Geog0111 Scientific Computing

Coursework (Assessed Practical) Part B
Instructions and marking grids

M. Mokros
(m.mokros@ucl.ac.uk)

P. Lewis

23rd September 2025

URL: https://github.com/UCL-EO/geog0111

1. Introduction

1.1 Task overview

The coursework for Geog0111 Scientific Computing consists of two parts (Part A and Part B). The course is assessed entirely using these two submissions. This document describes the requirements for Part B (50% of the total marks), which is due for submission on the first Monday after the start of Term 2. Part A covers data preparation and presentation, and part B covers environmental modelling (using the data from part A and other geospatial data).

In this task, you will be writing codes to do two tasks: (i) **snow data generation**. Develop a *function* to generate a gap-filled daily snow cover dataset for the Del Norte catchment in Colorado, USA for the years 2018 and 2019 using MODIS data and streamflow and temperature datasets you prepared in part A. Then, separately, demonstrate the *running* of the function and *plot* the datasets alongside one another: (ii) **snowmelt model calibration and validation**. Develop a *function* to **calibrate** the parameters of a snowmelt model driven by snow cover and temperature using observations of streamflow for one year of data for Del Norte. Develop *another function* to **validate** the model with these parameters against an independent year of data for the same area. Then demonstrate the *running* of these functions and *visualise* the results.

The main coding exercises involve building a set of Python functions, then running these, passing data between the functions, and visualising results. a set of You must provide and run the functions that you should develop in a Jupyter notebook, as well as showing results in the notebook.

1.2 Submission

The due dates for the two formally assessed pieces of coursework are:

- Part A (this piece of work): **10 November 2025** (50% of final mark) - first Monday after reading week.
- Part B (the next piece of work): **12 January 2026** (50% of final mark) - first day of term 2

Submission is through the usual Turnitin link on the course Moodle page.

You must develop and run the codes in a **single Jupyter notebook**, and submit the work in a single notebook as a PDF file. As usual with coursework, you must **attach a cover page** declaration.

2. Background

2.1 Model background

The hydrology of the Rio Grande Headwaters in Colorado, USA is snowmelt dominated. It varies considerably from year to year and may very further under a changing climate. One of the tools we use to understand monitor processes in such an area is a mathematical ('environmental') model describing the main physical processes affecting hydrology in the catchment. Such a model could help understand current behaviour and allow some prediction about possible future scenarios.

In this part of your assessment you will be using, calibrating and validating such a model that relates temperature and snow cover in the catchment to river flow.

We will use the model to describe the streamflow at the Del Norte measurement station, just on the edge of the catchment. You will use environmental (temperature) data and snow cover observations to drive the model. You will perform calibration and testing by comparing model output with observed streamflow data.

Figure 1. Del Norte station

2.2.Del Norte

Further general information is available from various websites. You can visualise the site Del Norte 2E here. This is the site we will be using for river discharge data.

Figure 2. River near Del Norte station

2.3.1 Model basics

We will build a simple **mass balance model** that is capable of predicting daily streamflow at some catchment location for given temperature and catchment snow cover. This defines the *purpose* of our model (how we will use it) and describes the *model output* (daily streamflow $Q_{dN}[t]$ at some catchment location) and drivers (temperature $T[t]$ and catchment snow cover $p[t]$):

We will need 2 dynamic datasets to *run* the model, given for samples in time t:

- $T[t]$: mean temperature (C) at the Del Norte monitoring station for each day of the year
- $p[t]$: Catchment snow cover (proportion)

and one dataset to *calibrate* and *test* the model, for samples in time:

- $Q_{dN}[t]$: stream flow data for each in units of megalitres/day (ML/day i.e. units of 1000000 litres a day) at the del Norte monitoring station.

You should already have the datasets $T[t]$ and $Q_{dN}[t]$ for the years 2016-2019 inclusive, and will need to make use of the datasets for 2018 and 2019 in this work. In the first part of this submission we will deriving the snow cover data from MODIS satellite data. We will explain this below.

