
  ae:  # actual_evapotranspiration
    ae_scale_factor: 1.5
  ss:  # Skaugen_Snow
    alpha_0: 40.55
    cfr: 0.0098
    cx: 0.5857
    d_range: 110.71
    max_water_fraction: 0.3453
    ts: 0.137
    tx: -0.0042
    unit_size: 0.1858
  kirchner:
    c1: -3.916197322290274
    c2: 0.52433661533385695
    c3: -0.019503959620315988
  p_corr:  # precipitation_correction
    scale_factor: 1.5
  pt:  # priestley_taylor
    albedo: 0.2
    alpha: 1.26
  routing:
    alpha: 0.9
    beta: 3.0
    velocity: 0.0
```

simulation.yaml (PTGSK)


```
---
neanidelva:
  region_config_file: neanidelva_region.yaml
  model_config_file: neanidelva_model.yaml
  datasets_config_file: neanidelva_datasets.yaml
  interpolation_config_file: neanidelva_interpolation.yaml
  start_datetime: 2012-09-01T00:00:00
  run_time_step: 86400  # set to 3600 1 hour time step(slower
simulations, but hourly details)
  number_of_steps: 1095    # set to 8759 for hours in 1 year
  region_model_id: 'neanidelva-ptgsk'
  #interpolation_id: 2   # this is optional (default 0)
  initial_state:
    repository:
      class:
!!python/name:shyft.repository.generated_state_repository.GeneratedStat
eRepository
      params:
        model: !!python/name:shyft.api.pt_gs_k.PTGSKModel
    tags: []
  references:
  - repository:
!!python/name:shyft.repository.netcdf.cf_ts_repository.CFTsRepository
    params:
      file: netcdf/orchestration-testdata/discharge.nc
      var_type: discharge
    1D_timeseries:
    - catch_id: [1308,1394,1867,2198,2402,2545]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Tya...........-
D9100A3B1060R123.999
      run_time_step: 86400 # 3600
    - catch_id:
[1228,1443,1726,2041,2129,2195,2277,2465,2718,3002,3630,1000010,1000011
]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Selbu-lok.....-
D9100A3B1070R123.020
      run_time_step: 86400 # 3600
    - catch_id: [1996,2446,2640,3536]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Nea...........-
D9100A3B1050R123.998
      run_time_step: 86400 # 3600
...
```

simulation.yaml (PTHSK)


```
---
neanidelva:
  region_config_file: neanidelva_region.yaml
  model_config_file: neanidelva_model.yaml
  datasets_config_file: neanidelva_datasets.yaml
  interpolation_config_file: neanidelva_interpolation.yaml
  start_datetime: 2012-09-01T00:00:00
  run_time_step: 86400  # set to 3600 1 hour time step(slower
simulations, but hourly details)
  number_of_steps: 1095    # set to 8759 for hours in 1 year
  region_model_id: 'neanidelva-pthsk'
  #interpolation_id: 2   # this is optional (default 0)
  initial_state:
    repository:
      class:
!!python/name:shyft.repository.generated_state_repository.GeneratedStat
eRepository
      params:
        model: !!python/name:shyft.api.pt_hs_k.PTHSKModel
    tags: []
  references:
  - repository:
!!python/name:shyft.repository.netcdf.cf_ts_repository.CFTsRepository
    params:
      file: netcdf/orchestration-testdata/discharge.nc
      var_type: discharge
    1D_timeseries:
    - catch_id: [1308,1394,1867,2198,2402,2545]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Tya...........-
D9100A3B1060R123.999
      run_time_step: 86400 # 3600
    - catch_id:
[1228,1443,1726,2041,2129,2195,2277,2465,2718,3002,3630,1000010,1000011
]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Selbu-lok.....-
D9100A3B1070R123.020
      run_time_step: 86400 # 3600
    - catch_id: [1996,2446,2640,3536]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Nea...........-
D9100A3B1050R123.998
      run_time_step: 86400 # 3600
...
```

simulation.yaml (PTSSK)


```
---
neanidelva:
  region_config_file: neanidelva_region.yaml
  model_config_file: neanidelva_model.yaml
  datasets_config_file: neanidelva_datasets.yaml
  interpolation_config_file: neanidelva_interpolation.yaml
  start_datetime: 2012-09-01T00:00:00
  run_time_step: 86400  # set to 3600 1 hour time step(slower
simulations, but hourly details)
  number_of_steps: 1095    # set to 8759 for hours in 1 year
  region_model_id: 'neanidelva-ptssk'
  #interpolation_id: 2   # this is optional (default 0)
  initial_state:
    repository:
      class:
!!python/name:shyft.repository.generated_state_repository.GeneratedStat
eRepository
      params:
        model: !!python/name:shyft.api.pt_ss_k.PTSSKModel
    tags: []
  references:
  - repository:
!!python/name:shyft.repository.netcdf.cf_ts_repository.CFTsRepository
    params:
      file: netcdf/orchestration-testdata/discharge.nc
      var_type: discharge
    1D_timeseries:
    - catch_id: [1308,1394,1867,2198,2402,2545]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Tya...........-
D9100A3B1060R123.999
      run_time_step: 86400 # 3600
    - catch_id:
[1228,1443,1726,2041,2129,2195,2277,2465,2718,3002,3630,1000010,1000011
]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Selbu-lok.....-
D9100A3B1070R123.020
      run_time_step: 86400 # 3600
    - catch_id: [1996,2446,2640,3536]
      type: discharge
      uid: smg://SMG_PROD?name=/TEV.-Nea...........-
D9100A3B1050R123.998
      run_time_step: 86400 # 3600
...
```

Calibration.yaml (PTGSK)


```
neanidelva:
  model_config_file: neanidelva_simulation.yaml
  calibrated_model_file: calibrated_model.yaml  # file where the
calibrated params will go
  optimization_method:
    name: min_bobyqa # can be 'min_bobyqa', 'dream' or 'sceua'
    params:
      max_n_evaluations: 1504 #1504/ 1543/1562/1571/1581 FOR CONSTANT
TR use 1404/1443/1462/1471/1481 - or 1504/1523/1542/1561/1571
      tr_start: 0.1
      tr_stop: 0.00001
    #name: sceua
    #params:
      #max_n_evaluations: 2500
      #x_eps: 0.15
      #y_eps: 0.1
    #name: dream
    #params:
      #max_n_evaluations: 1500
  target:
  - repository:
!!python/name:shyft.repository.netcdf.cf_ts_repository.CFTsRepository
    params:
      file: netcdf/orchestration-testdata/discharge.nc
      var_type: discharge
    1D_timeseries:
    - catch_id: [1308,1394,1867,2198,2402,2545]
      uid: smg://SMG_PROD?name=/TEV.-Tya...........-
D9100A3B1060R123.999
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
          s_var: 1.0
          s_bias: 1.0
    - catch_id:
[1228,1443,1726,2041,2129,2195,2277,2465,2718,3002,3630,1000010,1000011
]
      uid: smg://SMG_PROD?name=/TEV.-Selbu-lok.....-
D9100A3B1070R123.020
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
```

```
          s_var: 1.0
          s_bias: 1.0
    - catch_id: [1996,2446,2640,3536]
      uid: smg://SMG_PROD?name=/TEV.-Nea...........-
D9100A3B1050R123.998
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
          s_var: 1.0
          s_bias: 1.0
  overrides:
    model:
      model_t: !!python/name:shyft.api.pt_gs_k.PTGSKOptModel
  calibration_parameters:
    kirchner.c1:
      min: -8.0
      max: 0.0
    kirchner.c2:
      min: -1.0
      max: 1.2
    kirchner.c3:
      min: -0.15
      max: -0.04
    ae.ae_scale_factor:
      min: 0.5
      max: 2.5
    gs.tx:
      min: -3.0
      max: 2.0
    gs.wind_scale:
      min: 0.5
      max: 6.0
    gs.max_water:
      min: 0.06
      max: 0.19
    gs.wind_const:
      min: 1.0
      max: 6.0
    gs.fast_albedo_decay_rate:
      min: 1.0
      max: 15.0
    gs.slow_albedo_decay_rate:
      min: 2.0
      max: 40.0
    gs.surface_magnitude:
      min: 10.0
      max: 70.0
    gs.max_albedo:
      min: 0.7
      max: 0.95
    gs.min_albedo:
      min: 0.4
```

```yaml
    max: 0.6999
  gs.snowfall_reset_depth:
    min: 4.0
    max: 9.0
  gs.snow_cv:
    min: 0.1
    max: 0.8
  gs.snow_cv_forest_factor:
    min: 0.0
    max: 0.0
  gs.snow_cv_altitude_factor:
    min: 0.0
    max: 0.0
  gs.glacier_albedo:
    min: 0.4
    max: 0.4
  p_corr.scale_factor:
    min: 0.5
    max: 2.0
  pt.albedo:
    min: 0.2
    max: 0.2
  pt.alpha:
    min: 1.26
    max: 1.26
  gs.initial_bare_ground_fraction:
    min: 0.04
    max: 0.04
  gs.winter_end_day_of_year:
    min: 80
    max: 125
  gs.calculate_iso_pot_energy:
    min: 0
    max: 0
  gs.n_winter_days:
    min: 170
    max: 270
  gm.dtf:
    min: 6.0
    max: 6.0
  gm.direct_response:
    min: 0.475
    max: 0.475
  routing.velocity:
    min: 0.0
    max: 0.0
  routing.alpha:
    min: 0.9
    max: 0.9
  routing.beta:
    min: 3.0
    max: 3.0
```

Calibration.yaml (PTHSK)


```
neanidelva:
  model_config_file: neanidelva_simulation.yaml
  calibrated_model_file: calibrated_model.yaml  # file where the
calibrated params will go
  optimization_method:
    name: min_bobyqa # can be 'min_bobyqa', 'dream' or 'sceua'
    params:
      max_n_evaluations: 1504 #1504/ 1543/1562/1571/1581 FOR CONSTANT
TR use 1404/1443/1462/1471/1481 - or 1504/1523/1542/1561/1571
      tr_start: 0.1
      tr_stop: 0.00001
    #name: sceua
    #params:
      #max_n_evaluations: 2500
      #x_eps: 0.15
      #y_eps: 0.1
    #name: dream
    #params:
      #max_n_evaluations: 1500
  target:
  - repository:
!!python/name:shyft.repository.netcdf.cf_ts_repository.CFTsRepository
    params:
      file: netcdf/orchestration-testdata/discharge.nc
      var_type: discharge
    1D_timeseries:
    - catch_id: [1308,1394,1867,2198,2402,2545]
      uid: smg://SMG_PROD?name=/TEV.-Tya...........-
D9100A3B1060R123.999
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
          s_var: 1.0
          s_bias: 1.0
    - catch_id:
[1228,1443,1726,2041,2129,2195,2277,2465,2718,3002,3630,1000010,1000011
]
      uid: smg://SMG_PROD?name=/TEV.-Selbu-lok.....-
D9100A3B1070R123.020
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
```

```
          s_var: 1.0
          s_bias: 1.0
    - catch_id: [1996,2446,2640,3536]
      uid: smg://SMG_PROD?name=/TEV.-Nea...........-
D9100A3B1050R123.998
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
          s_var: 1.0
          s_bias: 1.0
  overrides:
    model:
      model_t: !!python/name:shyft.api.pt_hs_k.PTHSKOptModel
  calibration_parameters:
    kirchner.c1:
      min: -8.0
      max: 0.0
    kirchner.c2:
      min: -1.0
      max: 1.2
    kirchner.c3:
      min: -0.15
      max: -0.05
    ae.ae_scale_factor:
      min: 1.5
      max: 1.5
    hs.cfr:
      min: 0
      max: 1
    hs.cx:
      min: 0
      max: 1
    hs.lw:
      min: 0
      max: 0.5
    hs.ts:
      min: -0.5
      max: 0.5
    hs.tx:
      min: -0.5
      max: 0.5
    p_corr.scale_factor:
      min: 1.0
      max: 1.0
    pt.albedo:
      min: 0.2
      max: 0.2
    pt.alpha:
      min: 1.26
      max: 1.26
    gm.dtf:
      min: 6.0
```

```
  max: 6.0
gm.direct_response:
  min: 0.475
  max: 0.475
routing.velocity:
  min: 0.0
  max: 0.0
routing.alpha:
  min: 0.9
  max: 0.9
routing.beta:
  min: 3.0
  max: 3.0
```

Calibration.yaml (PTSSK)


```
neanidelva:
  model_config_file: neanidelva_simulation.yaml
  calibrated_model_file: calibrated_model.yaml  # file where the
calibrated params will go
  optimization_method:
    name: min_bobyqa # can be 'min_bobyqa', 'dream' or 'sceua'
    params:
      max_n_evaluations: 1541 #1541/1542/1543/1544 FOR CONSTANT TR use
1441/1442/1443/1444
      tr_start: 0.1
      tr_stop: 0.00001
    #name: sceua
    #params:
      #max_n_evaluations: 2500
      #x_eps: 0.15
      #y_eps: 0.1
    #name: dream
    #params:
      #max_n_evaluations: 1500
  target:
  - repository:
!!python/name:shyft.repository.netcdf.cf_ts_repository.CFTsRepository
    params:
      file: netcdf/orchestration-testdata/discharge.nc
      var_type: discharge
    1D_timeseries:
    - catch_id: [1308,1394,1867,2198,2402,2545]
      uid: smg://SMG_PROD?name=/TEV.-Tya...........-
D9100A3B1060R123.999
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
          s_var: 1.0
          s_bias: 1.0
    - catch_id:
[1228,1443,1726,2041,2129,2195,2277,2465,2718,3002,3630,1000010,1000011
]
      uid: smg://SMG_PROD?name=/TEV.-Selbu-lok.....-
D9100A3B1070R123.020
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
```

```
          s_var: 1.0
          s_bias: 1.0
    - catch_id: [1996,2446,2640,3536]
      uid: smg://SMG_PROD?name=/TEV.-Nea...........-
D9100A3B1050R123.998
      start_datetime: 2012-09-01T00:00:00
      run_time_step: 86400 # 3600
      number_of_steps: 1095
      weight: 1.0
      obj_func:
        name: NSE # Nash-Sutcliffe efficiency (NSE) or Kling-Gupta
efficiency (KGE)
        scaling_factors:
          s_corr: 1.0
          s_var: 1.0
          s_bias: 1.0
  overrides:
    model:
      model_t: !!python/name:shyft.api.pt_ss_k.PTSSKOptModel
  calibration_parameters:
    kirchner.c1:
      min: -8.0
      max: 0.0
    kirchner.c2:
      min: -1.0
      max: 1.2
    kirchner.c3:
      min: -0.15
      max: -0.03
    ae.ae_scale_factor:
      min: 0.55
      max: 2.5
    ss.alpha_0:
      min: 10
      max: 70
    ss.cfr:
      min: 0
      max: 1.4
    ss.cx:
      min: 0
      max: 7
    ss.d_range:
      min: 20
      max: 300
    ss.max_water_fraction:
      min: 0.001
      max: 0.35
    ss.ts:
      min: -0.99
      max: 0.99
    ss.tx:
      min: -0.99
      max: 0.99
    ss.unit_size:
      min: 0.01
      max: 0.4
    p_corr.scale_factor:
      min: 0.5
```

```
    max: 1.8
  pt.albedo:
    min: 0.2
    max: 0.2
  pt.alpha:
    min: 1.26
    max: 1.26
  gm.dtf:
    min: 6.0
    max: 6.0
  gm.direct_response:
    min: 0.475
    max: 0.475
  routing.velocity:
    min: 0.0
    max: 0.0
  routing.alpha:
    min: 0.9
    max: 0.9
  routing.beta:
    min: 3.0
    max: 3.0
```

Appendix 4

Calibration codes

Calibration

Conventional SHyFT calibration is a one-time calibration. The program stops after each calibration and saves calibrated parameters to a calibrated.yaml file. It then reruns the calibration codes to get another calibrated parameter set.

Validation of parameters in SHyFT is done by placing new parameter set into a model.yaml file manually and running the simulation codes. In this study, 200 calibrations for each method were done. A number of codes written in python as well as loop calibration scripts were developed. The codes described below are capable of running calibrations multiple times and saving the calibrated parameters in a CSV file. They can further update the saved file after each new loop is complete. The only limitation is the machine memory. For a typical today`s computer with a core i7 – 8Mb Ram, the memory will usually overload after 17 loops which crashes the program necessitating a restart.

in addition, the codes can generate the simulated and observed discharge graphs for every calibration time and save them. So, it is possible to have many calibrated parameters in a single CSV file and their graphs after a period of time. The CSV parameter file is used in the validation part without the need to modify the model.yaml manually. This calibration code can be used for all types of methods in SHyFT.

Loop calibration

```
# Importing the third-party python modules

import os
from os import path
import sys
import datetime as dt
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import time
import random

all_results, good_results = [], []
counter = -1

while counter < 0:
    shyft_data_path = path.abspath(r"C:\shyft_workspace\shyft-data")
```

```
    if path.exists(shyft_data_path) and 'SHYFT_DATA' not in
os.environ:
        os.environ['SHYFT_DATA']=shyft_data_path
    from shyft.repository.default_state_repository import
DefaultStateRepository
    from shyft.orchestration.configuration.yaml_configs import
YAMLCalibConfig, YAMLSimConfig
    from shyft.orchestration.simulators.config_simulator import
ConfigCalibrator, ConfigSimulator

    counter += 1
    t1 = time.time()

    config_file_path =
os.path.abspath(r"D:\Dropbox\Thesis\SHyFT\Yaml_files\Skaugen\neanidelv
a_simulation.yaml")
    cfg = YAMLSimConfig(config_file_path, "neanidelva")
    simulator = ConfigSimulator(cfg)
    simulator.run()
    state = simulator.region_model.state
    region_model = simulator.region_model

    config_file_path =
os.path.abspath(r"D:\Dropbox\Thesis\SHyFT\Yaml_files\Skaugen\neanidelv
a_simulation.yaml")
    cfg = YAMLCalibConfig(config_file_path, "neanidelva")

    calib = ConfigCalibrator(cfg)
    cfg.optimization_method['params']['tr_start'] =
random.randrange(1,2000)/10000

    state_repos = DefaultStateRepository(calib.region_model)
    results = calib.calibrate(cfg.sim_config.time_axis,
state_repos.get_state(0).state_vector,
                              cfg.optimization_method['name'],
cfg.optimization_method['params'])
    t2 = time.time()
    now = str(dt.datetime.now())
    result_params = []
    for i in range(results.size()):
        result_params.append(results.get(i))

    result_params.append(1-
calib.optimizer.calculate_goal_function(result_params))

    result_params.append(int((t2-t1)/60))
    result_params.append(now)
    result_params.append(str(cfg.optimization_method['name']))
    result_params.append(str(cfg.optimization_method['params']))
    result_params.append(str(region_model.time_axis)[-30:-20])
    result_params.append(str(str(region_model.time_axis).split(',')[-
1][:-1]))
    result_params.append(str(cfg.overrides['model']['model_t'])[-15:-
10])
    result_params.append(str(cfg.calibration_parameters))

    all_results.append(result_params)
```

```
    pd_results = pd.DataFrame(all_results)
    pd_good_results = pd.DataFrame(good_results)

    pd_results_2 = pd_results.transpose()
    pd_good_results_2 = pd_good_results.transpose()

    if str(cfg.overrides['model']['model_t'])[-15:-10] == 'PTGSK':
        param_list =
['kirchner.c1','kirchner.c2','kirchner.c3','ae.ae_scale_factor','gs.tx
','gs.wind_scale','gs.max_water','gs.wind_const','gs.fast_albedo_decay
_rate','gs.slow_albedo_decay_rate','gs.surface_magnitude','gs.max_albe
do','gs.min_albedo','gs.snowfall_reset_depth','gs.snow_cv','gs.glacier
_albedo','p_corr.scale_factor','gs.snow_cv_forest_factor','gs.snow_cv_
altitude_factor','pt.albedo','pt.alpha','gs.initial_bare_ground_fracti
on','gs.winter_end_day_of_year','gs.calculate_iso_pot_energy','gm.dtf'
,'routing.velocity','routing.alpha','routing.beta','gs.n_winter_days',
'gm.direct_response','NSE','Computation time(Minutes)','Date &
Time','Method name','Params method','Start datetime','Number of
days','Model name','Ranges']

    elif str(cfg.overrides['model']['model_t'])[-15:-10] == 'PTHSK':
        param_list =
['kirchner.c1','kirchner.c2','kirchner.c3','ae.ae_scale_factor','hs.lw
','hs.tx','hs.cx','hs.ts','hs.cfr','gm.dtf','p_corr.scale_factor','pt.
albedo','pt.alpha','routing.velocity','routing.alpha','routing.beta','
gm.direct_response','NSE','Computation time(Minutes)','Date &
Time','Method name','Params method','Start datetime','Number of
days','Model name','Ranges']

    elif str(cfg.overrides['model']['model_t'])[-15:-10] == 'PTSSK':
        param_list =
['kirchner.c1','kirchner.c2','kirchner.c3','ae.ae_scale_factor','ss.al
pha_0','ss.d_range','ss.unit_size','ss.max_water_fraction','ss.tx','ss
.cx','ss.ts','ss.cfr','p_corr.scale_factor','pt.albedo','pt.alpha','gm
.dtf','routing.velocity','routing.alpha','routing.beta','gm.direct_res
ponse','NSE','Computation time(Minutes)','Date & Time','Method
name','Params method','Start datetime','Number of days','Model
name','Ranges']

    pd_param = pd.DataFrame(param_list)
    pd_res = pd.concat([pd_param, pd_results_2], axis = 1)

    pd_res2 = pd.concat([pd_param, pd_good_results_2], axis = 1)

    pd_res.to_csv('D:\\Dropbox\\Thesis\\SHyFT\\Results\\results.csv')

    target_obs = calib.tv[0]
    disch_sim_all = np.linspace(0,0,target_obs.ts.time_axis.size())
    disch_obs_all = np.linspace(0,0,target_obs.ts.time_axis.size())

    for tar in range(calib.tv.size()):

        target_obs = calib.tv[tar]
        disch_sim =
calib.region_model.statistics.discharge(target_obs.catchment_indexes)
        disch_obs = target_obs.ts.values
        disch_sim_np = np.array(disch_sim.values)
        disch_obs_np = np.array(disch_obs)
```

```
        disch_sim_all += disch_sim_np
        disch_obs_all += disch_obs_np

    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
target_obs.ts.time_axis]

    fig, ax = plt.subplots(1, figsize=(45,10))
    ax.plot(ts_timestamps, disch_sim_all, lw=1, ls = '-', label =
"sim", color = 'navy')
    ax.plot(ts_timestamps, disch_obs_all, lw=1, ls='-', label = "obs",
color = 'crimson')
    ax.set_title(f"observed and simulated discharge (sum of all
catchments) {str(simulator.region_model.__class__)[-12:-7]}")
    ax.legend()
    ax.set_ylabel("discharge [m3 s-1]")

    plt.savefig(f"D:\\Dropbox\\Thesis\\SHyFT\\Results\\{counter}.png")
```

Appendix 5

Simulation codes

1. One-time Simulation code

This code does one-time simulation and gives out data about the model which are include the following

- ✓ Double check the main model data
- ✓ Make Precipitation and Temperature graphs
- ✓ Make a Data-Frame and put discharge of all sub-catchments into a CSV file
- ✓ Make a Data-Frame and put the distributed p, T, Geo and etc. into separated CSV file
- ✓ Generate SCA, SWE and outflow of all cells
- ✓ Generate all SCA & SWE images
- ✓ Discharge graphs of all targets
- ✓ make precipitation graph

2. Loop simulation code

This code does a loop simulation and reads data from a CSV file column by column generating discharge validation graphs.

---

1. One-time Simulation code

```
#  Importing the third-party python modules

from netCDF4 import Dataset
import os
from os import path
import sys
import datetime as dt
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import time
#  Recored the starting time

t1 = time.time()


#  Define SHyFT data path
```

```
#  Adding 'r' to avoid change slash or doubl backslash

shyft_data_path = path.abspath(r"C:\shyft_workspace\shyft-data")
if path.exists(shyft_data_path) and 'SHYFT_DATA' not in os.environ:
    os.environ['SHYFT_DATA']=shyft_data_path


#  Importing the shyft modules

import shyft
from shyft import api
from shyft.repository.default_state_repository import
DefaultStateRepository
from shyft.orchestration.configuration.yaml_configs import
YAMLSimConfig
from shyft.orchestration.simulators.config_simulator import
ConfigSimulator


#  Set up YAML files to configure simulation

config_file_path =
r'D:\Dropbox\Thesis\SHyFT\neanidelva_simulation.yaml'
cfg = YAMLSimConfig(config_file_path, "neanidelva")


#  Config the simulator

simulator = ConfigSimulator(cfg)
region_model = simulator.region_model


#  Double check the main information of the model

print('Number of steps is','\t\t\t', cfg.number_of_steps,'\n')
print('Start datetime is','\t\t\t', cfg.start_datetime,'\n')
print('Number of seconds of each step is','\t',
cfg.run_time_step,'\n')
print('Name and method of model is','\t\t', cfg.region_model_id,'\n')
print('Number of total cells are','\t\t',
simulator.region_model.size(),'\n')
print('catchment_ids
are:\n\n',simulator.region_model.catchment_ids,'\n')