As we have noted, you will be running, calibrating and testing a mass-balance snowmelt model in the Rio Grande Headwaters in Colorado, USA. In such a model, we keep track of the mass of some parameter of interest, here the snow water-equivalent (SWE), the amount of water in the snowpack for the catchment. We assume the reservoir of water is directly proportional to snow cover $p[t]$ at time t so:

$$SWE[t] = k_1 \, p[t] \tag{1}$$

with k_1 a constant of proportionality relating to *snow depth and density*.

In the model, *we do not consider* mechanisms of when the snow appears or disappears from any location or thinning/thickening of the snowpack. Rather, we use the snow cover as time t, $p[t]$, as a direct surrogate of the SWE at time t.

2.3.2 Water release

We need a mechanism in the model that converts from $SWE[t]$ to flow $dQ[t]$, the water flowing *into the system* from the snowpack at time t. We can consider this as:

$$dQ[t] = k_2 \, m[t] \, SWE[t]$$

$$= k_1 \, k_2 \, m[t] \, p[t]$$

$$= k \, m[t] \, p[t] \tag{2}$$

where $dQ[t]$ is the amount of water flowing *into the catchment* at time t, $m[t]$ is a proportion of the snowpack assumed to melt at time t, and k_2 a constant of proportionality relating to the proportion of the SWE that is available to be converted. We combine the constants of proportionality so $k = k_1 \, k_2$.

You can imagine the snow water equivalent $SWE[t]$ as a quantity of snow in a bucket observed at time t, and there being some proportion k_2 of this that has the potential to melt and become water $dQ[t]$. You can think of there being a tap or valve that releases that water into another bucket at time t, with $m[t]$ (between 0 and 1) being a measure of how much that valve is open or closed. The main control on this tap is temperature at time t, $T[t]$. In the simplest form, this would be a switch, so setting m to 0 when flow is off (too cold to melt) and m to 1 when flow is on (hot enough to melt). Practically, we model the rate of release of water from the snowpack as a logistic function of temperature:

$$m[t] = \text{expit}([T[t]-T_0]/x_p) \tag{3}$$

where expit the logistic function that we have previously used in phenology modelling. This is a form of 'soft' switch between the two states (still with melting, $m=0$, and melting $m=1$). If the temperature is very much less than the threshold T_0, it will remain frozen. If it is very much greater than T_0, there will be an amount proportionate to $SWE[t]$ flowing into the system on day t.
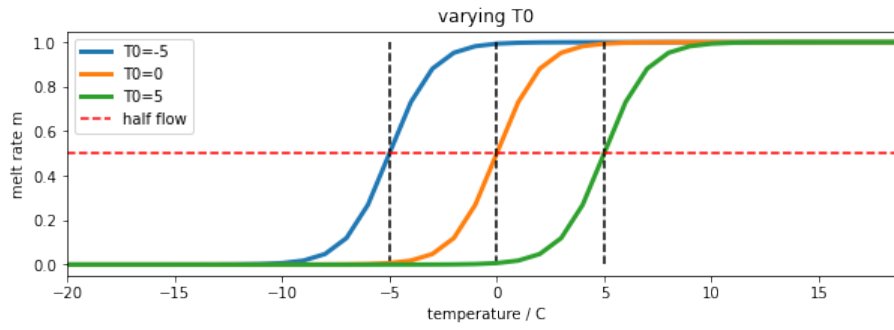
Figure 3. melt rate `m[t]` for varying value of parameter T_0