#  Run the simulation

simulator.run()


#  Make Precipitation and Temperature graph for a catchment or a cell
in a period

while True:
    question = input("make a P & T graph, for a catchment or a cell?")
    if question == 'catchment' or question == 'cell' or question ==
'stop':
        break
```

```
if question == 'catchment':

    print (region_model.catchment_ids)
    cid = int(input("Please enter catchment ID"))
    start_day = int(input("start day (0 to {})
?".format((cfg.number_of_steps-1))))
    left_days = cfg.number_of_steps - start_day
    n_day = int(input("how many days (0 to {}) ?".format(left_days)))

    ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(start_day),api.Cale
ndar.DAY,n_day)
    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

    pre_cell = region_model.statistics.precipitation([cid]).values
    temp_cell = region_model.statistics.temperature([cid]).values

    fig, ax1 = plt.subplots(figsize=(10,8))
    ax2 = ax1.twinx()
    ax1.plot(ts_timestamps,pre_cell[start_day:n_day+start_day],
c='black', lw=2, label='precipitation')
    ax2.plot(ts_timestamps, temp_cell[start_day:n_day+start_day],
c='purple', lw=2, label='Temperature')
    ax1.set_ylabel('daily precip [mm/h]')
    ax2.set_ylabel('temp [$°$ C]')
    ax1.set_xlabel('date')
    # loc = 1(right-top) 2(left-top) 3(bottom-left) 4(bottom-right)
    ax1.legend(loc=2); ax2.legend(loc=1)
    plt.show()
    print("precipitation = ", pre_cell[start_day:n_day+start_day])
    print("Temperature = ", temp_cell[start_day:n_day+start_day])


elif question == 'cell':

    print (f"Total number of cells are
{simulator.region_model.size()}, enter from 0 to
{simulator.region_model.size()-1}")
    cell_num = int(input("Enter the cell id"))
    start_day = int(input("start day (0 to {})
?".format((cfg.number_of_steps-1))))
    left_days = cfg.number_of_steps - start_day
    n_day = int(input("how many days (0 to {}) ?".format(left_days)))

    ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(start_day),api.Cale
ndar.DAY,n_day)
    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

    pre_cell =
region_model.cells[cell_num].env_ts.precipitation.values
    temp_cell = region_model.cells[cell_num].env_ts.temperature.values


    fig, ax1 = plt.subplots(figsize=(10,8))
    ax2 = ax1.twinx()
```

```
    ax1.plot(ts_timestamps,pre_cell[start_day:n_day+start_day],
c='cornflowerblue', lw=2, label='precipitation')
    ax2.plot(ts_timestamps, temp_cell[start_day:n_day+start_day],
c='orange', lw=2, label='Temperature')
    ax1.set_ylabel('daily precip [mm/h]')
    ax2.set_ylabel('temp [$°$ C]')
    ax1.set_xlabel('date')
    # loc = 1(right-top) 2(left-top) 3(bottom-left) 4(bottom-right)
    ax1.legend(loc=2); ax2.legend(loc=1)
    plt.show()
    print("precipitation = ", pre_cell[start_day:n_day+start_day])
    print("Temperature = ", temp_cell[start_day:n_day+start_day])
else:
    pass


#  Make a pandas DataFrame and put discharge of all subcatchments and
save into a CSV file

discharge_subcatch_pd = pd.DataFrame()
for cid in region_model.catchment_ids:
    discharge_subcatch_pd[cid] =
region_model.statistics.discharge([int(cid)]).values

ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
discharge_subcatch_pd.index = ts_timestamps
discharge_subcatch_pd.to_csv('discharge_subcatch_pd.csv')


#  Access to the whole dataframe

discharge_subcatch_pd.loc[:][:]


#  Access to the specific catchment and time

discharge_subcatch_pd.loc['2014-03-18'][1996]


#  Access to discharge of all catchments in specific date

discharge_subcatch_pd.loc['2014-03-18'][:]


#  Access to discharge of specific catchment in whole period

discharge_subcatch_pd.loc[:][1996]


#  Make a pandas DataFrame and put the distributed
precipitation,radiation,relative_humidity,temperature,wind_speed and
discharge of all cells and save into CSV files

precipitation_pd = pd.DataFrame()
radiation_pd = pd.DataFrame()
rel_hum_pd = pd.DataFrame()
temperature_pd = pd.DataFrame()
```

```
wind_speed_pd = pd.DataFrame()
disch_cell_pd = pd.DataFrame()

for num in range(region_model.size()):
    precipitation_pd[num] =
region_model.cells[num].env_ts.precipitation.values
    radiation_pd[num] =
region_model.cells[num].env_ts.radiation.values
    rel_hum_pd[num] = region_model.cells[num].env_ts.rel_hum.values
    temperature_pd[num] =
region_model.cells[num].env_ts.temperature.values
    wind_speed_pd[num] =
region_model.cells[num].env_ts.wind_speed.values
    disch_cell_pd[num] =
region_model.cells[num].rc.avg_discharge.values
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]

precipitation_pd.index = ts_timestamps
radiation_pd.index = ts_timestamps
rel_hum_pd.index = ts_timestamps
temperature_pd.index = ts_timestamps
wind_speed_pd.index = ts_timestamps
disch_cell_pd.index = ts_timestamps

precipitation_pd.to_csv('precipitation_pd.csv')
radiation_pd.to_csv('radiation_pd.csv')
rel_hum_pd.to_csv('rel_hum_pd.csv')
temperature_pd.to_csv('temperature_pd.csv')
wind_speed_pd.to_csv('wind_speed_pd.csv')
disch_cell_pd.to_csv('disch_cell_pd.csv')


#  Make discharge graph for a catchment or a cell in a period

while True:
    question = input("make a graph for a catchment or a cell?")
    if question == 'catchment' or question == 'cell' or question ==
'stop':
        break
if question == 'catchment':

    print (region_model.catchment_ids)
    cid = input("Please enter catchment ID")
    start_day = int(input("start day (0 to {})
?".format((cfg.number_of_steps-1))))
    left_days = cfg.number_of_steps - start_day
    n_day = int(input("how many days (0 to {}) ?".format(left_days)))

    fig, ax = plt.subplots(figsize=(10,8))
    ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(start_day),api.Cale
ndar.DAY,n_day)
    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]
    data = region_model.statistics.discharge([int(cid)]).values
    ax.plot(ts_timestamps,data[start_day:n_day+start_day], label =
"{}".format(cid))
```

```
    fig.autofmt_xdate()
    ax.legend(title="Catch. ID")
    ax.set_ylabel("discharge [m3 s-1]")
    plt.show()
    print(data[start_day:n_day+start_day])
elif question == 'cell':

    print (f"Total number of cells are
{simulator.region_model.size()}, enter from 0 to
{simulator.region_model.size()-1}")
    cell_num = int(input("Enter the cell id"))
    start_day = int(input("start day (0 to {})
?".format((cfg.number_of_steps-1))))
    left_days = cfg.number_of_steps - start_day
    n_day = int(input("how many days (0 to {}) ?".format(left_days)))

    fig, ax = plt.subplots(figsize=(10,8))
    ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(start_day),api.Cale
ndar.DAY,n_day)
    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

    data = region_model.cells[cell_num].rc.avg_discharge.values
    ax.plot(ts_timestamps,data[start_day:n_day+start_day], label =
f"Cell {cell_num}")

    fig.autofmt_xdate()
    ax.legend(title="Cell ID")
    ax.set_ylabel("discharge [m3 s-1]")
    plt.show()
    print(data[start_day:n_day+start_day])
else:
    pass


#  Access to the x, y ,z, area and catch_ids of all cells

cells = region_model.get_cells()

x = np.array([cell.geo.mid_point().x for cell in cells])
y = np.array([cell.geo.mid_point().y for cell in cells])
z = np.array([cell.geo.mid_point().z for cell in cells])
area = np.array([cell.geo.area() for cell in cells])
catch_ids = np.array([cell.geo.catchment_id() for cell in cells])


#  Make a panadas DataFrame for Geo. data save into a CSV file

geo_pd = pd.DataFrame()
geo_pd['x'] = x
geo_pd['y'] = y
geo_pd['z'] = z
geo_pd['catch_ids'] = catch_ids
geo_pd['area'] = area
geo_pd.to_csv('geo_pd.csv')
```

```
#  Do some calculation on Geo. data

np_z = np.array(geo_pd[:]['z'])
print(np_z.size)
print(np_z.mean())
print(np_z.max())
print(np_z.min())
print(np_z.std())


#   Access directly to the catchment_ids

catchment_ids = region_model.catchment_ids


#   Make a dictionary an enumarate them form zero to twenty-six

cid_z_map = dict([(catchment_ids[i],i) for i in
range(len(catchment_ids))])
print(cid_z_map)


#   Then create an array the same length as our 'x' and 'y', which
holds the integer reflecting values with the cid_z_map dictionary for
each single cells

catch_ids = np.array([cid_z_map[cell.geo.catchment_id()] for cell in
cells])


#   Illustrate the catchment

fig, ax = plt.subplots(figsize=(15,5))
cm = plt.cm.get_cmap(color[73])# color[0 to 75]
plot = ax.scatter(x, y, c=catch_ids, marker='s', s=7, lw=4, cmap=cm)
# plot = ax.scatter(x, y, c=z, marker='o', s=9, lw=5, cmap=cm)
plt.colorbar(plot).set_label('Numerate the sub-catchment IDs')
# plt.legend(title="sub-catchments", fontsize = 16, loc = 1)
plt.show()


#   Gamma-snow response
#   Set a date: year, month, day, (hour of day if hourly time step).
The oslo calendar(incl dst) converts calendar coordinates Y,M,D.. to
its utc-time. 1400104800 (seconds passed from 1970,1,1,1,0,0). It
needs to get the index of the time_axis for the time

oslo = api.Calendar('Europe/Oslo') # Europe/Berlin
time_x = oslo.time(2016,3,1)


#   Index of time x on time-axis

try:
    idx = simulator.region_model.time_axis.index_of(time_x)
except:
    print("Date out of range, setting index to 0")
```

```
    idx = 0


#   Snow Cover Area
#   In the mentioned day idx = (2016,3,1) for all all catchments ([])

sca = simulator.region_model.gamma_snow_response.sca([],idx)
# sca = simulator.region_model.hbv_snow_state.sca([],idx)
# sca = simulator.region_model.skaugen_snow_state.sca([],idx)

#   Snow Water Equivalent (mm)
#   In the mentioned day idx = (2016,3,1) for all catchments ([])

swe = simulator.region_model.gamma_snow_response.swe([],idx)
# swe = simulator.region_model.hbv_snow_state.swe([],idx)
# swe = simulator.region_model.skaugen_snow_state.swe([],idx)


#   The average of swe in the selected catchment, one value (mm)

swev = simulator.region_model.gamma_snow_response.swe_value([],idx)
# swev = simulator.region_model.hbv_snow_state.swe_value([],idx)
# swev = simulator.region_model.skaugen_snow_state.swe_value([],idx)

swe_np = np.array(swe)
area_np = np.array(area)
sum_np = swe_np * area_np
swe_average = sum_np.sum()/area_np.sum()
print(swe_average)
print(swev)
print(round(swe_average,3) == round(swev,3))


#   Do some calculation with numpy help

print(swe_np.mean())
print(swe_np.std())
print(swe_np.sum())
print(swe_np.max())
print(swe_np.min())


# The number of cells with more 250 mm Snow Water equivalent

swe_np[swe_np > 250].size


#   Snow outflow

sout = simulator.region_model.gamma_snow_response.outflow([],idx)
# sout = simulator.region_model.hbv_snow_response.outflow([],idx)
# sout = simulator.region_model.skaugen_snow_response.outflow([],idx)

sout_np = np.array(sout)
print(sout_np)
print(sout_np.sum())
print(sout_np.max())
print(sout_np.min())
```

```
#   Simple scatter plots for SCA , SWE , Outflow

fig, ax = plt.subplots(figsize=(15,5))
cm = plt.cm.get_cmap(color[2])# color[0 to 75]
plot = ax.scatter(x, y, c=sca, vmin=0, vmax=1,marker='s', s=40, lw=0,
cmap=cm)
plt.colorbar(plot)
plt.title('Snow Covered area of {0} on
{1}'.format(cfg.region_model_id, oslo.to_string(time_x)))

fig, ax = plt.subplots(figsize=(15,5))
cm = plt.cm.get_cmap(color[1]) # color[0 to 75]
plot = ax.scatter(x, y, c=swe, vmin=swe_np.min(), vmax=swe_np.max(),
marker='s', s=40, lw=0, cmap=cm)
plt.colorbar(plot)
plt.title('Snow Water Equivalent (mm) {0} on
{1}'.format(cfg.region_model_id, oslo.to_string(time_x)))

fig, ax = plt.subplots(figsize=(15,5))
cm = plt.cm.get_cmap(color[72])# color[0 to 75]
plot = ax.scatter(x, y, c=sout, vmin=sout_np.min(),
vmax=sout_np.max(), marker='s', s=40, lw=0, cmap=cm)
plt.colorbar(plot)
plt.title('Snow outflow {0} on {1}'.format(cfg.region_model_id,
oslo.to_string(time_x)))
plt.show()


#   Histogram of SCA and SWE

fig, ((ax1, ax2)) = plt.subplots(nrows=1, ncols=2, figsize = (15,6))
ax1.hist(sca, bins=20, range=(0,1), color='y', alpha=0.3)
ax1.set_xlabel("SCA of grid cell", fontsize=14)
ax1.set_ylabel("frequency", fontsize=14)

ax2.hist(swe, bins=20,  color='r', alpha=0.3)
ax2.set_xlabel("Snow Water Equivalent (mm) of grid cell", fontsize=14)
ax2.set_ylabel("frequency", fontsize=14)

plt.show()


#   Put SCA, SWE and outflow of all cells in  Pandas DataFrames and
save them into CSV files

SCA_pd = geo_pd.copy()
SWE_pd = geo_pd.copy()
outflow_pd = geo_pd.copy()
dic_swe_ptgsk = {}
dic_sca_ptgsk = {}

ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
for day in range(0,cfg.number_of_steps):
    sca = simulator.region_model.gamma_snow_response.sca([],day)
    SCA_pd[ts_timestamps[day]] = sca
```

```
    dic_sca_ptgsk.update({day:sca})
SCA_pd.to_csv('SCA_pd.csv')

ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
for day in range(0,cfg.number_of_steps):
    swe = simulator.region_model.gamma_snow_response.swe([],day)
    SWE_pd[ts_timestamps[day]] = swe
    dic_swe_ptgsk.update({day:swe})
SWE_pd.to_csv('SWE_pd.csv')

outflow_pd = pd.DataFrame()
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
for day in range(0,cfg.number_of_steps):
    outflow =
simulator.region_model.gamma_snow_response.outflow([],day)
    outflow_pd[ts_timestamps[day]] = outflow
outflow_pd.to_csv('outflow_pd.csv')

dic_swe_ptgsk_pd = pd.DataFrame(dic_swe_ptgsk)
dic_sca_ptgsk_pd = pd.DataFrame(dic_sca_ptgsk)

dic_swe_ptgsk_pd.to_csv('dic_swe_ptgsk_pd.csv')
dic_sca_ptgsk_pd.to_csv('dic_sca_ptgsk_pd.csv')

# SCA_pd = geo_pd.copy()
# SWE_pd = geo_pd.copy()
# outflow_pd = geo_pd.copy()
# dic_swe_pthsk = {}
# dic_sca_pthsk = {}

# ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
# for day in range(0,cfg.number_of_steps):
#     sca = simulator.region_model.hbv_snow_state.sca([],day)
#     SCA_pd[ts_timestamps[day]] = sca
#     dic_sca_pthsk.update({day:sca})
# SCA_pd.to_csv('SCA_pd.csv')

# ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
# for day in range(0,cfg.number_of_steps):
#     swe = simulator.region_model.hbv_snow_state.swe([],day)
#     SWE_pd[ts_timestamps[day]] = swe
#     dic_swe_pthsk.update({day:swe})
# SWE_pd.to_csv('SWE_pd.csv')

# outflow_pd = pd.DataFrame()
# ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
# for day in range(0,cfg.number_of_steps):
#     outflow =
simulator.region_model.hbv_snow_response.outflow([],day)
#     outflow_pd[ts_timestamps[day]] = outflow
# outflow_pd.to_csv('outflow_pd.csv')

# dic_swe_pthsk_pd = pd.DataFrame(dic_swe_pthsk)
```

```
# dic_sca_pthsk_pd = pd.DataFrame(dic_sca_pthsk)

# dic_swe_pthsk_pd.to_csv('dic_swe_pthsk_pd.csv')
# dic_sca_pthsk_pd.to_csv('dic_sca_pthsk_pd.csv')

# -----------------------------

# SCA_pd = geo_pd.copy()
# SWE_pd = geo_pd.copy()
# outflow_pd = geo_pd.copy()
# dic_swe_ptssk = {}
# dic_sca_ptssk = {}

# ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
# for day in range(0,cfg.number_of_steps):
#     sca = simulator.region_model.skaugen_snow_state.sca([],day)
#     SCA_pd[ts_timestamps[day]] = sca
#     dic_sca_ptssk.update({day:sca})
# SCA_pd.to_csv('SCA_pd.csv')

# ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
# for day in range(0,cfg.number_of_steps):
#     swe = simulator.region_model.skaugen_snow_state.swe([],day)
#     SWE_pd[ts_timestamps[day]] = swe
#     dic_swe_ptssk.update({day:swe})
# SWE_pd.to_csv('SWE_pd.csv')

# outflow_pd = pd.DataFrame()
# ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
region_model.time_axis]
# for day in range(0,cfg.number_of_steps):
# #     outflow =
simulator.region_model.gamma_snow_response.outflow([],day)
#     outflow =
simulator.region_model.skaugen_snow_response.outflow([],day)
#     outflow_pd[ts_timestamps[day]] = outflow
# outflow_pd.to_csv('outflow_pd.csv')

# dic_swe_ptssk_pd = pd.DataFrame(dic_swe_ptssk)
# dic_sca_ptssk_pd = pd.DataFrame(dic_sca_ptssk)

# dic_swe_ptssk_pd.to_csv('dic_swe_ptssk_pd.csv')
# dic_sca_ptssk_pd.to_csv('dic_sca_ptssk_pd.csv')


#  Make SWE graph for a catchment or a cell in a period

while True:
    question = input("make a SWE graph, for a catchment or a cell?")
    if question == 'catchment' or question == 'cell' or question ==
'stop':
        break
if question == 'catchment':

    print (region_model.catchment_ids)
    cid = input("Please enter catchment ID")
```

```
    start_day = int(input("start day (0 to {})
?".format((cfg.number_of_steps-1))))
    left_days = cfg.number_of_steps - start_day
    n_day = int(input("how many days (0 to {}) ?".format(left_days)))

    fig, ax = plt.subplots(figsize=(10,8))
    ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(start_day),api.Cale
ndar.DAY,n_day)
    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]
    swe_catch =
simulator.region_model.gamma_snow_response.swe([int(cid)]).v.to_numpy(
)
    ax.plot(ts_timestamps,swe_catch[start_day:n_day+start_day], label
= "{}".format(cid))

    fig.autofmt_xdate()
    ax.legend(title="Catch. ID")
    ax.set_ylabel("SWE (mm)")
    plt.show()
    print(swe_catch[start_day:n_day+start_day])

elif question == 'cell':

    print (f"Total number of cells are
{simulator.region_model.size()}, enter from 0 to
{simulator.region_model.size()-1}")
    cell_num = int(input("Enter the cell id"))
    start_day = int(input("start day (0 to {})
?".format((cfg.number_of_steps-1))))
    left_days = cfg.number_of_steps - start_day
    n_day = int(input("how many days (0 to {}) ?".format(left_days)))

    fig, ax = plt.subplots(figsize=(10,8))
    ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(start_day),api.Cale
ndar.DAY,n_day)
    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]
    swe_cell =
simulator.region_model.cells[cell_num].rc.snow_swe.v.to_numpy()
    ax.plot(ts_timestamps,swe_cell[start_day:n_day+start_day], label =
f"Cell {cell_num}")

    fig.autofmt_xdate()
    ax.legend(title="Cell ID")
    ax.set_ylabel("SWE (mm)")
    plt.show()
    print(swe_cell[start_day:n_day+start_day])

else:
    pass


#  Genarte all SCA images in whole period and save them in current
directory
```

```
q2 = input("Do you want to genarte all SCA images in whole period ? ")
if q2 == 'yes':
    for idx in range(365):
        tim_x = 1377986400+2*3600 + idx*86400
        sca = simulator.region_model.gamma_snow_response.sca([],idx)
        fig, ax = plt.subplots(figsize=(25,11))
        cm = plt.cm.get_cmap(color[2]) # color[0 to 75]
        plot = ax.scatter(x, y, c=sca, vmin=0, vmax=1, marker='s',
s=100, lw=0, cmap=cm)
        plt.colorbar(plot)
        plt.title('Snow Covered area of {0} on
{1}'.format(cfg.region_model_id,
dt.datetime.utcfromtimestamp(tim_x).date()),fontsize = 22)

        plt.savefig(f"SCA{idx}.png")
else:
    pass


#   Genarte all SWE images in whole period and save them in current
directory

max_swe = 0

q2 = input("Do you want to genarte all SWE images in whole period ? ")
if q2 == 'yes':

    for idx in range(365):
        tim_x = 1377986400+2*3600 + idx*86400
        swe = simulator.region_model.gamma_snow_response.swe([],idx)

        swe_np = np.array(swe)
        if swe_np.max() > max_swe:
            max_swe = swe_np.max()

    for idx in range(365):
        tim_x = 1377986400+2*3600 + idx*86400
        swe = simulator.region_model.gamma_snow_response.swe([],idx)


        fig, ax = plt.subplots(figsize=(25,11))
        cm = plt.cm.get_cmap(color[1]) # color[0 to 75]
        plot = ax.scatter(x, y, c=swe, vmin=0, vmax=max_swe,
marker='s', s=100, lw=0, cmap=cm)
        plt.colorbar(plot)
        plt.title('Snow Water Equivalent (mm) {0} on
{1}'.format(cfg.region_model_id,
dt.datetime.utcfromtimestamp(tim_x).date()),fontsize = 22)

        plt.savefig(f"SWE{idx}.png")

else:
    pass


#  Discharge graphs of all targets and sum of all targets for the
whole period for comparing the simulated ones and Ob. Ones with NSE
```

```
discharge_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-
testdata\discharge.nc'

discharge_data = Dataset(discharge_file)

dis_pd = pd.DataFrame(np.array(discharge_data['discharge'][:]))
startdatetime = (int(str(cfg.start_datetime -
datetime.datetime(2012,9,1)).split()[0])-1)

dis_target1 = dis_pd[0][:]
dis_target2 = dis_pd[1][:]
dis_target3 = dis_pd[2][:]
dis_targets = dis_pd[0][:] + dis_pd[1][:] + dis_pd[2][:]

dis_target1_np =
np.array(dis_target1[startdatetime:cfg.number_of_steps+startdatetime])
dis_target2_np =
np.array(dis_target2[startdatetime:cfg.number_of_steps+startdatetime])
dis_target3_np =
np.array(dis_target3[startdatetime:cfg.number_of_steps+startdatetime])
dis_targets_np =
np.array(dis_targets[startdatetime:cfg.number_of_steps+startdatetime])

target1 = [1308, 1394, 1867, 2198, 2402, 2545]
target2 = [1228, 1443, 1726, 2041, 2129, 2195, 2277, 2465, 2718, 3002,
3630, 1000010, 1000011]
target3 = [1996, 2446, 2640, 3536]

cid_z_map2 = {}
for key in cid_z_map.keys():
    if key in target1:
        cid_z_map2.update({key:1})
    elif key in target2:
        cid_z_map2.update({key:2})
    elif key in target3:
        cid_z_map2.update({key:3})
    else:
        cid_z_map2.update({key:0})

catch_ids2 = np.array([cid_z_map2[cell.geo.catchment_id()] for cell in
cells])

# ---------------------- Target1 ------------------------------------
--

dis_sim1 = region_model.statistics.discharge(target1).v.to_numpy() #
black

fig, ax = plt.subplots(figsize=(30,10))
ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(0),api.Calendar.DAY
,731)
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

ax.plot(ts_timestamps, dis_sim1, lw=1.5, ls ='-', color = 'black',
label = f'Sim from simulator {target1}')
```

```
ax.plot(ts_timestamps, dis_target1_np, lw=1.5, ls ='-', color = 'red',
label = 'Obs. Directly from discharge.nc')

NSE1 = 1-(((dis_sim1-dis_target1_np)**2).sum()/((dis_target1_np-
dis_target1_np.mean())**2).sum())

fig.autofmt_xdate()
ax.legend(title="Discharge", fontsize = 16, loc = 2)
ax.set_ylabel("discharge [m3 s-1]")
ax.set_title(f'Target 1, NSE1 = {round(NSE1,2)}', fontsize = 22)
plt.show()

# --------------------- Target2 -------------------------------------
--

dis_sim2 = region_model.statistics.discharge(target2).v.to_numpy() #
black

fig, ax = plt.subplots(figsize=(30,10))
ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(0),api.Calendar.DAY
,731)
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

ax.plot(ts_timestamps, dis_sim2, lw=1.5, ls ='-', color = 'black',
label = f'Sim from simulator {target2}')
ax.plot(ts_timestamps, dis_target2_np, lw=1.5, ls ='-', color = 'red',
label = 'Obs. Directly from discharge.nc')

NSE2 = 1-(((dis_sim2-dis_target2_np)**2).sum()/((dis_target2_np-
dis_target2_np.mean())**2).sum())

fig.autofmt_xdate()
ax.legend(title="Discharge", fontsize = 16, loc = 2)
ax.set_ylabel("discharge [m3 s-1]")
ax.set_title(f'Target 2, NSE2 = {round(NSE2,2)}', fontsize = 22)

plt.show()

# --------------------- Target3 -------------------------------------
--

dis_sim3 = region_model.statistics.discharge(target3).v.to_numpy() #
black

fig, ax = plt.subplots(figsize=(30,10))
ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(0),api.Calendar.DAY
,731)
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

ax.plot(ts_timestamps, dis_sim3, lw=1.5, ls ='-', color = 'black',
label = f'Sim from simulator {target3}')
ax.plot(ts_timestamps, dis_target3_np, lw=1.5, ls ='-', color = 'red',
label = 'Obs. Directly from discharge.nc')
```

```
NSE3 = 1-(((dis_sim3-dis_target3_np)**2).sum()/((dis_target3_np-
dis_target3_np.mean())**2).sum())

fig.autofmt_xdate()
ax.legend(title="Discharge", fontsize = 16, loc = 2)
ax.set_ylabel("discharge [m3 s-1]")
ax.set_title(f'Target 3, NSE3 = {round(NSE3,2)}', fontsize = 22)
plt.show()

# ---------------------- Targets -------------------------------------
--

dis_sims = dis_sim1 + dis_sim2 + dis_sim3

fig, ax = plt.subplots(figsize=(30,10))
ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(0),api.Calendar.DAY
,cfg.number_of_steps)
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

ax.plot(ts_timestamps, dis_sims, lw=1, ls ='-', color = 'black',
label = 'Sim. discharge')
ax.plot(ts_timestamps, dis_targets_np, lw=1, ls ='-', color = 'red',
label = 'Obs. discharge')

NSEs = 1-(((dis_sims-dis_targets_np)**2).sum()/((dis_targets_np-
dis_targets_np.mean())**2).sum())

fig.autofmt_xdate()
ax.legend(title="Discharge", fontsize = 16, loc = 2)
ax.set_ylabel("discharge [m3 s-1]")
ax.set_title(f'Targets, NSE = {round(NSEs,2)}', fontsize = 22)
plt.show()

# ---------------------- Catchments & Targets ------------------------
---

fig, ax = plt.subplots(figsize=(30,10))
cm = plt.cm.get_cmap(color[46])# color[0 to 75]
plot = ax.scatter(x, y, c=catch_ids2, marker='s',vmin = 0, vmax = 3,
s=30, lw=5, cmap=cm)
# plot = ax.scatter(x, y, c=catch_ids2, marker='s',vmin = 0, vmax = 3,
s=30, lw=5, cmap=cm)
plot = ax.scatter(x[140], y[140], marker='s',vmin = 0, vmax = 3, s=50,
lw=0, cmap=cm, label ="Target 1", color = 'slateblue')
plot = ax.scatter(x[140], y[140], marker='s',vmin = 0, vmax = 3, s=50,
lw=0, cmap=cm, label ="Target 2", color = 'deeppink')
plot = ax.scatter(x[140], y[140], marker='s',vmin = 0, vmax = 3, s=50,
lw=0, cmap=cm, label ="Target 3", color = 'maroon')

# plot = ax.scatter(x, y, c=z, marker='o', s=10, lw=5, cmap=cm)
# plt.colorbar(plot).set_label('sub-catchments assocciate to targets')
plt.legend( fontsize = 16, loc = 1)
plt.show()

timelist = []
for i in range(len(ts_timestamps)):
```

```
    timelist.append((str(ts_timestamps[i])[0:10],dis_sims[i]))
timelist_pd = pd.DataFrame(timelist)
timelist_pd.to_csv('dis_sims_G.csv')


#  Somthing more to know
#  Getting access to defualt values of variables (not used in 
simulation)

parameterg = api.GammaSnowParameter()
parameterk = api.KirchnerParameter()
print('slow_albedo_decay_rate = ', parameterg.slow_albedo_decay_rate)
print('Kirchner C1 = ', parameterk.c1)


#  Getting access to the values are used for simulation which are in 
model.yaml (used in simulation)

param = simulator.region_model.get_region_parameter()
print('slow_albedo_decay_rate = ',param.gs.slow_albedo_decay_rate)
print('Kirchner C1 = ', param.kirchner.c1)


#  Getting access to atributes of simulator

for attr in dir(simulator.region_model):
    if attr[0] is not '_': #ignore privates
        print(attr)


#  Precipitation graph

precipitation_r = simulator.region_model.statistics.precipitation([])
precipitation_r_np = precipitation_r.values.to_numpy()

fig, ax = plt.subplots(figsize=(30,10))
ta_statistics = 
api.TimeAxis(simulator.region_model.time_axis.time(0),api.Calendar.DAY
,cfg.number_of_steps)
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in 
ta_statistics]
ax.plot(ts_timestamps, precipitation_r_np, lw=1, ls ='-', color = 
'red',  label = 'precipitation')
ax.legend(fontsize = 16, loc = 2)

precipitation_r_np_pd = pd.DataFrame(precipitation_r_np)
precipitation_r_np_pd.to_csv('precipitation_r_np_pd_G.csv')


#  Average SWE

SWE_average = simulator.region_model.gamma_snow_response.swe([])
SWE_average_np = ss1.values.to_numpy()

fig, ax = plt.subplots(figsize=(30,10))
ta_statistics = 
api.TimeAxis(simulator.region_model.time_axis.time(0),api.Calendar.DAY
,cfg.number_of_steps)
```

```
ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]
ax.plot(ts_timestamps, SWE_average_np, lw=1, ls ='-', color = 'blue',
label = 'SWE_average')
ax.legend(fontsize = 16, loc = 2)

SWE_average_np_pd = pd.DataFrame(SWE_average_np)
SWE_average_np_pd.to_csv('SWE_average_np_pd_G.csv')


#  Calculation of the spent time

t2 = time.time()
t3 = t2-t1
hour1 = int(t3//3600)
minute1 = int((t3 % 3600)//60)
second1 = int(t3 - hour1*3600 - minute1*60)
print("",hour1,"Hours\n",minute1,"Minutes\n",second1,"Seconds")


#  Notify the end of simulation with an alarm

print('_It is done_'*7)
import winsound
for i in range(2500,3500,250):
    winsound.Beep(i, 850)
```

END

2. Loop simulation code

```
from netCDF4 import Dataset
import os
import time
from os import path
import sys
import datetime as dt
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
for column in range(1,20):
```

```
    my_data =
pd.read_csv(r"D:\Dropbox\Thesis\SHyFT\Gamma_parameters.csv")

    with open(r"D:\Dropbox\Thesis\SHyFT\neanidelva_model.yaml", 'w')
as parameters2:
        print(f"model_t: !!python/name:shyft.api.pt_gs_k.PTGSKModel  #
model to construct", file=parameters2)
        print(f"model_parameters:", file=parameters2)
        print(f"  ae:  # actual_evapotranspiration", file=parameters2)
        print(f"    ae_scale_factor: {my_data.iloc[3][c]}",
file=parameters2)
        print(f"  gs:  # gamma_snow", file=parameters2)
        print(f"    calculate_iso_pot_energy: false",
file=parameters2)
        print(f"    fast_albedo_decay_rate: {my_data.iloc[8][c]}",
file=parameters2)
        print(f"    glacier_albedo: {my_data.iloc[15][c]}",
file=parameters2)
        print(f"    initial_bare_ground_fraction:
{my_data.iloc[21][c]}", file=parameters2)
        print(f"    max_albedo: {my_data.iloc[11][c]}",
file=parameters2)
        print(f"    max_water: {my_data.iloc[6][c]}",
file=parameters2)
        print(f"    min_albedo: {my_data.iloc[12][c]}",
file=parameters2)
        print(f"    n_winter_days: {int(my_data.iloc[28][c])}",
file=parameters2)
        print(f"    slow_albedo_decay_rate: {my_data.iloc[9][c]}",
file=parameters2)
        print(f"    snow_cv: {my_data.iloc[14][c]}", file=parameters2)
        print(f"    snow_cv_altitude_factor: {my_data.iloc[18][c]}",
file=parameters2)
        print(f"    snow_cv_forest_factor: {my_data.iloc[17][c]}",
file=parameters2)
        print(f"    tx: {my_data.iloc[4][c]}", file=parameters2)
        print(f"    snowfall_reset_depth: {my_data.iloc[13][c]}",
file=parameters2)
        print(f"    surface_magnitude: {my_data.iloc[10][c]}",
file=parameters2)
        print(f"    wind_const: {my_data.iloc[7][c]}",
file=parameters2)
        print(f"    wind_scale: {my_data.iloc[5][c]}",
file=parameters2)
        print(f"    winter_end_day_of_year:
{int(my_data.iloc[22][c])}", file=parameters2)
        print(f"  kirchner:", file=parameters2)
        print(f"    c1: {my_data.iloc[0][c]}", file=parameters2)
        print(f"    c2: {my_data.iloc[1][c]}", file=parameters2)
        print(f"    c3: {my_data.iloc[2][c]}", file=parameters2)
        print(f"  p_corr:  # precipitation_correction",
file=parameters2)
        print(f"    scale_factor: {my_data.iloc[16][c]}",
file=parameters2)
        print(f"  pt:  # priestley_taylor", file=parameters2)
        print(f"    albedo: {my_data.iloc[19][c]}", file=parameters2)
        print(f"    alpha: {my_data.iloc[20][c]}", file=parameters2)
        print(f"  routing:", file=parameters2)
```

```
        print(f"    alpha: {my_data.iloc[26][c]}", file=parameters2)
        print(f"    beta: {my_data.iloc[27][c]}", file=parameters2)
        print(f"    velocity: {my_data.iloc[25][c]}",
file=parameters2)
        print(f"  gm:", file=parameters2)
        print(f"    direct_response: {my_data.iloc[29][c]}",
file=parameters2)

# for column in range(1,20):
#     my_data =
pd.read_csv(r"D:\Dropbox\Thesis\SHyFT\HBV_parameters.csv")

#     with open(r"D:\Dropbox\Thesis\SHyFT\neanidelva_model.yaml", 'w')
as parameters2:
#         print(f"model_t: !!python/name:shyft.api.pt_hs_k.PTHSKModel
# priestley_taylor HBV_Snow kirchner", file=parameters2)
#         print(f"model_parameters:", file=parameters2)
#         print(f"  ae:  # actual_evapotranspiration",
file=parameters2)
#         print(f"    ae_scale_factor: {my_data.iloc[3][column]}",
file=parameters2)
#         print(f"  hs:  # HBV_Snow", file=parameters2)
#         print(f"    cfr: {my_data.iloc[8][column]}",
file=parameters2)
#         print(f"    cx: {my_data.iloc[6][column]}",
file=parameters2)
#         print(f"    lw: {my_data.iloc[4][column]}",
file=parameters2)
#         print(f"    ts: {my_data.iloc[7][column]}",
file=parameters2)
#         print(f"    tx: {my_data.iloc[5][column]}  ",
file=parameters2)
#         print(f"  kirchner:", file=parameters2)
#         print(f"    c1: {my_data.iloc[0][column]}",
file=parameters2)
#         print(f"    c2: {my_data.iloc[1][column]}",
file=parameters2)
#         print(f"    c3: {my_data.iloc[2][column]}",
file=parameters2)
#         print(f"  p_corr:  # precipitation_correction",
file=parameters2)
#         print(f"    scale_factor: {my_data.iloc[10][column]}",
file=parameters2)
#         print(f"  pt:  # priestley_taylor", file=parameters2)
#         print(f"    albedo: {my_data.iloc[11][column]}",
file=parameters2)
#         print(f"    alpha: {my_data.iloc[12][column]}",
file=parameters2)
#         print(f"  routing:", file=parameters2)
#         print(f"    alpha: {my_data.iloc[14][column]}",
file=parameters2)
#         print(f"    beta: {my_data.iloc[15][column]}",
file=parameters2)
#         print(f"    velocity: {my_data.iloc[13][column]}",
file=parameters2)

# for column in range(1,20):
```

```
#     my_data =
pd.read_csv(r"D:\Dropbox\Thesis\SHyFT\Skaugen_parameters.csv")

#     with open(r"D:\Dropbox\Thesis\SHyFT\neanidelva_model.yaml", 'w')
as parameters2:
#         print(f"model_t: !!python/name:shyft.api.pt_ss_k.PTSSKModel
# priestley_taylor Skaugen_Snow kirchner", file=parameters2)
#         print(f"model_parameters:", file=parameters2)
#         print(f"  ae:  # actual_evapotranspiration",
file=parameters2)
#         print(f"    ae_scale_factor: {my_data.iloc[3][column]}",
file=parameters2)
#         print(f"  ss:  # Skaugen_Snow", file=parameters2)
#         print(f"    alpha_0: {my_data.iloc[4][column]}",
file=parameters2)
#         print(f"    cfr: {my_data.iloc[11][column]}",
file=parameters2)
#         print(f"    cx: {my_data.iloc[9][column]}",
file=parameters2)
#         print(f"    d_range: {my_data.iloc[5][column]}",
file=parameters2)
#         print(f"    max_water_fraction: {my_data.iloc[7][column]}",
file=parameters2)
#         print(f"    ts: {my_data.iloc[10][column]}",
file=parameters2)
#         print(f"    tx: {my_data.iloc[8][column]}",
file=parameters2)
#         print(f"    unit_size: {my_data.iloc[6][column]}",
file=parameters2)
#         print(f"  kirchner:", file=parameters2)
#         print(f"    c1: {my_data.iloc[0][column]}",
file=parameters2)
#         print(f"    c2: {my_data.iloc[1][column]}",
file=parameters2)
#         print(f"    c3: {my_data.iloc[2][column]}",
file=parameters2)
#         print(f"  p_corr:  # precipitation_correction",
file=parameters2)
#         print(f"    scale_factor: {my_data.iloc[12][column]}",
file=parameters2)
#         print(f"  pt:  # priestley_taylor", file=parameters2)
#         print(f"    albedo: {my_data.iloc[13][column]}",
file=parameters2)
#         print(f"    alpha: {my_data.iloc[14][column]}",
file=parameters2)
#         print(f"  routing:", file=parameters2)
#         print(f"    alpha: {my_data.iloc[17][column]}",
file=parameters2)
#         print(f"    beta: {my_data.iloc[18][column]}",
file=parameters2)
#         print(f"    velocity: {my_data.iloc[16][column]}",
file=parameters2)

    time.sleep(5)

    shyft_data_path = path.abspath(r"C:\shyft_workspace\shyft-data")
    if path.exists(shyft_data_path) and 'SHYFT_DATA' not in
os.environ:
```

```
        os.environ['SHYFT_DATA']=shyft_data_path

    import shyft
    from shyft import api
    from shyft.repository.default_state_repository import
DefaultStateRepository
    from shyft.orchestration.configuration.yaml_configs import
YAMLSimConfig
    from shyft.orchestration.simulators.config_simulator import
ConfigSimulator

    config_file_path =
r'D:\Dropbox\Thesis\SHyFT\neanidelva_simulation.yaml'
    cfg = YAMLSimConfig(config_file_path, "neanidelva")

    simulator = ConfigSimulator(cfg)
    region_model = simulator.region_model

    simulator.region_model.set_snow_sca_swe_collection(-1,True)
    simulator.region_model.set_state_collection(-1,True)
    simulator.run()

    cells = region_model.get_cells()

    x = np.array([cell.geo.mid_point().x for cell in cells])
    y = np.array([cell.geo.mid_point().y for cell in cells])
    z = np.array([cell.geo.mid_point().z for cell in cells])
    area = np.array([cell.geo.area() for cell in cells])
    catch_ids = np.array([cell.geo.catchment_id() for cell in cells])

    catchment_ids = region_model.catchment_ids

    cid_z_map = dict([(catchment_ids[i],i) for i in
range(len(catchment_ids))])
    print(cid_z_map)

    catch_ids = np.array([cid_z_map[cell.geo.catchment_id()] for cell
in cells])

    discharge_file = r'C:\shyft_workspace\shyft-
data\netcdf\orchestration-testdata\discharge.nc'

    discharge_data = Dataset(discharge_file)

    dis_pd = pd.DataFrame(np.array(discharge_data['discharge'][:]))
    startdatetime = (int(str(cfg.start_datetime -
datetime.datetime(2012,9,1)).split()[0])-1)

    dis_target1 = dis_pd[0][:]
    dis_target2 = dis_pd[1][:]
    dis_target3 = dis_pd[2][:]
    dis_targets = dis_pd[0][:] + dis_pd[1][:] + dis_pd[2][:]

    dis_target1_np =
np.array(dis_target1[startdatetime:cfg.number_of_steps+startdatetime])
    dis_target2_np =
np.array(dis_target2[startdatetime:cfg.number_of_steps+startdatetime])
```

```
    dis_target3_np =
np.array(dis_target3[startdatetime:cfg.number_of_steps+startdatetime])
    dis_targets_np =
np.array(dis_targets[startdatetime:cfg.number_of_steps+startdatetime])

    target1 = [1308, 1394, 1867, 2198, 2402, 2545]
    target2 = [1228, 1443, 1726, 2041, 2129, 2195, 2277, 2465, 2718,
3002, 3630, 1000010, 1000011]
    target3 = [1996, 2446, 2640, 3536]

    cid_z_map2 = {}
    for key in cid_z_map.keys():
        if key in target1:
            cid_z_map2.update({key:1})
        elif key in target2:
            cid_z_map2.update({key:2})
        elif key in target3:
            cid_z_map2.update({key:3})
        else:
            cid_z_map2.update({key:0})

    catch_ids2 = np.array([cid_z_map2[cell.geo.catchment_id()] for
cell in cells])

    dis_sim1 = region_model.statistics.discharge(target1).v.to_numpy()
    dis_sim2 = region_model.statistics.discharge(target2).v.to_numpy()
    dis_sim3 = region_model.statistics.discharge(target3).v.to_numpy()
    dis_sims = dis_sim1 + dis_sim2 + dis_sim3

    fig, ax = plt.subplots(figsize=(30,10))
    ta_statistics =
api.TimeAxis(simulator.region_model.time_axis.time(0),api.Calendar.DAY
,cfg.number_of_steps)
    ts_timestamps = [dt.datetime.utcfromtimestamp(p.start) for p in
ta_statistics]

    ax.plot(ts_timestamps, dis_sims, lw=1, ls ='-', color = 'black',
label = 'Sim from simulator for all targets')
    ax.plot(ts_timestamps, dis_targets_np, lw=1, ls ='-', color =
'red',  label = 'Obs. Directly from discharge.nc')

    NSEs = 1-(((dis_sims-dis_targets_np)**2).sum()/((dis_targets_np-
dis_targets_np.mean())**2).sum())

    fig.autofmt_xdate()
    ax.legend(title="Discharge", fontsize = 16, loc = 2)
    ax.set_ylabel("discharge [m3 s-1]")
    ax.set_title(f'Targets, NSE = {round(NSEs,2)}', fontsize = 22)

    file_name = str(column)
    plt.savefig(f"D:\\Dropbox\\Thesis\\{file_name}.png")

    plt.show()

print('_It is done_'*7)
import winsound
for i in range(2500,3500,250):
    winsound.Beep(i, 850)
```

Appendix 6

Miscellaneous codes

Miscellaneous codes

In this part some miscellaneous codes are presented. These codes were written in Python Scripts to make the work flow easier, more precise and presentable.

1. Python Scripts to generate SWE and SCA images to make a video file with generated database file in simulation part.
2. Python Scripts to read NC file

1. Python Scripts to generate SWE and SCA

```
from netCDF4 import Dataset
import os
from os import path
import sys
import datetime as dt
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
color = {0:'viridis', 1:'plasma', 2:'inferno', 3:'magma', 4:'Greys',
5:'Purples', 6:'Blues', 7:'Greens',  8:'Oranges', 9:'Reds',
10:'YlOrBr'}


while True:

    dic_sca_ptgsk_pd2=pd.read_csv('dic_sca_ptgsk_pd.csv')
    dic_sca_pthsk_pd2=pd.read_csv('dic_sca_pthsk_pd.csv')
    dic_sca_ptssk_pd2=pd.read_csv('dic_sca_ptssk_pd.csv')

    dic_swe_ptgsk_pd2=pd.read_csv('dic_swe_ptgsk_pd.csv')
    dic_swe_pthsk_pd2=pd.read_csv('dic_swe_pthsk_pd.csv')
    dic_swe_ptssk_pd2=pd.read_csv('dic_swe_ptssk_pd.csv')

    geo_data = pd.read_csv('geo_pd.csv')

    for idx in range(364): # one year
        tim_x = 1377986400+2*3600 + idx*86400 # 2013/9/1

        fig, ((ax1, ax2)) = plt.subplots(nrows=1, ncols=2, figsize =
(25,13))

        cm = plt.cm.get_cmap(color[3]) # color[0 to 75]
        ax1.scatter(geo_data['x'][:]-260000, geo_data['y'][:]-6830000,
c=dic_sca_ptgsk_pd2[str(idx)][:], vmin=0, vmax = 1, marker='s', s=100,
lw=0, cmap=cm)
        ax1.scatter(geo_data['x'][:]-260000, geo_data['y'][:]-60000-
6830000,  c=dic_sca_pthsk_pd2[str(idx)][:], vmin=0, vmax = 1,
marker='s', s=100, lw=0, cmap=cm)
```

```
        ax1.scatter(geo_data['x'][:]-260000, geo_data['y'][:]-120000-
6830000, c=dic_sca_ptssk_pd2[str(idx)][:], vmin=0, vmax = 1,
marker='s', s=100, lw=0, cmap=cm)

        ax1.annotate('Gamma Snow (0 - 0.96)', xy =(5500,211000),
fontsize = 16, color = "black")
        ax1.annotate('HBV Snow (0 - 1)', xy =(5500,152000), fontsize =
16, color = "black")
        ax1.annotate('Skaugen Snow (0 - 1)', xy =(5500,92000), fontsize
= 16, color = "black")

        ax1.set_xticks([])
        ax1.set_yticks([])

        ax1.set_title('Snow Cover Area on
{}'.format(dt.datetime.utcfromtimestamp(tim_x).date()),fontsize = 20)

        cm = plt.cm.get_cmap(color[1]) # color[0 to 75]
        ax2.scatter(geo_data['x'][:]-260000, geo_data['y'][:]-6830000,
c=dic_swe_ptgsk_pd2[str(idx)][:], vmin=0, vmax = 600, marker='s',
s=100, lw=0, cmap=cm)
        ax2.scatter(geo_data['x'][:]-260000, geo_data['y'][:]-60000-
6830000, c=dic_swe_pthsk_pd2[str(idx)][:], vmin=0, vmax = 25,
marker='s', s=100, lw=0, cmap=cm)
        ax2.scatter(geo_data['x'][:]-260000, geo_data['y'][:]-120000-
6830000, c=dic_swe_ptssk_pd2[str(idx)][:], vmin=0, vmax = 25,
marker='s', s=100, lw=0, cmap=cm)

        ax2.annotate('Gamma Snow', xy =(5500,211000), fontsize = 16,
color = "black")
        ax2.annotate('HBV Snow', xy =(5500,152000), fontsize = 16,
color = "black")
        ax2.annotate('Skaugen Snow', xy =(5500,92000), fontsize = 16,
color = "black")

        ax2.set_xticks([])
        ax2.set_yticks([])

        ax2.set_title('Snow Water Equivalent (mm) on
{}'.format(dt.datetime.utcfromtimestamp(tim_x).date()),fontsize = 20)

        plt.savefig(f"SCA_PTGSK_PTHSK_PTSSK{idx}.png")
```

2. *Python Scripts to read NC file*

```
from netCDF4 import Dataset
import pandas as pd
import numpy as np
import os


# 1. Precipitation

precipitation_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-
testdata\precipitation.nc'

precipitation_data = Dataset(precipitation_file)

series_pd = pd.DataFrame()
pre_pd = pd.DataFrame(np.array(precipitation_data['precipitation'][:]))

for item in ['x', 'y', 'z', 'series_name']:
    series_pd[repr(item)] = np.array(precipitation_data[item][:])
series_2_pd = series_pd.transpose()

frames = [series_2_pd, pre_pd]
precipitation = pd.concat(frames)

# set the current directory to the file directory
os.chdir(os.path.dirname(precipitation_file))
# get the file name without extension
file_name = precipitation_file.split('\\')[-1].split(".")[-2]

precipitation.to_csv(f'{file_name}.csv')