The parameter, x_p (C) increases the slope of the function at $T=T_0$ with increasing x_p. So it can be used to modify the 'speed' of action, or 'sensitivity' of the soft switch. We will use a default value of $x_p=1$. It is likely to have only a minor impact on the modelling results so we can use this assumed value of the parameter. By keeping this parameter at a fixed value, we can simplify the problem you need to solve to one involving a single parameter to model the water release. We assume that T_0 can be calibrated for a given catchment.
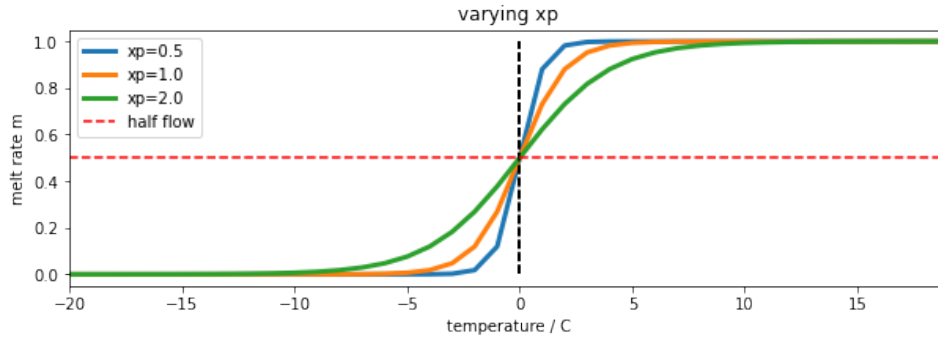

Figure 4. melt rate `m[t]` for varying value of parameter x_p for $T_0=0$

Notice that the threshold temperature `T0` is **_not_** really the temperature at which melting **_begins_**: we can see from above that for `xp=1.0`, melting occurs at temperatures above around `T0-5C`. Also, consider that the temperature data we will be using is the temperature at the Del Norte station, and that this is likely some degrees higher than the temperature in the mountains where the snowmelt will be occurring. For these various reasons, we would not expect `T0` to be `0C`, but somewhat higher, perhaps closer to `10 C`. We should call this parameter a threshold temperature, rather than a 'melting temperature'. Notice that if we decrease the sensitivity parameter x_p, then the range of 'melting temperatures' is also reduced, so we might expect T_0 to decrease with decreasing x_p. So, although changing x_p might have only a small impact on the result, it is likely to have a correlation with the parameter T_0.

So:

$$dQ[t] = k\ p[t]\ \text{expit}((T[t]-T_0)/x_p) \tag{4}$$

This is driven by `T[t]` and `p[t]` and controlled by parameters T_0 and to a lesser extent x_p. If we normalise our measures, i.e. `dQ[t]/dQmax` and `QdN[t]/QdNmax`, with `dQmax` and `QdNmax`, being the maximum values of `Q[t]` and `QdN[t]` respectively, then we can compare the quantity of melt water at time `t` with the measured flow at the monitoring station.

In the figure below, we see the normalised modelled melt water `dQ[t]` (scaled) that corresponds to a T_0 of 9 C derived from datasets `T[t]` and `p[t]`, alongside the normalised measured flow `QdN[t]` (scaled) at the del Norte station. `dQ[t]` is remarkably similar to the flow data `QdN[t]`, but much noisier. We also see that it occurs some time before we see the water flow at the monitoring station. The reason for this is that there is a 'network delay' between the melt happening in the snowpack and it reaching the monitoring station. This final component of our model is a network response function (NRF) that models this delay in the routing of the melt water.
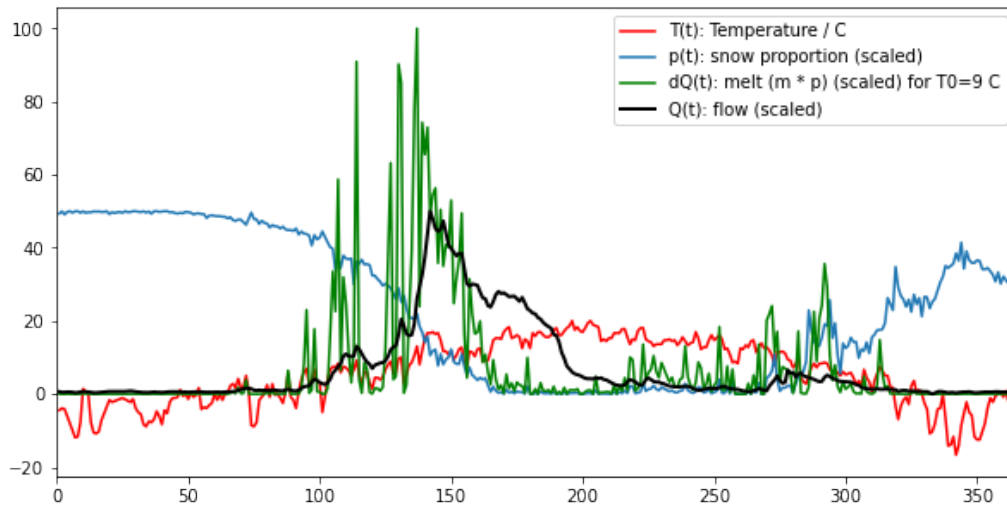
Figure 5. Available water `dQ[t]` for `T₀=9 C` (**green**) plotted alongside measured flow (**black**) and driving data