# 2. GeoCell

cells_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-testdata\cell_data.nc'
cell_data = Dataset(cells_file)

cells_pd = pd.DataFrame()

for key in cell_data.variables.keys():
    cells_pd[key] = np.array(cell_data[key][:])

# set the current directory to the file directory
os.chdir(os.path.dirname(cells_file))
```

```
# get the file name without extension
file_name = cells_file.split('\\')[-1].split(".")[-2]

cells_pd.to_csv(f'{file_name}.csv')


# 3. Disharge

discharge_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-testdata\discharge.nc'

discharge_data = Dataset(discharge_file)

series_pd = pd.DataFrame()
dis_pd = pd.DataFrame(np.array(discharge_data['discharge'][:]))

for item in ['x', 'y', 'z', 'series_name']:
    series_pd[repr(item)] = np.array(discharge_data[item][:])
series_2_pd = series_pd.transpose()

frames = [series_2_pd, dis_pd]
discharge = pd.concat(frames)

# set the current directory to the file directory
os.chdir(os.path.dirname(discharge_file))
# get the file name without extension
file_name = discharge_file.split('\\')[-1].split(".")[-2]

discharge.to_csv(f'{file_name}.csv')


# 4. Radiation

radiation_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-testdata\radiation.nc'

radiation_data = Dataset(radiation_file)

series_pd = pd.DataFrame()
radi_pd = pd.DataFrame(np.array(radiation_data['global_radiation'][:]))

for item in ['x', 'y', 'z', 'series_name']:
    series_pd[repr(item)] = np.array(radiation_data[item][:])
series_2_pd = series_pd.transpose()

frames = [series_2_pd, radi_pd]
radiation = pd.concat(frames)

# set the current directory to the file directory
```

```
os.chdir(os.path.dirname(radiation_file))
# get the file name without extension
file_name = radiation_file.split('\\')[-1].split(".")[-2]

radiation.to_csv(f'{file_name}.csv')


# 5. Relative_humidity

humidity_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-
testdata\relative_humidity.nc'

humidity_data = Dataset(humidity_file)

series_pd = pd.DataFrame()
humi_pd = pd.DataFrame(np.array(humidity_data['relative_humidity'][:]))

for item in ['x', 'y', 'z', 'series_name']:
    series_pd[repr(item)] = np.array(humidity_data[item][:])
series_2_pd = series_pd.transpose()

frames = [series_2_pd, radi_pd]
humidity = pd.concat(frames)

# set the current directory to the file directory
os.chdir(os.path.dirname(humidity_file))
# get the file name without extension
file_name = humidity_file.split('\\')[-1].split(".")[-2]

humidity.to_csv(f'{file_name}.csv')


# 6. Temperature

temperature_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-
testdata\temperature.nc'

temperature_data = Dataset(temperature_file)

series_pd = pd.DataFrame()
temp_pd = pd.DataFrame(np.array(temperature_data['temperature'][:]))

for item in ['x', 'y', 'z', 'series_name']:
    series_pd[repr(item)] = np.array(temperature_data[item][:])
series_2_pd = series_pd.transpose()

frames = [series_2_pd, temp_pd]
```

```
temperature = pd.concat(frames)

# set the current directory to the file directory
os.chdir(os.path.dirname(temperature_file))
# get the file name without extension
file_name = temperature_file.split('\\')[-1].split(".")[-2]

temperature.to_csv(f'{file_name}.csv')


# 7. wind_speed

wind_file = r'C:\shyft_workspace\shyft-data\netcdf\orchestration-testdata\wind_speed.nc'

wind_data = Dataset(wind_file)

series_pd = pd.DataFrame()
wind_pd = pd.DataFrame(np.array(wind_data['wind_speed'][:]))

for item in ['x', 'y', 'z', 'series_name']:
    series_pd[repr(item)] = np.array(wind_data[item][:])
series_2_pd = series_pd.transpose()

frames = [series_2_pd, wind_pd]
wind = pd.concat(frames)

# set the current directory to the file directory
os.chdir(os.path.dirname(wind_file))
# get the file name without extension
file_name = wind_file.split('\\')[-1].split(".")[-2]

wind.to_csv(f'{file_name}.csv')
```

Appendix 7

Calibration results

Table Ap7.1 200 calibration results for PTSSK method

PTSSK calibrations									
No.	NSE	No.	NSE	No.	NSE	No.	NSE	No.	NSE
1	77.2%	21	76.7%	41	76.6%	61	76.6%	81	76.5%
2	77.0%	22	76.7%	42	76.6%	62	76.6%	82	76.5%
3	77.0%	23	76.7%	43	76.6%	63	76.6%	83	76.5%
4	76.9%	24	76.7%	44	76.6%	64	76.6%	84	76.5%
5	76.8%	25	76.7%	45	76.6%	65	76.6%	85	76.5%
6	76.8%	26	76.7%	46	76.6%	66	76.6%	86	76.5%
7	76.8%	27	76.7%	47	76.6%	67	76.6%	87	76.5%
8	76.8%	28	76.7%	48	76.6%	68	76.5%	88	76.5%
9	76.7%	29	76.6%	49	76.6%	69	76.5%	89	76.5%
10	76.7%	30	76.6%	50	76.6%	70	76.5%	90	76.5%
11	76.7%	31	76.6%	51	76.6%	71	76.5%	91	76.5%
12	76.7%	32	76.6%	52	76.6%	72	76.5%	92	76.5%
13	76.7%	33	76.6%	53	76.6%	73	76.5%	93	76.5%
14	76.7%	34	76.6%	54	76.6%	74	76.5%	94	76.5%
15	76.7%	35	76.6%	55	76.6%	75	76.5%	95	76.5%
16	76.7%	36	76.6%	56	76.6%	76	76.5%	96	76.5%
17	76.7%	37	76.6%	57	76.6%	77	76.5%	97	76.5%
18	76.7%	38	76.6%	58	76.6%	78	76.5%	98	76.5%
19	76.7%	39	76.6%	59	76.6%	79	76.5%	99	76.5%
20	76.7%	40	76.6%	60	76.6%	80	76.5%	100	76.5%

PTSSK calibrations									
No.	NSE	No.	NSE	No.	NSE	No.	NSE	No.	NSE
101	76.5%	121	76.4%	141	76.3%	161	76.1%	181	75.6%
102	76.5%	122	76.4%	142	76.3%	162	76.1%	182	75.6%
103	76.5%	123	76.4%	143	76.3%	163	76.0%	183	75.5%
104	76.5%	124	76.4%	144	76.3%	164	76.0%	184	75.5%
105	76.4%	125	76.4%	145	76.3%	165	76.0%	185	75.5%
106	76.4%	126	76.4%	146	76.3%	166	75.9%	186	75.4%
107	76.4%	127	76.4%	147	76.3%	167	75.9%	187	75.4%
108	76.4%	128	76.4%	148	76.3%	168	75.9%	188	75.4%
109	76.4%	129	76.4%	149	76.3%	169	75.9%	189	75.4%
110	76.4%	130	76.4%	150	76.2%	170	75.9%	190	75.3%
111	76.4%	131	76.4%	151	76.2%	171	75.9%	191	75.3%
112	76.4%	132	76.4%	152	76.2%	172	75.8%	192	75.3%
113	76.4%	133	76.4%	153	76.2%	173	75.8%	193	75.2%
114	76.4%	134	76.4%	154	76.2%	174	75.8%	194	75.1%
115	76.4%	135	76.4%	155	76.2%	175	75.8%	195	75.1%
116	76.4%	136	76.4%	156	76.2%	176	75.8%	196	75.1%
117	76.4%	137	76.4%	157	76.2%	177	75.7%	197	75.1%
118	76.4%	138	76.4%	158	76.2%	178	75.7%	198	75.0%
119	76.4%	139	76.3%	159	76.2%	179	75.7%	199	74.9%
120	76.4%	140	76.3%	160	76.1%	180	75.6%	200	74.5%

Table Ap7.2 Top 36 calibration results parameters for PTSSK method

PTSSK	1	2	3	4	5	6	7	8	9
kirchner.c1	-3.909	-3.807	-3.764	-3.948	-3.732	-3.606	-3.606	-3.793	-3.770
kirchner.c2	0.392	0.402	0.435	0.317	0.415	0.469	0.469	0.296	0.444
kirchner.c3	-0.030	-0.030	-0.030	-0.030	-0.031	-0.032	-0.032	-0.032	-0.030
ae.ae_scale_factor	0.402	0.509	0.364	0.453	0.399	0.640	0.640	0.663	0.330
ss.alpha_0	27.56	53.24	24.75	20.12	45.04	28.70	28.70	63.30	38.50
ss.d_range	433	242	343	539	259	399	399	312	256
ss.unit_size	0.148	0.186	0.171	0.120	0.209	0.130	0.130	0.206	0.236
ss.max_water_fraction	0.143	0.116	0.073	0.070	0.066	0.076	0.076	0.132	0.057
ss.tx	-0.252	-0.237	-0.202	-0.282	-0.161	0.126	0.126	-0.247	-0.246
ss.cx	7.666	7.102	7.011	7.851	6.409	6.461	6.461	6.142	7.246
ss.ts	0.296	0.358	0.335	0.581	0.175	0.204	0.204	0.323	0.416
ss.cfr	0.001	0.001	0.000	0.000	0.005	0.000	0.000	0.001	0.000
p_corr.scale_factor	0.815	0.790	0.813	0.789	0.802	0.781	0.781	0.756	0.821
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	77.2%	77.0%	77.0%	76.9%	76.8%	76.8%	76.8%	76.8%	76.7%
NSE (2012 - 2017)	79%	79%	79%	79%	79%	78%	78%	78%	79%

PTSSK	10	11	12	13	14	15	16	17	18
kirchner.c1	-3.770	-3.921	-3.713	-4.058	-3.667	-3.667	-3.807	-3.709	-3.689
kirchner.c2	0.444	0.355	0.410	0.148	0.452	0.452	0.357	0.445	0.458
kirchner.c3	-0.030	-0.031	-0.030	-0.059	-0.030	-0.030	-0.035	-0.030	-0.030
ae.ae_scale_factor	0.330	0.557	0.355	0.439	0.423	0.423	0.598	0.358	0.394
ss.alpha_0	38.50	30.31	41.60	43.48	30.19	30.19	69.94	12.82	21.88
ss.d_range	256	376	65	221	298	298	118	309	562
ss.unit_size	0.236	0.076	0.399	0.069	0.287	0.287	0.034	0.154	0.063
ss.max_water_fraction	0.057	0.059	0.080	0.267	0.050	0.050	0.062	0.000	0.000
ss.tx	-0.246	-0.344	-0.035	-0.447	0.060	0.060	-0.038	-0.034	-0.234
ss.cx	7.246	8.089	6.320	6.284	6.543	6.543	6.909	6.656	6.815
ss.ts	0.416	0.566	0.335	0.040	0.346	0.346	0.407	0.395	0.461
ss.cfr	0.000	0.001	0.000	0.001	0.000	0.000	0.001	1.198	2.037
p_corr.scale_factor	0.821	0.786	0.796	0.777	0.797	0.797	0.771	0.815	0.813
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%
NSE (2012 - 2017)	79%	79%	78%	79%	78%	78%	79%	78%	78%

PTSSK	19	20	21	22	23	24	25	26	27
kirchner.c1	-3.644	-3.644	-3.719	-3.690	-3.788	-3.782	-3.720	-3.743	-3.675
kirchner.c2	0.467	0.467	0.384	0.454	0.421	0.416	0.426	0.437	0.470
kirchner.c3	-0.030	-0.030	-0.032	-0.030	-0.030	-0.030	-0.030	-0.030	-0.030
ae.ae_scale_factor	0.393	0.393	0.476	0.359	0.369	0.390	0.443	0.362	0.365
ss.alpha_0	36.21	36.21	55.73	31.81	30.89	28.97	22.24	21.05	28.59
ss.d_range	565	565	438	514	599	473	458	465	459
ss.unit_size	0.068	0.068	0.268	0.092	0.056	0.068	0.099	0.123	0.105
ss.max_water_fraction	0.000	0.000	0.073	0.000	0.000	0.000	0.000	0.000	0.000
ss.tx	0.089	0.089	0.107	-0.193	-0.246	-0.024	-0.076	-0.176	-0.210
ss.cx	6.305	6.305	5.936	6.648	7.147	6.863	6.879	6.700	6.622
ss.ts	0.298	0.298	0.212	0.435	0.517	0.423	0.518	0.386	0.397
ss.cfr	1.284	1.284	0.000	1.412	0.022	2.230	2.585	0.267	3.268
p_corr.scale_factor	0.806	0.806	0.780	0.813	0.813	0.809	0.795	0.820	0.817
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%	76.7%
NSE (2012 - 2017)	78%	78%	78%	78%	78%	78%	78%	78%	78%

PTSSK	28	29	30	31	32	33	34	35	36
kirchner.c1	-3.675	-3.770	-3.755	-3.656	-3.672	-3.675	-3.789	-3.787	-3.970
kirchner.c2	0.470	0.426	0.352	0.477	0.461	0.466	0.417	0.419	0.246
kirchner.c3	-0.030	-0.030	-0.031	-0.030	-0.030	-0.030	-0.030	-0.030	-0.049
ae.ae_scale_factor	0.365	0.337	0.675	0.346	0.373	0.382	0.397	0.427	0.430
ss.alpha_0	28.59	20.60	35.14	15.13	34.15	43.31	27.79	17.31	25.46
ss.d_range	459	333	363	344	512	549	558	484	395
ss.unit_size	0.105	0.133	0.205	0.185	0.062	0.082	0.096	0.074	0.193
ss.max_water_fraction	0.000	0.000	0.149	0.000	0.000	0.000	0.000	0.000	0.122
ss.tx	-0.210	-0.035	-0.875	-0.042	-0.220	-0.197	-0.176	-0.246	-0.097
ss.cx	6.622	6.792	6.442	6.598	6.694	6.615	7.338	7.463	8.010
ss.ts	0.397	0.400	0.365	0.357	0.475	0.412	0.578	0.626	0.455
ss.cfr	3.268	1.074	0.003	1.946	1.267	0.393	1.547	1.291	0.001
p_corr.scale_factor	0.817	0.818	0.756	0.818	0.808	0.813	0.808	0.804	0.784
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	76.7%	76.6%	76.6%	76.6%	76.6%	76.6%	76.6%	76.6%	76.6%
NSE (2012 - 2017)	78%	78%	77%	78%	78%	78%	78%	78%	78%

Table Ap7.3 200 calibration results for PTHSK method

PTHSK calibrations									
No.	NSE	No.	NSE	No.	NSE	No.	NSE	No.	NSE
1	77.8%	21	77.3%	41	76.9%	61	76.7%	81	76.5%
2	77.7%	22	77.2%	42	76.9%	62	76.7%	82	76.5%
3	77.6%	23	77.2%	43	76.9%	63	76.7%	83	76.5%
4	77.6%	24	77.2%	44	76.8%	64	76.7%	84	76.5%
5	77.6%	25	77.2%	45	76.8%	65	76.7%	85	76.5%
6	77.5%	26	77.1%	46	76.8%	66	76.6%	86	76.5%
7	77.5%	27	77.1%	47	76.8%	67	76.6%	87	76.5%
8	77.5%	28	77.1%	48	76.8%	68	76.6%	88	76.5%
9	77.5%	29	77.1%	49	76.8%	69	76.6%	89	76.4%
10	77.5%	30	77.1%	50	76.8%	70	76.6%	90	76.4%
11	77.4%	31	77.1%	51	76.8%	71	76.6%	91	76.4%
12	77.4%	32	77.0%	52	76.7%	72	76.6%	92	76.4%
13	77.4%	33	77.0%	53	76.7%	73	76.6%	93	76.4%
14	77.4%	34	77.0%	54	76.7%	74	76.6%	94	76.4%
15	77.4%	35	77.0%	55	76.7%	75	76.6%	95	76.4%
16	77.4%	36	77.0%	56	76.7%	76	76.6%	96	76.3%
17	77.4%	37	77.0%	57	76.7%	77	76.6%	97	76.3%
18	77.4%	38	76.9%	58	76.7%	78	76.5%	98	76.3%
19	77.3%	39	76.9%	59	76.7%	79	76.5%	99	76.3%
20	77.3%	40	76.9%	60	76.7%	80	76.5%	100	76.3%

PTHSK calibrations									
No.	NSE	No.	NSE	No.	NSE	No.	NSE	No.	NSE
101	76.3%	121	76.1%	141	76.0%	161	75.8%	181	75.5%
102	76.3%	122	76.1%	142	76.0%	162	75.8%	182	75.5%
103	76.3%	123	76.1%	143	75.9%	163	75.8%	183	75.5%
104	76.3%	124	76.1%	144	75.9%	164	75.8%	184	75.5%
105	76.2%	125	76.1%	145	75.9%	165	75.7%	185	75.4%
106	76.2%	126	76.1%	146	75.9%	166	75.7%	186	75.4%
107	76.2%	127	76.1%	147	75.9%	167	75.7%	187	75.4%
108	76.2%	128	76.1%	148	75.9%	168	75.6%	188	75.3%
109	76.2%	129	76.1%	149	75.9%	169	75.6%	189	75.3%
110	76.2%	130	76.1%	150	75.9%	170	75.6%	190	75.3%
111	76.2%	131	76.1%	151	75.9%	171	75.6%	191	75.3%
112	76.2%	132	76.0%	152	75.9%	172	75.6%	192	75.2%
113	76.2%	133	76.0%	153	75.9%	173	75.6%	193	75.2%
114	76.2%	134	76.0%	154	75.9%	174	75.6%	194	75.1%
115	76.2%	135	76.0%	155	75.9%	175	75.6%	195	75.1%
116	76.2%	136	76.0%	156	75.8%	176	75.5%	196	75.1%
117	76.1%	137	76.0%	157	75.8%	177	75.5%	197	75.1%
118	76.1%	138	76.0%	158	75.8%	178	75.5%	198	75.0%
119	76.1%	139	76.0%	159	75.8%	179	75.5%	199	75.0%
120	76.1%	140	76.0%	160	75.8%	180	75.5%	200	74.6%

Table Ap7.4 Top 36 calibration results parameters for PTHSK method

PTHSK	1	2	3	4	5	6	7	8	9
kirchner.c1	-3.619	-3.703	-3.704	-3.678	-3.718	-3.702	-3.591	-3.649	-3.774
kirchner.c2	0.456	0.432	0.416	0.406	0.416	0.418	0.470	0.422	0.425
kirchner.c3	-0.030	-0.035	-0.035	-0.037	-0.030	-0.030	-0.030	-0.031	-0.030
ae.ae_scale_factor	0.509	0.340	0.287	0.356	0.385	0.623	0.317	0.526	0.594
hs.lw	0.748	0.524	0.582	0.464	0.281	0.544	0.432	0.556	0.577
hs.tx	-0.217	-0.310	-0.632	-0.251	-0.639	-0.008	0.089	0.108	-0.079
hs.cx	5.682	5.782	5.655	6.040	6.232	5.986	5.333	5.604	6.185
hs.ts	-0.598	-0.473	-0.478	-0.277	0.071	-0.429	-0.588	-0.585	-0.541
hs.cfr	0.0004	0.0005	0.0006	0.0006	0.0008	0.0008	0.0007	0.0009	0.0012
gm.dtf	6	6	6	6	6	6	6	6	6
p_corr.scale_factor	0.832	0.857	0.867	0.852	0.846	0.817	0.858	0.832	0.820
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	77.8%	77.7%	77.6%	77.6%	77.6%	77.5%	77.5%	77.5%	77.5%
NSE (2012 - 2017)	79%	79%	79%	79%	79%	80%	79%	79%	80%

PTHSK	10	11	12	13	14	15	16	17	18
kirchner.c1	-3.627	-3.578	-3.725	-3.888	-3.707	-3.824	-3.824	-3.742	-3.722
kirchner.c2	0.391	0.461	0.470	0.301	0.353	0.273	0.273	0.386	0.411
kirchner.c3	-0.034	-0.030	-0.030	-0.030	-0.032	-0.051	-0.051	-0.033	-0.035
ae.ae_scale_factor	0.484	0.709	0.307	0.577	0.392	0.457	0.457	0.445	0.417
hs.lw	0.327	0.614	0.335	0.288	0.488	0.574	0.574	0.299	0.202
hs.tx	-0.245	0.002	-0.132	-0.305	-0.229	-0.300	-0.300	0.024	-0.201
hs.cx	5.356	5.665	6.219	6.625	5.158	6.422	6.422	6.210	6.646
hs.ts	-0.279	-0.566	-0.370	0.088	-0.511	-0.248	-0.248	-0.151	0.086
hs.cfr	0.0008	0.0007	0.0011	0.0009	0.0004	0.0008	0.0008	0.0008	0.0010
gm.dtf	6	6	6	6	6	6	6	6	6
p_corr.scale_factor	0.819	0.820	0.859	0.808	0.835	0.821	0.821	0.832	0.838
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	77.5%	77.4%	77.4%	77.4%	77.4%	77.4%	77.4%	77.4%	77.4%
NSE (2012 - 2017)	78%	79%	80%	80%	79%	80%	80%	80%	79%

PTHSK	19	20	21	22	23	24	25	26	27
kirchner.c1	-3.483	-3.840	-3.605	-3.733	-3.776	-3.810	-3.671	-3.687	-3.795
kirchner.c2	0.516	0.287	0.374	0.326	0.301	0.249	0.349	0.344	0.268
kirchner.c3	-0.030	-0.051	-0.039	-0.033	-0.036	-0.050	-0.048	-0.043	-0.055
ae.ae_scale_factor	0.608	0.426	0.758	0.475	0.565	0.813	0.477	0.583	0.603
hs.lw	0.745	0.690	0.351	0.253	0.315	0.632	0.415	0.264	0.348
hs.tx	0.013	-0.173	-0.162	-0.575	-0.072	-0.273	-0.025	-0.077	-0.327
hs.cx	5.129	6.143	6.053	5.684	6.094	6.359	5.711	5.573	6.124
hs.ts	-0.794	-0.472	-0.053	0.101	0.001	-0.184	-0.326	-0.185	-0.126
hs.cfr	0.0004	0.0004	0.0008	0.0007	0.0006	0.0004	0.0006	0.0008	0.0010
gm.dtf	6	6	6	6	6	6	6	6	6
p_corr.scale_factor	0.818	0.845	0.787	0.827	0.817	0.788	0.821	0.813	0.811
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	77.3%	77.3%	77.3%	77.2%	77.2%	77.2%	77.2%	77.1%	77.1%
NSE (2012 - 2017)	78%	80%	78%	78%	79%	79%	79%	79%	79%

PTHSK	28	29	30	31	32	33	34	35	36
kirchner.c1	-3.630	-3.852	-3.686	-4.058	-3.629	-3.865	-3.597	-3.755	-3.644
kirchner.c2	0.458	0.273	0.430	0.143	0.403	0.204	0.376	0.308	0.359
kirchner.c3	-0.030	-0.053	-0.030	-0.063	-0.034	-0.061	-0.043	-0.049	-0.043
ae.ae_scale_factor	0.310	0.334	0.758	0.471	0.484	0.417	0.507	0.394	0.361
hs.lw	0.136	0.750	0.750	0.634	0.129	0.476	0.184	0.373	0.210
hs.tx	0.191	-0.039	0.613	-0.635	0.034	-0.280	0.079	0.211	0.120
hs.cx	5.840	6.251	5.838	7.861	5.447	6.063	5.651	5.900	5.247
hs.ts	-0.120	-0.493	-0.731	0.114	-0.072	-0.204	-0.067	-0.366	-0.278
hs.cfr	0.0008	0.0003	0.0009	0.0006	0.0009	0.0004	0.0009	0.0010	0.0006
gm.dtf	6	6	6	6	6	6	6	6	6
p_corr.scale_factor	0.852	0.851	0.805	0.834	0.823	0.826	0.824	0.834	0.788
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	77.1%	77.1%	77.1%	77.1%	77.0%	77.0%	77.0%	77.0%	76.9%
NSE (2012 - 2017)	79%	80%	79%	80%	78%	79%	78%	79%	79%

Table Ap7.5 Calibration results for PTGSK method

PTGSK calibrations									
No.	NSE	No.	NSE	No.	NSE	No.	NSE	No.	NSE
1	79.9%	21	78.5%	41	77.7%	61	77.4%	81	77.2%
2	79.8%	22	78.4%	42	77.7%	62	77.4%	82	77.2%
3	79.7%	23	78.4%	43	77.7%	63	77.4%	83	77.2%
4	79.4%	24	78.4%	44	77.7%	64	77.4%	84	77.2%
5	79.4%	25	78.3%	45	77.7%	65	77.3%	85	77.2%
6	79.3%	26	78.3%	46	77.6%	66	77.3%	86	77.2%
7	79.3%	27	78.3%	47	77.6%	67	77.3%	87	77.2%
8	79.3%	28	78.2%	48	77.6%	68	77.3%	88	77.2%
9	79.2%	29	78.2%	49	77.5%	69	77.3%	89	77.2%
10	79.1%	30	78.1%	50	77.5%	70	77.3%	90	77.2%
11	79.1%	31	78.0%	51	77.5%	71	77.3%	91	77.2%
12	79.1%	32	77.9%	52	77.5%	72	77.3%	92	77.1%
13	79.0%	33	77.9%	53	77.4%	73	77.3%	93	77.1%
14	79.0%	34	77.8%	54	77.4%	74	77.3%	94	77.1%
15	78.8%	35	77.8%	55	77.4%	75	77.3%	95	77.1%
16	78.6%	36	77.8%	56	77.4%	76	77.2%	96	77.1%
17	78.6%	37	77.8%	57	77.4%	77	77.2%	97	77.1%
18	78.6%	38	77.8%	58	77.4%	78	77.2%	98	77.1%
19	78.5%	39	77.7%	59	77.4%	79	77.2%	99	77.1%
20	78.5%	40	77.7%	60	77.4%	80	77.2%	100	77.1%

PTGSK calibrations									
No.	NSE	No.	NSE	No.	NSE	No.	NSE	No.	NSE
101	77.1%	121	77.0%	141	76.9%	161	76.7%	181	76.4%
102	77.1%	122	77.0%	142	76.9%	162	76.7%	182	76.3%
103	77.1%	123	77.0%	143	76.9%	163	76.7%	183	76.3%
104	77.1%	124	77.0%	144	76.9%	164	76.7%	184	76.2%
105	77.1%	125	77.0%	145	76.9%	165	76.7%	185	76.2%
106	77.1%	126	77.0%	146	76.9%	166	76.6%	186	76.1%
107	77.1%	127	77.0%	147	76.8%	167	76.6%	187	76.1%
108	77.1%	128	77.0%	148	76.8%	168	76.6%	188	76.1%
109	77.1%	129	77.0%	149	76.8%	169	76.6%	189	76.0%
110	77.1%	130	76.9%	150	76.8%	170	76.6%	190	76.0%
111	77.0%	131	76.9%	151	76.8%	171	76.6%	191	76.0%
112	77.0%	132	76.9%	152	76.8%	172	76.5%	192	75.9%
113	77.0%	133	76.9%	153	76.8%	173	76.5%	193	75.6%
114	77.0%	134	76.9%	154	76.8%	174	76.4%	194	75.3%
115	77.0%	135	76.9%	155	76.8%	175	76.4%	195	75.0%
116	77.0%	136	76.9%	156	76.8%	176	76.4%	196	74.3%
117	77.0%	137	76.9%	157	76.8%	177	76.4%	197	74.2%
118	77.0%	138	76.9%	158	76.8%	178	76.4%	198	73.3%
119	77.0%	139	76.9%	159	76.8%	179	76.4%	199	73.2%
120	77.0%	140	76.9%	160	76.8%	180	76.4%	200	72.8%

Table Ap7.6 Top 36 calibration results parameters for PTGSK method

PTGSK	1	2	3	4	5	6	7	8	9
kirchner.c1	-3.659	-3.647	-3.634	-3.764	-3.838	-3.555	-3.724	-3.915	-3.684
kirchner.c2	0.498	0.493	0.516	0.311	0.280	0.515	0.306	0.242	0.320
kirchner.c3	-0.010	-0.010	-0.010	-0.040	-0.040	-0.010	-0.040	-0.040	-0.040
ae.ae_scale_factor	0.388	0.399	0.339	0.658	0.552	0.346	0.551	0.551	0.550
gs.tx	-1.297	-0.931	-1.293	-0.943	-0.943	-0.791	-0.916	-0.997	-0.891
gs.wind_scale	0.606	0.495	0.625	0.543	0.520	0.603	0.633	0.615	0.624
gs.max_water	0.178	0.079	0.166	0.137	0.127	0.099	0.144	0.153	0.161
gs.wind_const	4.727	5.998	4.111	5.356	5.427	4.342	4.627	4.525	4.643
gs.fast_albedo_decay_rate	1.265	1.001	1.226	1.312	1.228	2.467	1.612	1.018	1.471
gs.slow_albedo_decay_rate	9.180	34.430	57.769	19.274	23.155	36.663	7.669	11.927	29.151
gs.surface_magnitude	49.17	56.41	67.31	50.60	19.80	36.05	45.49	32.96	32.10
gs.max_albedo	0.867	0.894	0.840	0.878	0.870	0.856	0.886	0.869	0.890
gs.min_albedo	0.647	0.652	0.644	0.635	0.644	0.572	0.623	0.650	0.611
gs.snowfall_reset_depth	6.200	10.147	6.963	6.654	8.264	12.510	7.026	6.916	7.116
gs.snow_cv	0.374	0.329	0.459	0.388	0.331	0.472	0.514	0.232	0.532
gs.glacier_albedo	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
p_corr.scale_factor	0.784	0.780	0.799	0.747	0.752	0.783	0.756	0.746	0.756
gs.snow_cv_forest_factor	0	0	0	0	0	0	0	0	0
gs.snow_cv_altitude_factor	0	0	0	0	0	0	0	0	0
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gs.initial_bare_ground_fraction	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
gs.winter_end_day_of_year	107	115	89	109	93	100	93	104	105
gs.calculate_iso_pot_energy	0	0	0	0	0	0	0	0	0
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gs.n_winter_days	235	206	192	255	193	226	245	244	217
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	80%	80%	80%	79%	79%	79%	79%	79%	79%
NSE (2012 - 2017)	74%	75%	75%	75%	75%	75%	75%	75%	75%

PTGSK	10	11	12	13	14	15	16	17	18
kirchner.c1	-3.758	-3.725	-3.835	-3.826	-3.744	-3.773	-3.661	-3.610	-3.745
kirchner.c2	0.306	0.309	0.264	0.229	0.301	0.284	0.425	0.437	0.253
kirchner.c3	-0.040	-0.040	-0.040	-0.048	-0.040	-0.040	-0.013	-0.010	-0.044
ae.ae_scale_factor	0.583	0.579	0.785	0.926	0.948	0.790	0.662	0.959	1.298
gs.tx	-0.932	-0.857	-0.891	-0.922	-1.091	-0.995	-1.042	-0.887	-0.922
gs.wind_scale	0.729	0.699	0.644	0.537	0.524	0.545	0.726	0.582	0.606
gs.max_water	0.140	0.155	0.137	0.156	0.156	0.152	0.129	0.148	0.160
gs.wind_const	3.087	3.638	4.102	5.413	5.108	4.784	2.109	3.933	4.472
gs.fast_albedo_decay_rate	1.200	1.555	1.024	1.057	1.612	3.354	4.498	4.007	2.546
gs.slow_albedo_decay_rate	26.862	22.608	21.656	28.552	23.057	13.184	6.321	10.724	8.922
gs.surface_magnitude	45.05	32.70	29.01	44.80	37.48	58.08	31.39	42.13	42.24
gs.max_albedo	0.884	0.877	0.880	0.884	0.860	0.837	0.803	0.832	0.862
gs.min_albedo	0.616	0.603	0.639	0.646	0.630	0.580	0.579	0.527	0.595
gs.snowfall_reset_depth	8.103	6.838	7.334	5.859	6.066	8.200	10.428	8.020	6.888
gs.snow_cv	0.378	0.455	0.252	0.348	0.357	0.390	0.268	0.412	0.380
gs.glacier_albedo	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
p_corr.scale_factor	0.752	0.750	0.730	0.722	0.727	0.736	0.744	0.721	0.706
gs.snow_cv_forest_factor	0	0	0	0	0	0	0	0	0
gs.snow_cv_altitude_factor	0	0	0	0	0	0	0	0	0
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gs.initial_bare_ground_fraction	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
gs.winter_end_day_of_year	84	94	98	114	100	111	96	104	99
gs.calculate_iso_pot_energy	0	0	0	0	0	0	0	0	0
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gs.n_winter_days	243	212	247	182	221	214	242	221	209
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	79%	79%	79%	79%	79%	79%	79%	79%	79%
NSE (2012 - 2017)	74%	74%	74%	73%	74%	75%	73%	74%	74%

PTGSK	19	20	21	22	23	24	25	26	27
kirchner.c1	-3.593	-3.874	-3.806	-3.856	-3.991	-3.760	-3.730	-3.632	-3.809
kirchner.c2	0.441	0.148	0.282	0.179	0.037	0.248	0.253	0.287	0.196
kirchner.c3	-0.016	-0.054	-0.033	-0.052	-0.075	-0.044	-0.040	-0.040	-0.045
ae.ae_scale_factor	0.789	1.023	0.466	0.901	0.550	0.980	1.323	1.363	1.358
gs.tx	-1.000	-0.893	-0.887	-0.888	-0.913	-0.919	-0.857	-0.741	-0.792
gs.wind_scale	0.647	0.580	0.669	0.666	0.749	0.561	0.632	0.517	0.646
gs.max_water	0.138	0.145	0.169	0.167	0.120	0.125	0.119	0.148	0.160
gs.wind_const	3.572	4.652	3.152	3.519	2.032	3.993	4.168	5.730	3.763
gs.fast_albedo_decay_rate	4.304	2.715	4.061	2.508	1.841	3.189	2.433	3.196	2.551
gs.slow_albedo_decay_rate	35.521	21.991	41.864	20.124	17.648	21.257	20.636	27.558	18.877
gs.surface_magnitude	41.71	50.62	37.79	60.53	30.40	31.25	25.25	31.43	50.42
gs.max_albedo	0.827	0.859	0.817	0.837	0.835	0.835	0.871	0.874	0.847
gs.min_albedo	0.542	0.580	0.531	0.578	0.590	0.552	0.579	0.562	0.588
gs.snowfall_reset_depth	9.470	7.026	10.051	6.707	5.564	7.154	6.271	8.990	9.985
gs.snow_cv	0.356	0.385	0.440	0.486	0.316	0.382	0.473	0.549	0.387
gs.glacier_albedo	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
p_corr.scale_factor	0.734	0.714	0.750	0.734	0.744	0.722	0.711	0.700	0.700
gs.snow_cv_forest_factor	0	0	0	0	0	0	0	0	0
gs.snow_cv_altitude_factor	0	0	0	0	0	0	0	0	0
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gs.initial_bare_ground_fraction	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
gs.winter_end_day_of_year	121	105	90	97	110	108	102	104	104
gs.calculate_iso_pot_energy	0	0	0	0	0	0	0	0	0
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gs.n_winter_days	246	238	221	225	240	240	234	197	224
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	79%	79%	78%	78%	78%	78%	78%	78%	78%
NSE (2012 - 2017)	74%	74%	74%	74%	73%	74%	74%	74%	73%

PTGSK	28	29	30	31	32	33	34	35	36
kirchner.c1	-3.947	-3.812	-3.902	-4.051	-3.940	-3.980	-3.916	-3.824	-3.821
kirchner.c2	0.004	0.143	0.141	-0.047	0.040	0.030	0.049	0.152	0.178
kirchner.c3	-0.081	-0.058	-0.057	-0.076	-0.070	-0.076	-0.070	-0.062	-0.049
ae.ae_scale_factor	0.617	1.467	0.863	0.898	1.295	0.776	1.364	0.953	0.768
gs.tx	-0.934	-0.856	-0.887	-0.913	-0.741	-0.857	-0.738	-1.067	-0.918
gs.wind_scale	0.512	0.592	0.572	0.646	0.699	0.630	0.582	0.585	0.648
gs.max_water	0.165	0.100	0.142	0.143	0.133	0.127	0.115	0.142	0.148
gs.wind_const	5.181	4.054	4.653	3.732	3.372	3.806	4.019	3.687	3.366
gs.fast_albedo_decay_rate	3.539	2.281	3.896	2.968	2.055	3.833	2.796	4.802	7.464
gs.slow_albedo_decay_rate	30.050	21.780	20.471	17.498	26.510	17.779	10.579	18.525	21.942
gs.surface_magnitude	49.35	31.17	35.41	49.33	54.58	45.71	65.43	45.41	33.28
gs.max_albedo	0.841	0.853	0.850	0.851	0.877	0.841	0.838	0.809	0.810
gs.min_albedo	0.573	0.596	0.521	0.566	0.574	0.525	0.570	0.560	0.488
gs.snowfall_reset_depth	7.742	6.312	6.010	7.277	6.419	6.836	5.837	6.893	10.259
gs.snow_cv	0.482	0.363	0.392	0.348	0.430	0.336	0.481	0.450	0.406
gs.glacier_albedo	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
p_corr.scale_factor	0.721	0.699	0.720	0.711	0.702	0.723	0.702	0.727	0.721
gs.snow_cv_forest_factor	0	0	0	0	0	0	0	0	0
gs.snow_cv_altitude_factor	0	0	0	0	0	0	0	0	0
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gs.initial_bare_ground_fraction	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
gs.winter_end_day_of_year	103	110	99	109	101	107	89	109	106
gs.calculate_iso_pot_energy	0	0	0	0	0	0	0	0	0
gm.dtf	6	6	6	6	6	6	6	6	6
routing.velocity	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3
gs.n_winter_days	203	208	211	223	222	221	220	202	203
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
NSE (2012 - 2015)	78%	78%	78%	78%	78%	78%	78%	78%	78%
NSE (2012 - 2017)	74%	73%	74%	73%	73%	74%	73%	73%	74%

Table Ap7.7 All parameters ranges for all methods

Parameters	PTGSK				PTSSK				PTHSK			
	For Calibration		200 Calibration results		For Calibration		200 Calibration results		For Calibration		200 Calibration results	
	lower limit	Upper limit	Lowest	Highest	lower limit	Upper limit	Lowest	Highest	lower limit	Upper limit	Lowest	Highest
kirchner.c1	-8	0	-4.943	-3.555	-8	0	-4.41	-3.55	-8	0	-4.058	-3.483
kirchner.c2	-1	1.2	-0.740	0.516	-1	1.2	-0.04	0.49	-1	1.2	0.006	0.516
kirchner.c3	-0.25	-0.01	-0.120	-0.017	-0.25	-0.01	-0.09	-0.03	-0.25	-0.01	-0.089	-0.030
ae.ae_scale_factor	0.2	2.5	0.339	2.385	0.2	2.5	0.30	2.49	0.2	2.5	0.268	2.484
p_corr.scale_factor	0.5	1.5	0.624	0.799	0.5	1.5	0.71	0.86	0.5	1.5	0.686	0.838
gs.calculate_iso_pot_energy	0	0	0	0	-	-	-	-	-	-	-	-
gs.tx	-3	2	-1.297	-0.352	-	-	-	-	-	-	-	-
gs.wind_scale	0.5	6	0.495	0.769	-	-	-	-	-	-	-	-
gs.max_water	0.06	0.2	0.079	0.189	-	-	-	-	-	-	-	-
gs.wind_const	1	7	2.032	6.043	-	-	-	-	-	-	-	-
gs.fast_albedo_decay_rate	1	15	1.001	12.223	-	-	-	-	-	-	-	-
gs.slow_albedo_decay_rate	2	70	4.799	69.008	-	-	-	-	-	-	-	-
gs.surface_magnitude	10	70	17.664	67.314	-	-	-	-	-	-	-	-
gs.max_albedo	0.7	0.95	0.789	0.933	-	-	-	-	-	-	-	-
gs.min_albedo	0.4	0.7	0.423	0.652	-	-	-	-	-	-	-	-
gs.snowfall_reset_depth	4	9	4.710	14.274	-	-	-	-	-	-	-	-
gs.snow_cv	0.1	0.8	0.050	0.556	-	-	-	-	-	-	-	-
gs.glacier_albedo	0.4	0.4	0.4	0.4	-	-	-	-	-	-	-	-
gs.snow_cv_forest_factor	0	0	0	0	-	-	-	-	-	-	-	-
gs.snow_cv_altitude_factor	0	0	0	0	-	-	-	-	-	-	-	-
gs.initial_bare_ground_fraction	0.04	0.04	0.04	0.04	-	-	-	-	-	-	-	-
gs.winter_end_day_of_year	80	125	84	121	-	-	-	-	-	-	-	-
gs.n_winter_days	170	270	175	263	-	-	-	-	-	-	-	-
ss.alpha_0	-	-	-	-	8	75	10	70	-	-	-	-
ss.d_range	-	-	-	-	4	650	5	604	-	-	-	-
ss.unit_size	-	-	-	-	0.001	0.45	0.0012	0.399	-	-	-	-
ss.max_water_fraction	-	-	-	-	0	0.35	0.0000	0.277	-	-	-	-
ss.tx	-	-	-	-	-1.2	1.2	-0.987	0.877	-	-	-	-
ss.cx	-	-	-	-	0	10	5.61	9.08	-	-	-	-
ss.ts	-	-	-	-	-1.2	1.2	-0.17	0.98	-	-	-	-
ss.cfr	-	-	-	-	0	4	0.0001	3.484	-	-	-	-
hs.lw	-	-	-	-	-	-	-	-	0	0.85	0	0.750
hs.tx	-	-	-	-	-	-	-	-	-1.2	1.2	-0.977	0.989
hs.cx	-	-	-	-	-	-	-	-	0	10	4.626	7.861
hs.ts	-	-	-	-	-	-	-	-	-1.2	1.2	-0.799	0.522
hs.cfr	-	-	-	-	-	-	-	-	0	1.2	0.0001	3.234
pt.albedo	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
pt.alpha	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26	1.26
gm.dtf	6	6	6	6	6	6	6	6	6	6	6	6
gm.direct_response	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475	0.475
routing.velocity	0	0	0	0	0	0	0	0	0	0	0	0
routing.alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
routing.beta	3	3	3	3	3	3	3	3	3	3	3	3

Appendix 8

Summary of SWE

calculations

Table Ap8.1 Summary of SWE calculation in passed cell in PTGSK method

Total absolute error	133	PTGSK																			
Cells	Cell No.	279	280	308	336	337	359	377	378	832	833	855	856	857	858	864	879	880	893	894	907
	Orientation	West	West	West	West	South-West	West	West	North-West	North-West	West	South-West	South-West	West	North-West	Easth	Easth	Easth	North	North	North-West
	Elevation gradient	11%	12%	11%	6%	7%	12%	18%	21%	7%	11%	9%	11%	11%	12%	15%	13%	15%	5%	5%	3%
2013	SWE Obs.	354	459	377	494	391	529	310	514	446	273	496	817	740	804	409	768	491	450	118	420
	SWE Model	285	283	325	373	348	394	506	454	251	276	285	316	330	386	277	239	232	218	219	219
	Real error	-69		-52	-121	-43	-135		-61		3	-211	-501	-409	-418	-131	-529		-232		-201
	Minimum	85	203	81	121	85	117	147	181	107	98	102	102	125	116	114	105	235	105	110	105
	Maximum	1103	1041	1499	1461	915	1554	715	3015	1647	1144	2237	4160	2416	2336	796	3249	1007	1619	123	2279
	Std	207	164	273	243	205	263	112	400	449	197	437	747	453	441	162	632	198	265	3	377
	mean + std	561	623	651	737	596	792	423	914	895	470	933	1564	1192	1245	571	1400	689	715	121	797
	mean - std	147	295	104	250	186	266	198	114	-2	77	59	70	287	362	246	136	293	185	115	43
	CV	58%	36%	72%	49%	52%	50%	36%	78%	101%	72%	88%	91%	61%	55%	40%	82%	40%	59%	2%	90%
	No.	737	115	1128	587	520	1115	161	679	83	207	997	1094	1010	972	406	989	67	1030	31	1063
166	Accuracy	69		52	121	43	135		61		3	211	501	409	418	131	529		232		201
2014	SWE Obs.	364	439	327	363	313	350	334	340	309	267	389	440	473	447	358	322	422	373	135	288
	SWE Model	223	220	276	361	326	445	573	522	245	279	318	377	399	453	377	350	338	317	310	322
	Real error	-141		-50	-2	12	95		182			-71	-63	-73	6	20	28		-56		34
	Minimum	19	199	19	76	76	67	67	76	21	84	94	84	104	63	10	10	29	29	39	10
	Maximum	800	798	793	791	641	797	627	804	809	703	868	871	858	871	702	725	725	720	263	725
	Std	194	168	182	121	123	161	141	166	246	141	179	193	195	213	191	207	212	183	90	202
	mean + std	558	606	509	485	436	511	475	506	555	407	568	633	668	660	549	529	634	556	225	490
	mean - std	170	271	144	242	190	188	194	174	63	126	209	247	277	234	167	115	211	191	45	86
	CV	53%	38%	56%	33%	39%	46%	42%	49%	80%	53%	46%	44%	41%	48%	53%	64%	50%	49%	67%	70%
	No.	737	120	1118	469	514	735	128	479	90	182	722	956	667	925	413	684	188	849	21	952
113	Accuracy	141		50	2	12	95		182			71	63	73	6	20	28		56		34
2015	SWE Obs.	313	422	236	373	320	366	281	412	489	249	394	733	656	600	347	479	496	372	133	341
	SWE Model	315	304	362	426	423	483	566	540	277	312	335	375	389	427	325	284	281	271	269	271
	Real error	2		125	53	103	117		128		63	-59	-359	-267	-173	-22	-194	-215	-101		-70
	Minimum	39	162	46	106	52	79	106	134	88	47	47	39	31	39	65	65	72	72	117	57
	Maximum	1105	1369	923	1155	1195	1411	561	2297	1618	1008	1700	4078	1537	1971	1517	1859	1698	1331	171	1646
	Std	236	254	168	221	204	226	97	320	431	178	324	759	334	336	202	375	376	228	15	305
	mean + std	549	676	405	594	524	592	378	733	920	427	718	1492	990	936	549	854	872	601	149	646
	mean - std	76	169	68	153	117	139	184	92	58	70	71	-25	322	264	145	103	120	144	118	36
	CV	76%	60%	71%	59%	64%	62%	35%	78%	88%	72%	82%	103%	51%	56%	58%	78%	76%	61%	11%	89%
	No.	718	139	1134	638	461	1109	127	672	74	203	988	1097	1002	965	392	672	332	1024	26	1058
124	Accuracy	2		125	53	103	117		128		63	59	359	267	173	22	194	215	101		70
2016	SWE Obs.	473	596	474	481	461	625	548	615	543	321	490	622	725	744	543	455	870	427	159	377
	SWE Model	310	304	359	409	382	435	534	498	308	333	347	371	384	414	326	298	302	287	282	289
	Real error	-163		-115	-72	-80	-189		-117		12	-143	-250	-340	-330	-218	-157	-568	-141		-88