2.3.3 Flow delay to the measuring stations

To be able to generate our desired model output, we now have to consider how this reservoir of water is transported to predict daily streamflow at some catchment measurement location. We do this using the concept of a Network Response Function (NRF). In this sub-model, we assume that the flow from out reservoir `dQ[t]` to the catchment measurement location can be characterised as a decay function after time `t=0`. So, some proportion of the water released is immediately transported to the station, and a lesser amount reaches the next day, and less the next day and so on. So, if we put a *pulse* of water into the system we would measure a simple decay function. This pulse response is the NRF.
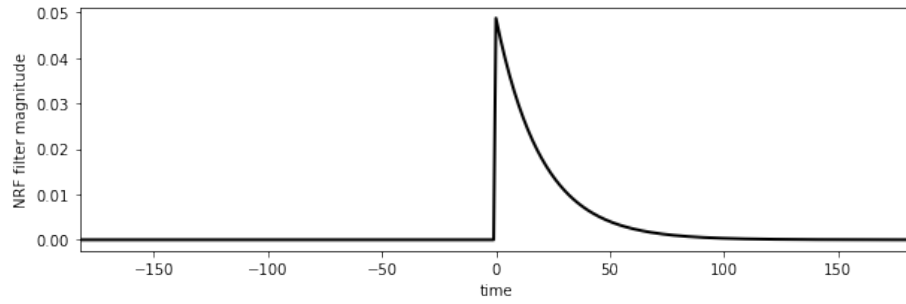

Figure 6. Network Response Function (NRF) as a function of time to `f=20` days

The NRF is effectively a one-sided smoothing filter. It imparts a delay on the signal `dQ[t]` and smooths it. We can use a function such as a one-sided Laplace distribution, a one-sided exponential parameterised by a rate of decay value `f` (days) to model the probability of water reaching the monitoring station at time `t`.

$$Q'_{dN}[t] = dQ[t] * L^1(t/f) \tag{5}$$

Where `Q'`$_{dN}$`[t]` is the *modelled* flow (we use `'` for *modelled* flow here) output at the del Norte station, `*` is the convolution operator, `dQ[t]` is the snowmelt entering the reservoir at time `t`, and `L`^{1}`(t/f)` is the NRF response, a one-sided Laplace (exponential) distribution:

$$L^1(x) = \begin{array}{ll} \exp[-x]/N & | \quad x >= 0 \\ 0 & | \end{array} \tag{6}$$

with the normalisation factor:
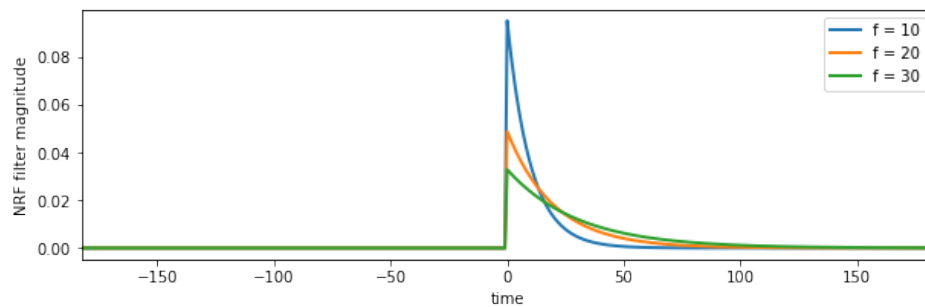
$$N = \Sigma \; L^1(x)$$