	Minimum	97	345	97	187	160	160	223	124	134	100	126	92	126	67	86	51	289	43	112	26
	Maximum	1601	1346	1410	1640	1071	1775	896	2578	1448	1139	1795	1980	1636	1899	1119	1699	2072	1619	381	1965
	Std	281	216	257	203	212	356	123	374	419	192	276	432	304	362	245	375	454	251	58	334
	mean + std	754	811	731	684	673	980	672	989	962	513	766	1053	1028	1106	788	830	1324	678	218	711
	mean - std	192	380	217	278	250	269	425	241	123	129	214	190	421	382	299	80	415	177	101	44
	CV	59%	36%	54%	42%	46%	57%	23%	61%	77%	60%	56%	69%	42%	49%	45%	82%	52%	59%	36%	88%
	No.	731	132	1145	595	514	1142	139	676	72	209	985	1280	1013	973	417	822	204	1047	29	1061
145	Accuracy	163		115	72	80	189		117		12	143	250	340	330	218	157	568	141		88
2017	SWE Obs.	437	523	468	563	465	593	624	621	396	164	414	738	568	727	361	487	765	391	143	366
	SWE Model	274	270	373	423	421	479	623	553	294	323	352	449	446	516	374	315	314	293	293	293
	Real error	-164		-95	-140	-44	-115		-68			-62	-289	-122	-211	14	-172	-451	-97		-73
	Minimum	102	353	75	271	68	60	324	349	51	42	68	38	38	38	7	100	143	100	111	104
	Maximum	1136	1059	1324	1471	983	1564	1269	2010	1124	908	1366	2529	1581	1784	961	1408	1769	1173	212	1764
	Std	178	146	218	167	191	205	175	252	354	172	251	584	308	353	173	325	406	198	32	263
	mean + std	616	669	686	730	656	799	800	872	750	336	665	1322	876	1079	534	812	1171	588	175	629
	mean - std	259	376	250	396	273	388	449	369	41	-8	163	154	260	374	188	162	359	193	111	102
	CV	41%	28%	47%	30%	41%	35%	28%	41%	90%	105%	61%	79%	54%	49%	48%	67%	53%	51%	22%	72%
	No.	738	118	1118	590	515	1113	145	662	80	200	996	1124	1011	975	547	781	233	1023	22	1066
115	Accuracy	164		95	140	44	115		68			62	289	122	211	14	172	451	97		73
		Line 04								**Line 05**						**Line 06**					

Total absolute error	133	PTGSK																							
Cells	Cell No.	755	756	778	779	800	724	744	745	765	786	210	211	212	213	238	239	242	243	2627
	Orientation	South-East	South-East	Easth	North-East	North	West	North-East	Easth	North-East	North-East	North-West	North-West	North-West	North-West	North-West	North-West	North-East	North	North-West
	Elevation gradient	31%	31%	21%	10%	10%	11%	4%	9%	5%	3%	29%	23%	25%	5%	15%	25%	6%	5%	29%
2013	SWE Obs.	310	450	244	317	452	624	360	684	500	428	314	324	358	222	281	227	369	378	432
	SWE Model	374	395	353	327	293	290	270	271	256	245	408	354	308	277	467	459	279	261	456
	Real error		-55	110	11	-159			-413	-244	-183	94	30	-50	56		232	-90	-117	24
	Minimum	212	88	101	105	105	293	360	78	78	83	11	25	70	73	93	74	73	84	78
	Maximum	454	1105	1505	2420	1170	833	360	3663	2852	1666	1122	1216	1816	731	613	543	1064	994	2610
	Std	76	211	242	356	264	181	0	610	519	321	222	249	327	135	141	94	221	203	480
	mean + std	387	660	486	672	716	805	360	1295	1019	749	537	573	685	356	422	321	590	581	912
	mean - std	234	239	2	-39	188	444	360	74	-18	107	92	75	31	87	140	132	148	175	-48
	CV	25%	47%	99%	112%	58%	29%	0%	89%	104%	75%	71%	77%	91%	61%	50%	42%	60%	54%	111%
	No.	24	1003	1027	885	462	59	1	1024	1019	958	571	897	1062	962	123	241	422	208	333
166	Accuracy		55	110	11	159			413	244	183	94	30	50	56		232	90	117	24
2014	SWE Obs.	170	326	200	170	242	286	493	386	376	394	183	234	241	173	238	175	288	219	154
	SWE Model	563	608	548	498	464	339	311	310	277	263	391	319	256	232	445	439	242	218	436
	Real error		281	348	328	222			-76	-99	-130	208	85	15	59		265	-46	-1	282
	Minimum	32	16	16	0	16	19	272	19	19	48	26	17	17	17	35	26	35	26	26
	Maximum	219	556	556	555	539	425	650	810	810	785	680	692	712	510	428	514	656	635	428
	Std	58	130	125	145	151	127	106	230	218	204	164	167	193	136	87	129	165	135	134
	mean + std	228	457	324	315	393	413	599	616	593	598	347	401	435	309	325	303	453	354	289
	mean - std	112	196	75	25	91	159	387	156	158	189	18	67	48	37	152	46	123	84	20
	CV	34%	40%	62%	86%	62%	45%	22%	60%	58%	52%	90%	71%	80%	79%	36%	74%	57%	62%	87%
	No.	14	1017	975	811	365	60	15	709	659	674	507	925	1037	887	123	351	413	264	308
113	Accuracy		281	348	328	222			76	99	130	208	85	15	59		265	46	1	282
2015	SWE Obs.	Line didn´t pass this cell	382	326	371	507	251	332	461	472	519	212	218	319	292	184	213	353	431	308
	SWE Model		492	412	381	350	338	300	314	296	284	420	399	332	303	475	482	304	289	480
	Real error		110	86	10	-157			-147	-176	-235	208	181	13	11			-49		172
	Minimum		60	45	68	68	66	199	58	22	22	36	29	36	22	73	104	44	150	44
	Maximum		1234	1234	3022	1503	424	433	2743	2976	1849	773	678	1872	1284	293	471	1072	870	2607
	Std		237	242	456	360	119	73	493	527	324	123	135	272	219	51	67	200	193	395
	mean + std		619	568	827	867	371	406	954	999	843	335	353	592	511	235	279	553	624	703
	mean - std		145	84	-85	147	132	259	-32	-55	195	88	83	47	72	134	146	153	238	-87
	CV		62%	74%	123%	71%	48%	22%	107%	112%	62%	58%	62%	85%	75%	27%	31%	57%	45%	128%
	No.		983	1069	814	482	64	9	1006	1022	955	652	903	1084	943	67	91	442	161	427
124	Accuracy		110	86	10	157			147	176	235	208	181	13	11			49		172
2016	SWE Obs.	289	575	310	419	393	466	518	528	204	422	422	397	434	369	Line didn´t pass this cell	317	412	407	393
	SWE Model	384	409	373	354	332	364	340	330	326	313	458	412	348	332		490	327	312	500
	Real error		-165	63	-65	-61		-178	-198		-109	36	15	-86	-37		173	-85		107
	Minimum	63	24	32	0	111	313	70	70	150	79	180	136	119	171		215	223	17	223
	Maximum	693	1430	1270	3058	904	501	2002	2553	360	1396	1785	1094	1897	1176		1186	884	825	1238
	Std	105	273	235	512	191	47	356	460	49	298	244	202	292	138		130	124	185	221
	mean + std	394	848	545	931	583	513	874	988	252	720	666	600	727	507		447	537	592	614
	mean - std	184	301	75	-93	202	419	162	67	155	124	178	195	142	231		188	288	222	173
	CV	36%	48%	76%	122%	49%	10%	69%	87%	24%	71%	58%	51%	67%	37%		41%	30%	45%	56%
	No.	136	952	1050	831	469	67	1033	974	55	961	556	914	1068	1107		329	395	179	303
145	Accuracy		165	63	65	61		178	198		109	36	15	86	37		173	85		107
2017	SWE Obs.	500	378	383	419	509	688	618	475	231	398	552	455	405	358	Line didn´t pass this cell	488	365	378	590
	SWE Model	516	491	452	439	381	431	372	351	349	336	558	481	341	316		655	309	291	642
	Real error				20			-246	-124		-62	5	26	-64	-42			-56		
	Minimum	5	0	5	30	45	321	59	55	63	55	198	0	45	45		198	198	198	198
	Maximum	1360	1128	1085	2182	1203	878	2689	2204	523	1361	1402	1286	1407	1151		1275	770	1072	1745
	Std	313	260	257	431	294	165	454	398	109	273	274	270	237	199		283	187	245	344
	mean + std	814	637	641	850	803	853	1071	873	339	671	826	725	642	556		771	551	623	934
	mean - std	187	118	126	-12	214	523	164	77	122	125	279	186	167	159		205	178	133	246
	CV	63%	69%	67%	103%	58%	24%	73%	84%	47%	69%	50%	59%	59%	56%		58%	51%	65%	58%
	No.	97	137	168	217	113	55	1020	972	41	979	250	455	524	474		176	215	110	102
115	Accuracy				20			246	124		62	5	26	64	42			56		
		Line 07					Line 08					Line 09								

Table Ap8.2 Summary of SWE calculation in passed cell in PTHSK method

Total absolute error	129	PTHSK																			
Cells	Cell No.	279	280	308	336	337	359	377	378	832	833	855	856	857	858	864	879	880	893	894	907
	Orientation	West	West	West	West	South-West	West	West	North-West	North-West	West	South-West	South-West	West	North-West	Easth	Easth	Easth	North	North	North-West
	Elevation gradient	11%	12%	11%	6%	7%	12%	18%	21%	7%	11%	9%	11%	11%	12%	15%	13%	15%	5%	5%	3%
2013	SWE Obs.	354	459	377	494	391	529	310	514	446	273	496	817	740	804	409	768	491	450	118	420
	SWE Model	269	268	346	445	444	505	604	569	221	266	328	392	409	456	334	292	282	254	244	259
	Real error	-85		-31	-49	53	-24		55		-8	-168	-425	-330	-347	-75	-476		-195		-161
	Minimum	85	203	81	121	85	117	147	181	107	98	102	102	125	116	114	105	235	105	110	105
	Maximum	1103	1041	1499	1461	915	1554	715	3015	1647	1144	2237	4160	2416	2336	796	3249	1007	1619	123	2279
	Std	207	164	273	243	205	263	112	400	449	197	437	747	453	441	162	632	198	265	3	377
	mean + std	561	623	651	737	596	792	423	914	895	470	933	1564	1192	1245	571	1400	689	715	121	797
	mean - std	147	295	104	250	186	266	198	114	-2	77	59	70	287	362	246	136	293	185	115	43
	CV	58%	36%	72%	49%	52%	50%	36%	78%	101%	72%	88%	91%	61%	55%	40%	82%	40%	59%	2%	90%
	No.	737	115	1128	587	520	1115	161	679	83	207	997	1094	1010	972	406	989	67	1030	31	1063
158	Accuracy	85		31	49	53	24		55		8	168	425	330	347	75	476		195		161
2014	SWE Obs.	364	439	327	363	313	350	334	340	309	267	389	440	473	447	358	322	422	373	135	288
	SWE Model	304	263	461	536	500	575	673	657	290	351	378	437	467	513	418	397	382	366	355	361
	Real error	-60		134	172	186	225		317			-10	-3	-6	66	60	75		-7		73
	Minimum	19	199	19	76	76	67	67	76	21	84	94	84	104	63	10	10	29	29	39	10
	Maximum	800	798	793	791	641	797	627	804	809	703	868	871	858	871	702	725	725	720	263	725
	Std	194	168	182	121	123	161	141	166	246	141	179	193	195	213	191	207	212	183	90	202
	mean + std	558	606	509	485	436	511	475	506	555	407	568	633	668	660	549	529	634	556	225	490
	mean - std	170	271	144	242	190	188	194	174	63	126	209	247	277	234	167	115	211	191	45	86
	CV	53%	38%	56%	33%	39%	46%	42%	49%	80%	53%	46%	44%	41%	48%	53%	64%	50%	49%	67%	70%
	No.	737	120	1118	469	514	735	128	479	90	182	722	956	667	925	413	684	188	849	21	952
153	Accuracy	60		134	172	186	225		317			10	3	6	66	60	75		7		73
2015	SWE Obs.	313	422	236	373	320	366	281	412	489	249	394	733	656	600	347	479	496	372	133	341
	SWE Model	392	385	438	494	465	526	637	600	333	364	378	409	428	474	335	311	311	301	301	301
	Real error	80		201	121	145	160		188		116	-16	-325	-228	-126	-12	-168	-185	-71		-40
	Minimum	39	162	46	106	52	79	106	134	88	47	47	39	31	39	65	65	72	72	117	57
	Maximum	1105	1369	923	1155	1195	1411	561	2297	1618	1008	1700	4078	1537	1971	1517	1859	1698	1331	171	1646
	Std	236	254	168	221	204	226	97	320	431	178	324	759	334	336	202	375	376	228	15	305
	mean + std	549	676	405	594	524	592	378	733	920	427	718	1492	990	936	549	854	872	601	149	646
	mean - std	76	169	68	153	117	139	184	92	58	70	71	-25	322	264	145	103	120	144	118	36
	CV	76%	60%	71%	59%	64%	62%	35%	78%	88%	72%	82%	103%	51%	56%	58%	78%	76%	61%	11%	89%
	No.	718	139	1134	638	461	1109	127	672	74	203	988	1097	1002	965	392	672	332	1024	26	1058
139	Accuracy	80		201	121	145	160		188		116	16	325	228	126	12	168	185	71		40
2016	SWE Obs.	473	596	474	481	461	625	548	615	543	321	490	622	725	744	543	455	870	427	159	377
	SWE Model	363	357	432	494	464	531	608	594	361	389	404	427	442	473	374	357	357	343	340	343
	Real error	-110		-42	12	3	-94		-21		68	-86	-194	-282	-271	-169	-98	-512	-85		-35
	Minimum	97	345	97	187	160	160	223	124	134	100	126	92	126	67	86	51	289	43	112	26
	Maximum	1601	1346	1410	1640	1071	1775	896	2578	1448	1139	1795	1980	1636	1899	1119	1699	2072	1619	381	1965
	Std	281	216	257	203	212	356	123	374	419	192	276	432	304	362	245	375	454	251	58	334
	mean + std	754	811	731	684	673	980	672	989	962	513	766	1053	1028	1106	788	830	1324	678	218	711
	mean - std	192	380	217	278	250	269	425	241	123	129	214	190	421	382	299	80	415	177	101	44
	CV	59%	36%	54%	42%	46%	57%	23%	61%	77%	60%	56%	69%	42%	49%	45%	82%	52%	59%	36%	88%
	No.	731	132	1145	595	514	1142	139	676	72	209	985	1280	1013	973	417	822	204	1047	29	1061
112	Accuracy	110		42	12	3	94		21		68	86	194	282	271	169	98	512	85		35
2017	SWE Obs.	437	523	468	563	465	593	624	621	396	164	414	738	568	727	361	487	765	391	143	366
	SWE Model	341	314	459	602	566	650	754	727	281	379	509	544	568	609	451	420	421	368	363	371
	Real error	-96		-9	38	101	56		106			94	-194	-1	-117	90	-67	-344	-23		5
	Minimum	102	353	75	271	68	60	324	349	51	42	68	38	38	38	7	100	143	100	111	104
	Maximum	1136	1059	1324	1471	983	1564	1269	2010	1124	908	1366	2529	1581	1784	961	1408	1769	1173	212	1764
	Std	178	146	218	167	191	205	175	252	354	172	251	584	308	353	173	325	406	198	32	263
	mean + std	616	669	686	730	656	799	800	872	750	336	665	1322	876	1079	534	812	1171	588	175	629
	mean - std	259	376	250	396	273	388	449	369	41	-8	163	154	260	374	188	162	359	193	111	102
	CV	41%	28%	47%	30%	41%	35%	28%	41%	90%	105%	61%	79%	54%	49%	48%	67%	53%	51%	22%	72%
	No.	738	118	1118	590	515	1113	145	662	80	200	996	1124	1011	975	547	781	233	1023	22	1066
85	Accuracy	96		9	38	101	56		106			94	194	1	117	90	67	344	23		5
		Line 04								Line 05						Line 06					

Total absolute error	129	PTHSK																		
Cells	Cell No.	755	756	778	779	800	724	744	745	765	786	210	211	212	213	238	239	242	243	2627
	Orientation	South-East	South-East	Easth	North-East	North	West	North-East	Easth	North-East	North-East	North-West	North-West	North-West	North-West	North-West	North-West	North-East	North	North-West
	Elevation gradient	31%	31%	21%	10%	10%	11%	4%	9%	5%	3%	29%	23%	25%	5%	15%	25%	6%	5%	29%
2013	SWE Obs.	310	450	244	317	452	624	360	684	500	428	314	324	358	222	281	227	369	378	432
	SWE Model	415	452	409	405	363	361	325	335	280	250	519	446	263	248	547	552	247	236	555
	Real error		2	165	88	-88			-349	-220	-178	205	122	-95	26		325	-122	-142	122
	Minimum	212	88	101	105	105	293	360	78	78	83	11	25	70	73	93	74	73	84	78
	Maximum	454	1105	1505	2420	1170	833	360	3663	2852	1666	1122	1216	1816	731	613	543	1064	994	2610
	Std	76	211	242	356	264	181	0	610	519	321	222	249	327	135	141	94	221	203	480
	mean + std	387	660	486	672	716	805	360	1295	1019	749	537	573	685	356	422	321	590	581	912
	mean - std	234	239	2	-39	188	444	360	74	-18	107	92	75	31	87	140	132	148	175	-48
	CV	25%	47%	99%	112%	58%	29%	0%	89%	104%	75%	71%	77%	91%	61%	50%	42%	60%	54%	111%
	No.	24	1003	1027	885	462	59	1	1024	1019	958	571	897	1062	962	123	241	422	208	333
158	Accuracy		2	165	88	88			349	220	178	205	122	95	26		325	122	142	122
2014	SWE Obs.	170	326	200	170	242	286	493	386	376	394	183	234	241	173	238	175	288	219	154
	SWE Model	619	670	603	572	532	402	377	366	357	340	491	431	303	235	545	552	234	159	551
	Real error		344	404	403	289			-20	-19	-54	309	196	62	62		377	-54	-60	397
	Minimum	32	16	16	0	16	19	272	19	19	48	26	17	17	17	35	26	35	26	26
	Maximum	219	556	556	555	539	425	650	810	810	785	680	692	712	510	428	514	656	635	428
	Std	58	130	125	145	151	127	106	230	218	204	164	167	193	136	87	129	165	135	134
	mean + std	228	457	324	315	393	413	599	616	593	598	347	401	435	309	325	303	453	354	289
	mean - std	112	196	75	25	91	159	387	156	158	189	18	67	48	37	152	46	123	84	20
	CV	34%	40%	62%	86%	62%	45%	22%	60%	58%	52%	90%	71%	80%	79%	36%	74%	57%	62%	87%
	No.	14	1017	975	811	365	60	15	709	659	674	507	925	1037	887	123	351	413	264	308
153	Accuracy		344	404	403	289			20	19	54	309	196	62	62		377	54	60	397
2015	SWE Obs.	Line didn`t pass this cell	382	326	371	507	251	332	461	472	519	212	218	319	292	184	213	353	431	308
	SWE Model		595	444	401	369	379	345	345	334	320	495	452	380	359	534	548	357	337	548
	Real error		213	118	30	-139			-116	-138	-199	283	234	60	67			4		240
	Minimum		60	45	68	68	66	199	58	22	22	36	29	36	22	73	104	44	150	44
	Maximum		1234	1234	3022	1503	424	433	2743	2976	1849	773	678	1872	1284	293	471	1072	870	2607
	Std		237	242	456	360	119	73	493	527	324	123	135	272	219	51	67	200	193	395
	mean + std		619	568	827	867	371	406	954	999	843	335	353	592	511	235	279	553	624	703
	mean - std		145	84	-85	147	132	259	-32	-55	195	88	83	47	72	134	146	153	238	-87
	CV		62%	74%	123%	71%	48%	22%	107%	112%	62%	58%	62%	85%	75%	27%	31%	57%	45%	128%
	No.		983	1069	814	482	64	9	1006	1022	955	652	903	1084	943	67	91	442	161	427
139	Accuracy		213	118	30	139			116	138	199	283	234	60	67			4		240
2016	SWE Obs.	289	575	310	419	393	466	518	528	204	422	422	397	434	369	Line didn`t pass this cell	317	412	407	393
	SWE Model	431	470	425	425	400	428	402	389	386	373	549	500	418	372		576	362	314	583
	Real error		-104	115	5	7		-116	-138		-49	127	103	-16	2		259	-50		190
	Minimum	63	24	32	0	111	313	70	70	150	79	180	136	119	171		215	223	17	223
	Maximum	693	1430	1270	3058	904	501	2002	2553	360	1396	1785	1094	1897	1176		1186	884	825	1238
	Std	105	273	235	512	191	47	356	460	49	298	244	202	292	138		130	124	185	221
	mean + std	394	848	545	931	583	513	874	988	252	720	666	600	727	507		447	537	592	614
	mean - std	184	301	75	-93	202	419	162	67	155	124	178	195	142	231		188	288	222	173
	CV	36%	48%	76%	122%	49%	10%	69%	87%	24%	71%	58%	51%	67%	37%		41%	30%	45%	56%
	No.	136	952	1050	831	469	67	1033	974	55	961	556	914	1068	1107		329	395	179	303
112	Accuracy		104	115	5	7		116	138		49	127	103	16	2		259	50		190
2017	SWE Obs.	500	378	383	419	509	688	618	475	231	398	552	455	405	358	Line didn`t pass this cell	488	365	378	590
	SWE Model	592	564	542	539	500	565	520	483	413	375	679	616	357	303		754	296	249	737
	Real error				120			-98	8		-23	126	160	-48	-54			-69		
	Minimum	5	0	5	30	45	321	59	55	63	55	198	0	45	45		198	198	198	198
	Maximum	1360	1128	1085	2182	1203	878	2689	2204	523	1361	1402	1286	1407	1151		1275	770	1072	1745
	Std	313	260	257	431	294	165	454	398	109	273	274	270	237	199		283	187	245	344
	mean + std	814	637	641	850	803	853	1071	873	339	671	826	725	642	556		771	551	623	934
	mean - std	187	118	126	-12	214	523	164	77	122	125	279	186	167	159		205	178	133	246
	CV	63%	69%	67%	103%	58%	24%	73%	84%	47%	69%	50%	59%	59%	56%		58%	51%	65%	58%
	No.	97	137	168	217	113	55	1020	972	41	979	250	455	524	474		176	215	110	102
85	Accuracy				120			98	8		23	126	160	48	54			69		
		Line 07					Line 08					Line 09								

Table Ap8.3 Summary of SWE calculation in passed cell in PTSSK method

Total absolute error	131	PTSSK																			
Cells	Cell No.	279	280	308	336	337	359	377	378	832	833	855	856	857	858	864	879	880	893	894	907
	Orientation	West	West	West	West	South-West	West	West	North-West	North-West	West	South-West	South-West	West	North-West	Easth	Easth	Easth	North	North	North-West
	Elevation gradient	11%	12%	11%	6%	7%	12%	18%	21%	7%	11%	9%	11%	11%	12%	15%	13%	15%	5%	5%	3%
2013	SWE Obs.	354	459	377	494	391	529	310	514	446	273	496	817	740	804	409	768	491	450	118	420
	SWE Model	295	287	394	462	446	507	609	590	255	299	349	402	421	458	326	288	288	259	259	263
	Real error	-59		16	-31	55	-22		76		26	-147	-415	-318	-346	-82	-481		-191		-158
	Minimum	85	203	81	121	85	117	147	181	107	98	102	102	125	116	114	105	235	105	110	105
	Maximum	1103	1041	1499	1461	915	1554	715	3015	1647	1144	2237	4160	2416	2336	796	3249	1007	1619	123	2279
	Std	207	164	273	243	205	263	112	400	449	197	437	747	453	441	162	632	198	265	3	377
	mean + std	561	623	651	737	596	792	423	914	895	470	933	1564	1192	1245	571	1400	689	715	121	797
	mean - std	147	295	104	250	186	266	198	114	-2	77	59	70	287	362	246	136	293	185	115	43
	CV	58%	36%	72%	49%	52%	50%	36%	78%	101%	72%	88%	91%	61%	55%	40%	82%	40%	59%	2%	90%
	No.	737	115	1128	587	520	1115	161	679	83	207	997	1094	1010	972	406	989	67	1030	31	1063
155	Accuracy	59		16	31	55	22		76		26	147	415	318	346	82	481		191		158
2014	SWE Obs.	364	439	327	363	313	350	334	340	309	267	389	440	473	447	358	322	422	373	135	288
	SWE Model	268	263	419	504	470	551	671	642	260	317	372	427	458	501	401	382	367	344	336	336
	Real error	-96		92	141	156	201		303			-17	-13	-15	54	43	60		-29		48
	Minimum	19	199	19	76	76	67	67	76	21	84	94	84	104	63	10	10	29	29	39	10
	Maximum	800	798	793	791	641	797	627	804	809	703	868	871	858	871	702	725	725	720	263	725
	Std	194	168	182	121	123	161	141	166	246	141	179	193	195	213	191	207	212	183	90	202
	mean + std	558	606	509	485	436	511	475	506	555	407	568	633	668	660	549	529	634	556	225	490
	mean - std	170	271	144	242	190	188	194	174	63	126	209	247	277	234	167	115	211	191	45	86
	CV	53%	38%	56%	33%	39%	46%	42%	49%	80%	53%	46%	44%	41%	48%	53%	64%	50%	49%	67%	70%
	No.	737	120	1118	469	514	735	128	479	90	182	722	956	667	925	413	684	188	849	21	952
148	Accuracy	96		92	141	156	201		303			17	13	15	54	43	60		29		48
2015	SWE Obs.	313	422	236	373	320	366	281	412	489	249	394	733	656	600	347	479	496	372	133	341
	SWE Model	331	322	411	478	456	526	636	608	304	342	367	418	437	478	349	304	304	290	289	293
	Real error	18		174	105	135	160		196		93	-27	-316	-219	-122	2	-174	-192	-82		-49
	Minimum	39	162	46	106	52	79	106	134	88	47	47	39	31	39	65	65	72	72	117	57
	Maximum	1105	1369	923	1155	1195	1411	561	2297	1618	1008	1700	4078	1537	1971	1517	1859	1698	1331	171	1646
	Std	236	254	168	221	204	226	97	320	431	178	324	759	334	336	202	375	376	228	15	305
	mean + std	549	676	405	594	524	592	378	733	920	427	718	1492	990	936	549	854	872	601	149	646
	mean - std	76	169	68	153	117	139	184	92	58	70	71	-25	322	264	145	103	120	144	118	36
	CV	76%	60%	71%	59%	64%	62%	35%	78%	88%	72%	82%	103%	51%	56%	58%	78%	76%	61%	11%	89%
	No.	718	139	1134	638	461	1109	127	672	74	203	988	1097	1002	965	392	672	332	1024	26	1058
134	Accuracy	18		174	105	135	160		196		93	27	316	219	122	2	174	192	82		49
2016	SWE Obs.	473	596	474	481	461	625	548	615	543	321	490	622	725	744	543	455	870	427	159	377
	SWE Model	328	320	406	471	447	510	610	573	317	359	385	416	431	470	365	339	339	320	319	321
	Real error	-145		-68	-10	-14	-114		-42		39	-104	-205	-293	-273	-178	-116	-531	-108		-56
	Minimum	97	345	97	187	160	160	223	124	134	100	126	92	126	67	86	51	289	43	112	26
	Maximum	1601	1346	1410	1640	1071	1775	896	2578	1448	1139	1795	1980	1636	1899	1119	1699	2072	1619	381	1965
	Std	281	216	257	203	212	356	123	374	419	192	276	432	304	362	245	375	454	251	58	334
	mean + std	754	811	731	684	673	980	672	989	962	513	766	1053	1028	1106	788	830	1324	678	218	711
	mean - std	192	380	217	278	250	269	425	241	123	129	214	190	421	382	299	80	415	177	101	44
	CV	59%	36%	54%	42%	46%	57%	23%	61%	77%	60%	56%	69%	42%	49%	45%	82%	52%	59%	36%	88%
	No.	731	132	1145	595	514	1142	139	676	72	209	985	1280	1013	973	417	822	204	1047	29	1061
125	Accuracy	145		68	10	14	114		42		39	104	205	293	273	178	116	531	108		56
2017	SWE Obs.	437	523	468	563	465	593	624	621	396	164	414	738	568	727	361	487	765	391	143	366
	SWE Model	316	299	433	516	482	568	739	666	262	343	443	501	519	599	402	355	349	322	320	323
	Real error	-121		-35	-47	17	-26		45			29	-237	-49	-127	42	-132	-416	-68		-42
	Minimum	102	353	75	271	68	60	324	349	51	42	68	38	38	38	7	100	143	100	111	104
	Maximum	1136	1059	1324	1471	983	1564	1269	2010	1124	908	1366	2529	1581	1784	961	1408	1769	1173	212	1764
	Std	178	146	218	167	191	205	175	252	354	172	251	584	308	353	173	325	406	198	32	263
	mean + std	616	669	686	730	656	799	800	872	750	336	665	1322	876	1079	534	812	1171	588	175	629
	mean - std	259	376	250	396	273	388	449	369	41	-8	163	154	260	374	188	162	359	193	111	102
	CV	41%	28%	47%	30%	41%	35%	28%	41%	90%	105%	61%	79%	54%	49%	48%	67%	53%	51%	22%	72%
	No.	738	118	1118	590	515	1113	145	662	80	200	996	1124	1011	975	547	781	233	1023	22	1066
91	Accuracy	121		35	47	17	26		45			29	237	49	127	42	132	416	68		42
		Line 04								**Line 05**						**Line 06**					

Total absolute error	131	PTSSK																		
Cells	Cell No.	755	756	778	779	800	724	744	745	765	786	210	211	212	213	238	239	242	243	2627
	Orientation	South-East	South-East	Easth	North-East	North	West	North-East	Easth	North-East	North-East	North-West	North-West	North-West	North-West	North-West	North-West	North-East	North	North-West
	Elevation gradient	31%	31%	21%	10%	10%	11%	4%	9%	5%	3%	29%	23%	25%	5%	15%	25%	6%	5%	29%
2013	SWE Obs.	310	450	244	317	452	624	360	684	500	428	314	324	358	222	281	227	369	378	432
	SWE Model	414	482	404	394	350	372	326	334	304	276	520	461	305	273	560	571	275	241	576
	Real error		32	160	77	-102			-350	-197	-152	206	137	-53	51		344	-94	-137	143
	Minimum	212	88	101	105	105	293	360	78	78	83	11	25	70	73	93	74	73	84	78
	Maximum	454	1105	1505	2420	1170	833	360	3663	2852	1666	1122	1216	1816	731	613	543	1064	994	2610
	Std	76	211	242	356	264	181	0	610	519	321	222	249	327	135	141	94	221	203	480
	mean + std	387	660	486	672	716	805	360	1295	1019	749	537	573	685	356	422	321	590	581	912
	mean - std	234	239	2	-39	188	444	360	74	-18	107	92	75	31	87	140	132	148	175	-48
	CV	25%	47%	99%	112%	58%	29%	0%	89%	104%	75%	71%	77%	91%	61%	50%	42%	60%	54%	111%
	No.	24	1003	1027	885	462	59	1	1024	1019	958	571	897	1062	962	123	241	422	208	333
155	Accuracy		32	160	77	102			350	197	152	206	137	53	51		344	94	137	143
2014	SWE Obs.	170	326	200	170	242	286	493	386	376	394	183	234	241	173	238	175	288	219	154
	SWE Model	626	683	611	562	509	383	360	345	340	310	478	403	280	229	547	551	221	187	555
	Real error		357	411	392	267			-41	-36	-84	295	169	39	56		377	-67	-32	400
	Minimum	32	16	16	0	16	19	272	19	19	48	26	17	17	17	35	26	35	26	26
	Maximum	219	556	556	555	539	425	650	810	810	785	680	692	712	510	428	514	656	635	428
	Std	58	130	125	145	151	127	106	230	218	204	164	167	193	136	87	129	165	135	134
	mean + std	228	457	324	315	393	413	599	616	593	598	347	401	435	309	325	303	453	354	289
	mean - std	112	196	75	25	91	159	387	156	158	189	18	67	48	37	152	46	123	84	20
	CV	34%	40%	62%	86%	62%	45%	22%	60%	58%	52%	90%	71%	80%	79%	36%	74%	57%	62%	87%
	No.	14	1017	975	811	365	60	15	709	659	674	507	925	1037	887	123	351	413	264	308
148	Accuracy		357	411	392	267			41	36	84	295	169	39	56		377	67	32	400
2015	SWE Obs.	Line didn`t pass this cell	382	326	371	507	251	332	461	472	519	212	218	319	292	184	213	353	431	308
	SWE Model		543	478	410	365	371	338	339	325	308	501	434	358	300	547	569	316	278	569
	Real error		161	152	39	-142			-122	-147	-211	289	215	39	8			-37		260
	Minimum		60	45	68	68	66	199	58	22	22	36	29	36	22	73	104	44	150	44
	Maximum		1234	1234	3022	1503	424	433	2743	2976	1849	773	678	1872	1284	293	471	1072	870	2607
	Std		237	242	456	360	119	73	493	527	324	123	135	272	219	51	67	200	193	395
	mean + std		619	568	827	867	371	406	954	999	843	335	353	592	511	235	279	553	624	703
	mean - std		145	84	-85	147	132	259	-32	-55	195	88	83	47	72	134	146	153	238	-87
	CV		62%	74%	123%	71%	48%	22%	107%	112%	62%	58%	62%	85%	75%	27%	31%	57%	45%	128%
	No.		983	1069	814	482	64	9	1006	1022	955	652	903	1084	943	67	91	442	161	427
134	Accuracy		161	152	39	142			122	147	211	289	215	39	8			37		260
2016	SWE Obs.	289	575	310	419	393	466	518	528	204	422	422	397	434	369	Line didn`t pass this cell	317	412	407	393
	SWE Model	436	513	429	412	386	414	380	364	357	344	546	485	381	341		603	337	289	611
	Real error		-62	119	-7	-6		-138	-163		-79	125	88	-53	-28		286	-75		217
	Minimum	63	24	32	0	111	313	70	70	150	79	180	136	119	171		215	223	17	223
	Maximum	693	1430	1270	3058	904	501	2002	2553	360	1396	1785	1094	1897	1176		1186	884	825	1238
	Std	105	273	235	512	191	47	356	460	49	298	244	202	292	138		130	124	185	221
	mean + std	394	848	545	931	583	513	874	988	252	720	666	600	727	507		447	537	592	614
	mean - std	184	301	75	-93	202	419	162	67	155	124	178	195	142	231		188	288	222	173
	CV	36%	48%	76%	122%	49%	10%	69%	87%	24%	71%	58%	51%	67%	37%		41%	30%	45%	56%
	No.	136	952	1050	831	469	67	1033	974	55	961	556	914	1068	1107		329	395	179	303
125	Accuracy		62	119	7	6		138	163		79	125	88	53	28		286	75		217
2017	SWE Obs.	500	378	383	419	509	688	618	475	231	398	552	455	405	358	Line didn`t pass this cell	488	365	378	590
	SWE Model	612	564	523	479	431	489	445	402	404	359	649	528	320	285		750	277	239	734
	Real error				60			-173	-73		-38	96	72	-84	-73			-87		
	Minimum	5	0	5	30	45	321	59	55	63	55	198	0	45	45		198	198	198	198
	Maximum	1360	1128	1085	2182	1203	878	2689	2204	523	1361	1402	1286	1407	1151		1275	770	1072	1745
	Std	313	260	257	431	294	165	454	398	109	273	274	270	237	199		283	187	245	344
	mean + std	814	637	641	850	803	853	1071	873	339	671	826	725	642	556		771	551	623	934
	mean - std	187	118	126	-12	214	523	164	77	122	125	279	186	167	159		205	178	133	246
	CV	63%	69%	67%	103%	58%	24%	73%	84%	47%	69%	50%	59%	59%	56%		58%	51%	65%	58%
	No.	97	137	168	217	113	55	1020	972	41	979	250	455	524	474		176	215	110	102
91	Accuracy				60			173	73		38	96	72	84	73			87		
		Line 07					Line 08					Line 09								

Appendix 9

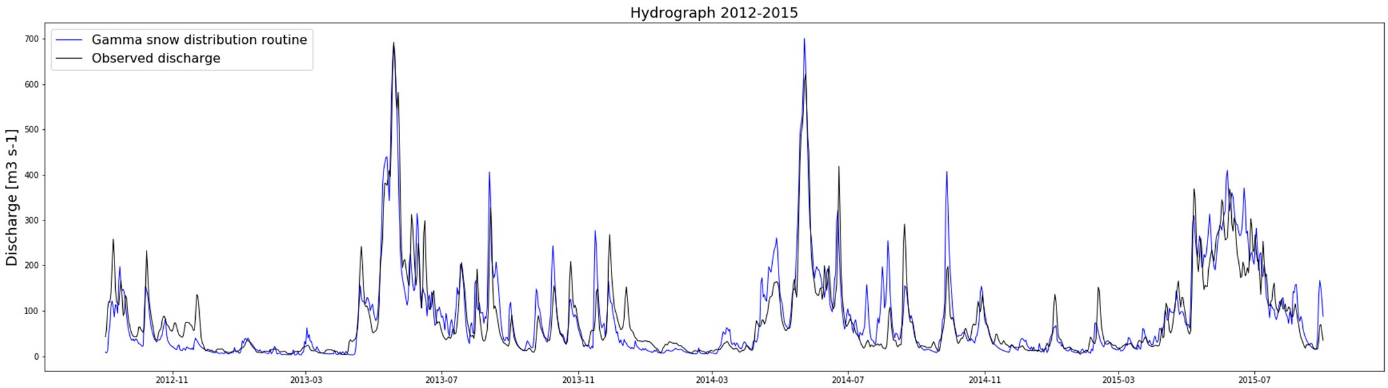
Calibrated and validated Hydrographs

Calibrated and validated Hydrographs

In this appendix the average of 36 best calibrations and validations hydrographs out of 200 tries for each method are presented. The calibrations and verifications were done for 3 years (2012-2015) and 2 years (2015-2017) respectively.

Figure Ap9.1 Observed and PTGSK simulated hydrographs

NSE (Calibration): 77%   NSE (Validation): 67%



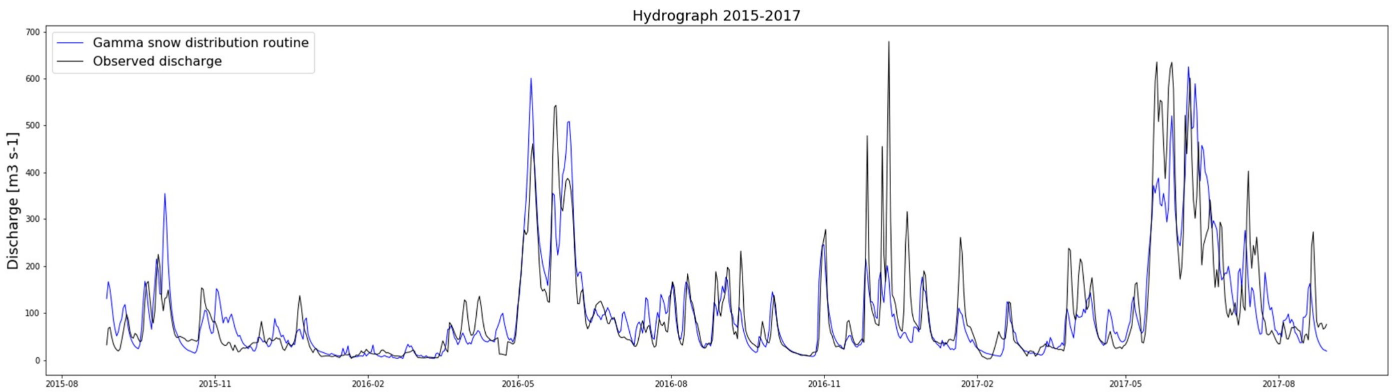
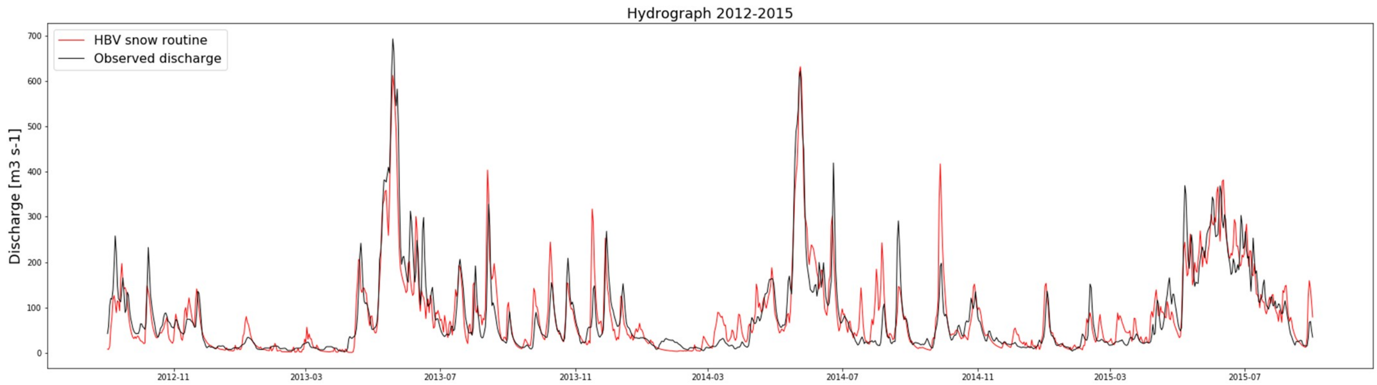


Figure Ap9.2 Observed and PTHSK simulated hydrographs

NSE (Calibration): 77% NSE (Validation): 81%



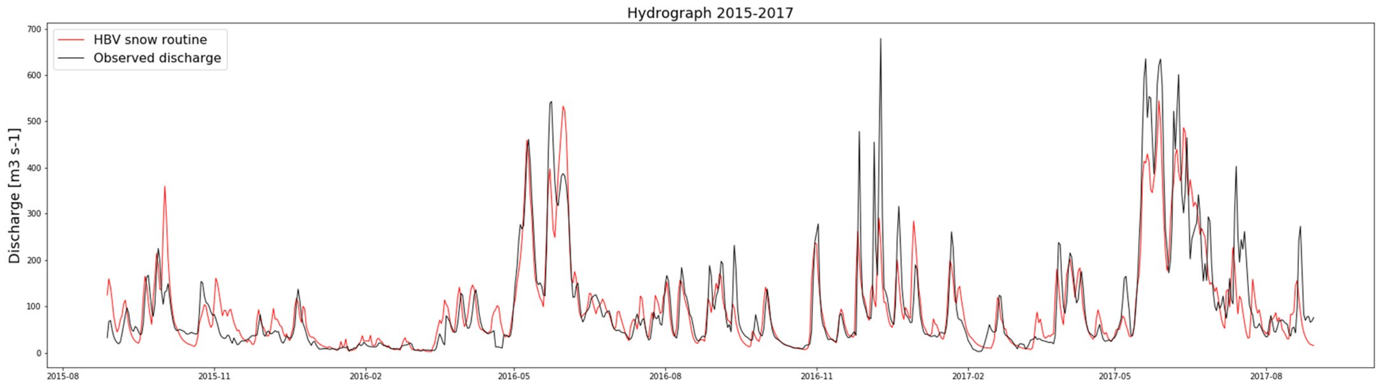
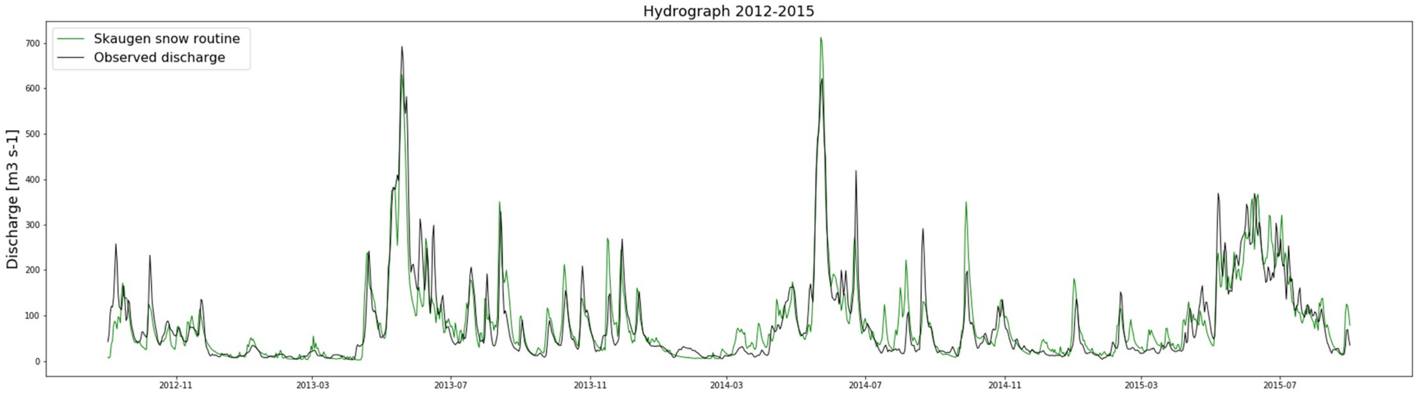
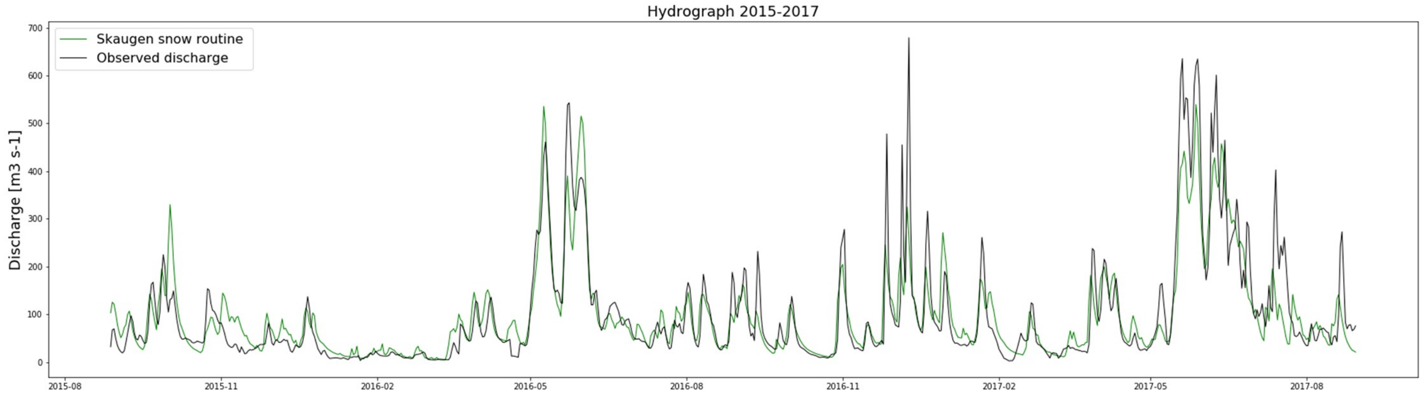


Figure Ap9.3 Observed and PTSSK simulated hydrographs

NSE (Calibration): 77% NSE (Validation): 79%





Appendix 10

Graphs code in Seaborn

(Python)

Graphs code in seaborn (Python)

```
import seaborn as sb
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

file_name = r'D:\Dropbox\Thesis\SHyFT\seaborn.csv'
my_data = pd.DataFrame()
my_data =pd.read_csv(file_name)
my_data.index= my_data['Cell No.'][:]
del my_data['Cell No.']


with sb.axes_style("white"):
    sb.jointplot(x=np.array(my_data1['Elevation'][:]),
y=np.array(my_data1['Accuracy'][:]), kind="hex", color="k");


sb.set(style="darkgrid", color_codes=True)
plt.figure(figsize = (13,9))
tips = sb.load_dataset("tips")
g = sb.jointplot("Number of points", "Accuracy", data=my_data,
kind="reg", xlim=(-10, 1400), ylim=(-0.1, 1.1), color="r", size=7)


plt.figure(figsize = (13,9))
sb.swarmplot(x = "range", y='Accuracy', data = my_data , size = 7,hue
= "snow_course", edgecolor='gray')
sb.boxplot(x = "range", y='Accuracy', data = my_data , whis=np.inf)
plt.grid()
plt.show()


plt.figure(figsize = (13,9))
sb.swarmplot(x = "range", y='Accuracy', data = my_data , size = 7,hue
= "snow_course", edgecolor='gray')
plt.grid()
plt.show()


plt.figure(figsize = (13,9))
sb.swarmplot(x = "snow_course", y='Accuracy', data = my_data , size =
7,hue = "range", edgecolor='gray')
plt.legend(loc = 0)
sb.boxplot(x = "snow_course", y='Accuracy', data = my_data ,
whis=np.inf)
plt.grid()
plt.show()


plt.figure(figsize = (13,9))
sb.swarmplot(x = "snow_course", y='Accuracy', data = my_data , size =
7,hue = "range", edgecolor='gray')
```

```
sb.violinplot(x = "snow_course", y='Accuracy', data = my_data,
inner=None)
plt.grid()
plt.show()


fig, ax = plt.subplots(figsize=(15,8))
sb.set(style="ticks", palette="pastel")
sb.stripplot(x = "snow_course", y='Accuracy', hue = 'Year', data =
my_data , size = 8)
ax.legend(loc = 2)
plt.savefig("snow_course_accuracy_year2.png")
plt.show()


fig, ax = plt.subplots(figsize=(20,10))
sb.stripplot(x = 'Elevation', y='Accuracy',hue = 'Year', data =
my_data , size = 6)
plt.savefig("elevation_accuracy_year.png")
plt.show()


plt.figure(figsize = (13,8))
# do not overlap on them, hue make a legend and shows the"Company"
with color
sb.swarmplot(x = 'range', y='Accuracy', data = my_data , size = 10,
hue = "snow_course", vmin = 0, vmax =10)
# plt.grid()
plt.savefig("Range_accuracy_snow.png")
# plt.xticks([1,20,30,40,50])
plt.show()

fig, ax = plt.subplots(figsize=(15,8))
sb.boxplot(x = 'range', y='Accuracy',hue="Year", data = my_data )
# sb.despine(offset=6, trim=False)
plt.savefig("Range_accuracy_year.png")
# sns.boxplot(x="Cells", y="SWE [mm]",hue="Method", palette=["m", "g",
"b", "r"],data=line42013)
plt.show()
```

Appendix 11

miscellaneous graphs

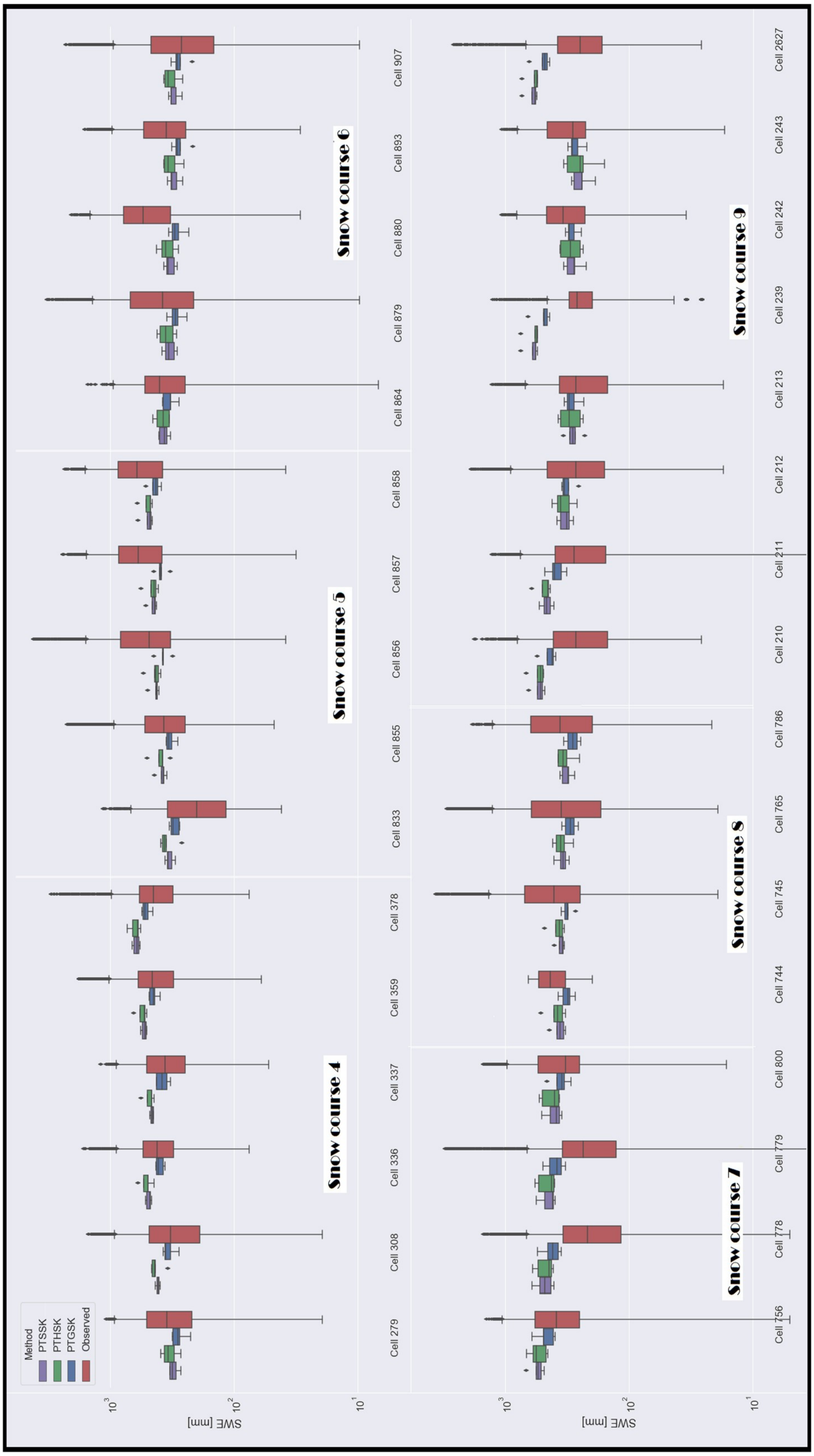


Figure Ap11.1 Logarithmic SWE axis in different cells

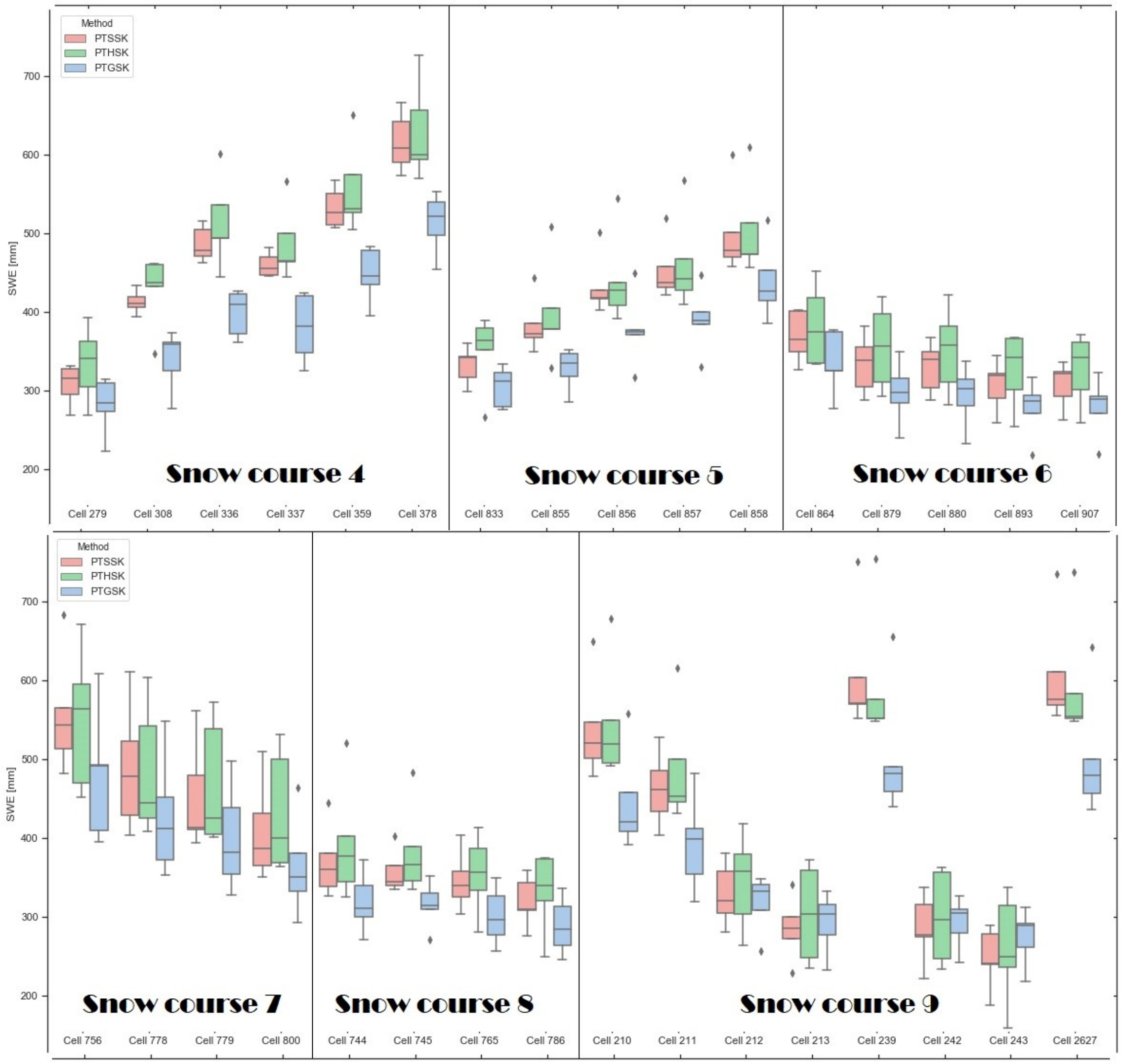


Figure AP11.2 SWE boxplot of different cells

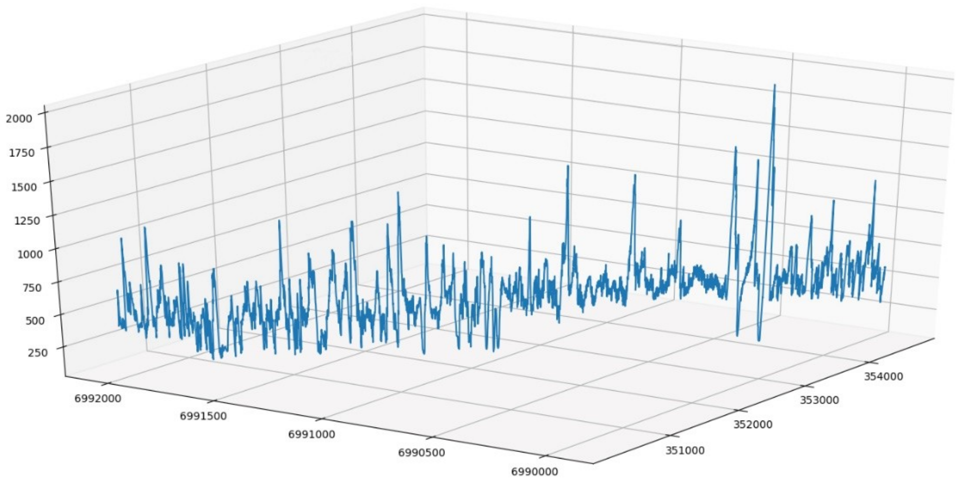


Figure AP11.3 An example of a SWE depth profile

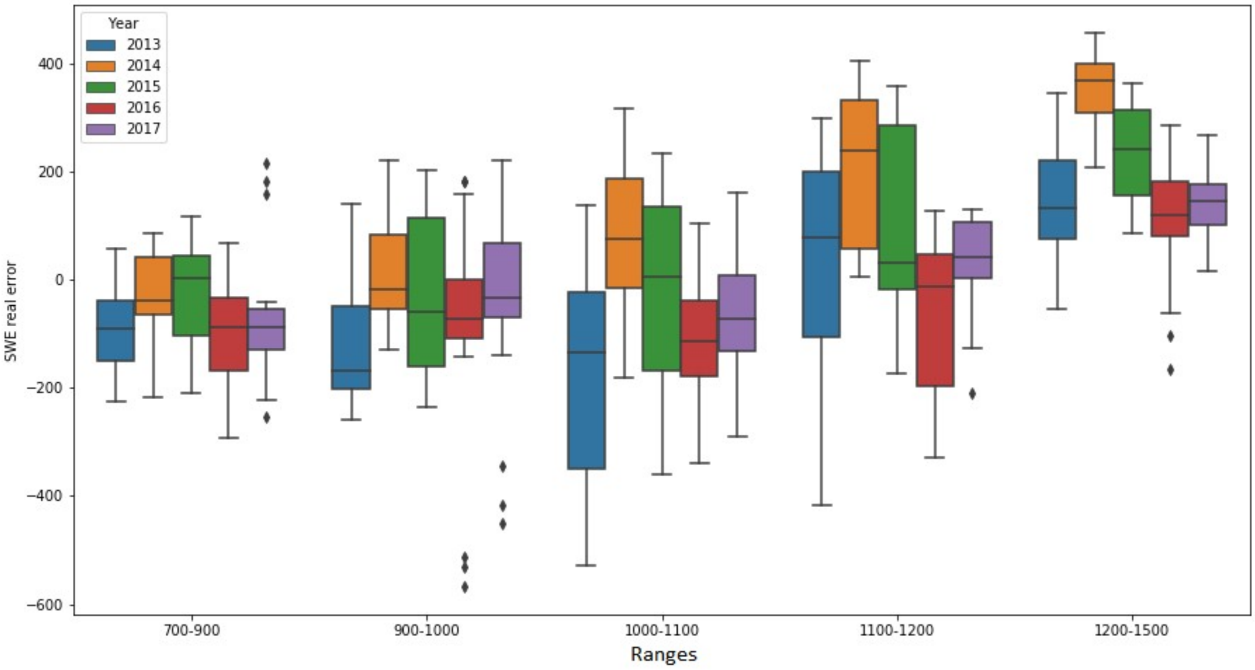


Figure AP11.4 SWE real error boxplot against ranges 1

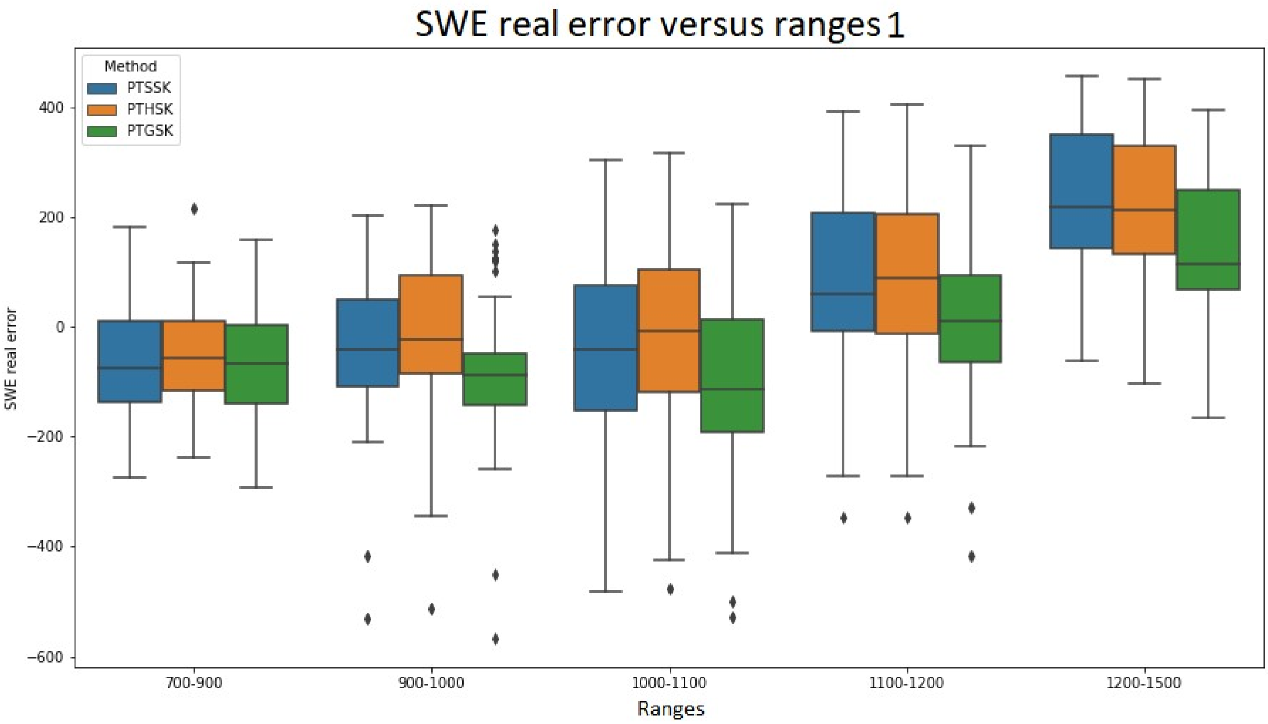


Figure AP11.5 SWE real error boxplot against ranges 2

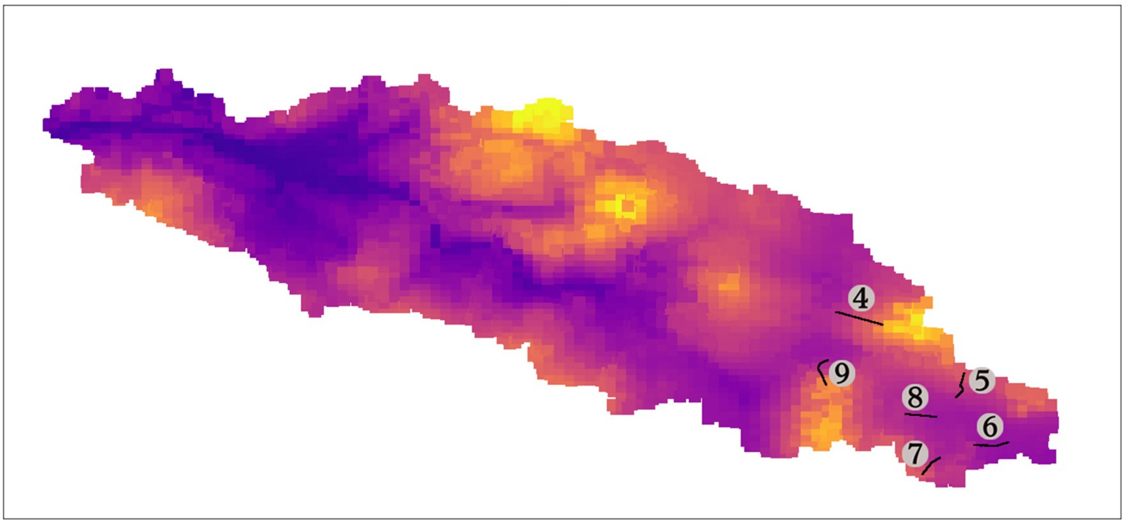


Figure AP11.6 Snow course on the catchment layout

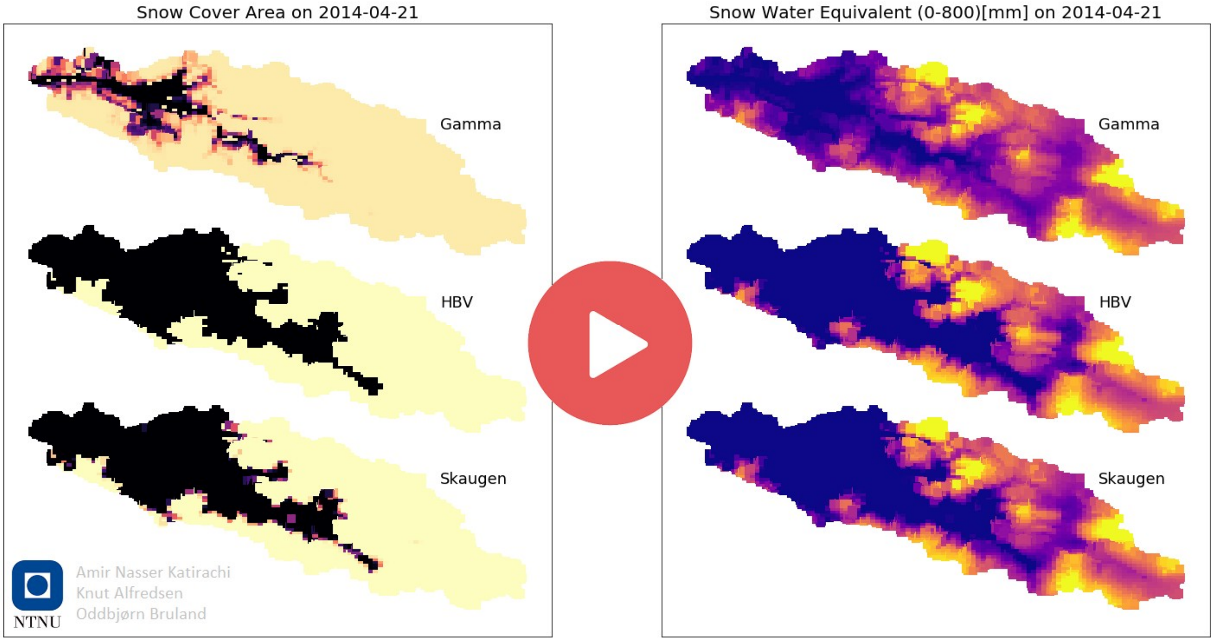
Appendix 12

YouTube movie

YouTube video

It shows 2 years simulation with three methods in 2 minutes

https://youtu.be/HeLNBz_tszo



Appendix 13

Satellite images

Figure Ap13.1 A typical satellite image during snow season

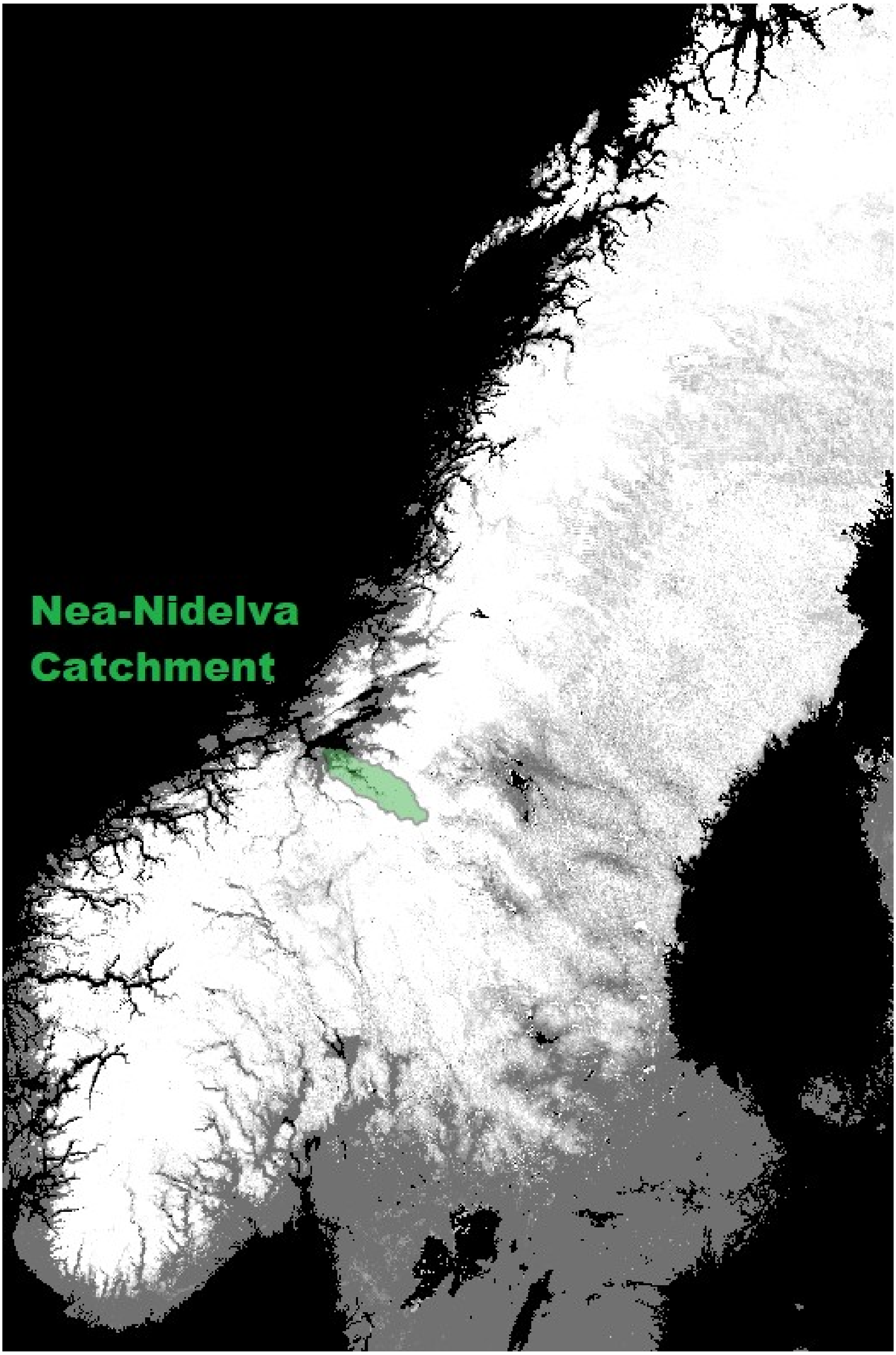
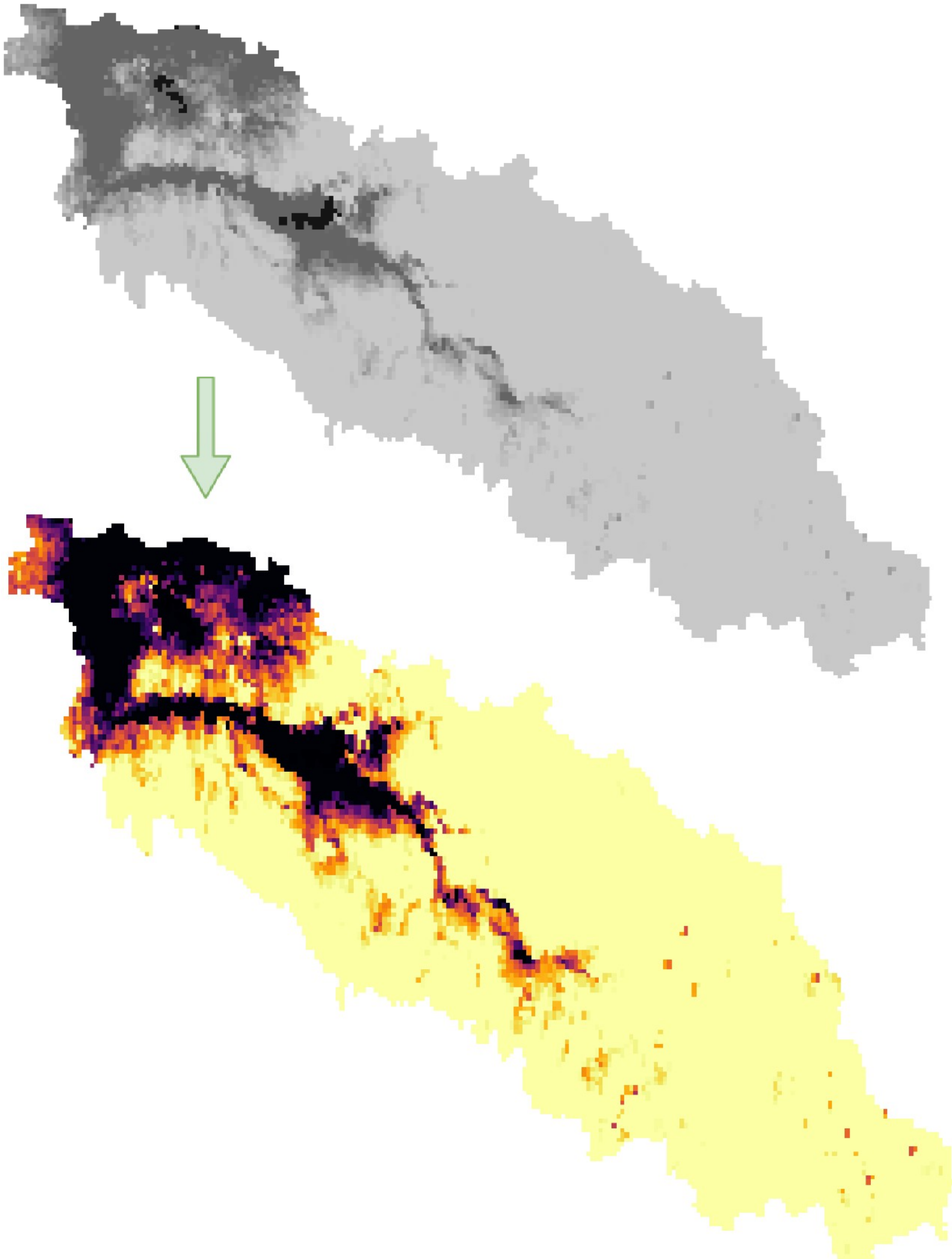


Figure Ap13.2 Changing the image Style in Qgis



The same color pallet which was used in SHyFT for SCA image for better comparison (a singleband gray to a singleband pseudocolor)