Figure 7. Network Response Function (NRF) as a function of time for varying f

You can think of this model as allowing the meltwater that becomes available at time t, $dQ[t]$, to be spread out over time when it reaches the monitoring station. So, in figure 7 for $f=10$, around 0.09 of the melt water goes immediately to the monitoring station, then less than that the next day, and so on, until almost all of the water has reached the station after around 40 days. The 'spread' of this then is controlled by the parameter f. If f is increased to 20, only 0.05 of the water goes directly to the station, and it is spread out over a longer time period. So, f is characteristic of the catchment physical properties and the location of the snow sources relative to the monitoring station. We assume it to be a constant here, something that can be calibrated for the catchment and station.

2.3.4 Model

If, for the moment, we ignore the parameter x_p, then we can illustrate out model as in the figure below:
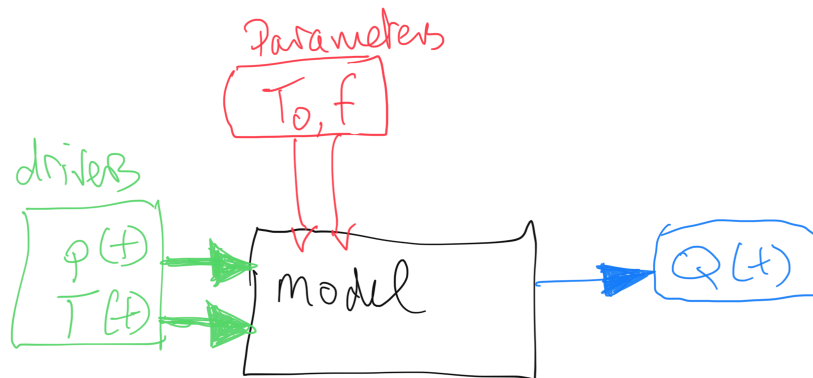

Figure 8. Model description

The complete model has three parameters that control model behaviour:

- the threshold temperature T_0 (C);
- the delay parameter f (days) of the Network Response Function (NRF)
- the temperature sensitivity parameter x_p (C)

Summary:

In this coursework, you must estimate the two main parameters (T_0 and f) for the catchment in a model calibration stage using data from 2018, and then validate the calibrated model against independent data from the year 2019. This requires you to compare the predicted station streamflow $Q'_{dN}[t]$ with measured values $Q_{dN}[t]$ in some optimisation routine using a model driven by daily temperature $T[t]$ and snow cover $p[t]$. You will first need to derive the estimate of snow cover for the two years.

You can find code implementing and running the model in `notebooks/066_Part2_code.ipynb` **in the notes.**

3. Coursework Detail

3.1 Basic requirements for this submission

In Part A of the assessment (that you have already completed part), you generated and visualised environmental datasets for daily temperature `T[t]` (C) and measured stream flow Q_{dN}`[t]` (ML/day) that we will be using in this part.

These should broadly look like the data presented in Figure 5 above (for the year 2005). If you believe you have made a significant error in generating these data and you don't have anything useable for this part, you should contact the course tutor to obtain alternative datasets. If you use these alternative data in your submission, *you must indicate that you have done so*.

3.2 Required components

- Snow data preparation [40%]
- Model inversion [60%]

Each of these parts has multiple components.

3.2.1 Snow data preparation

The aim of this part of the work is for you to produce datasets of snow cover for the Hydrological Unit Code (HUC code) catchment 13010001 (Rio Grande headwaters in Colorado, USA) using MODIS snow cover data. You should by now have plenty of experience of accessing and using the MODIS LAI product, and we have already come across the snow product in 030_NASA_MODIS_Earthdata.

You must provide:

- **A function** with **argument** `year` (integer) that **returns** a Pandas dataframe or dictionary containing keys for *day of year* (`doy`) and daily *catchment mean* snow cover `p[doy]` for HUC catchment 13010001.
    - When you run it for `year`, it must return a **gap-filled** measured snow cover dataset for that year, for each day of the year, averaged (mean) over HUC catchment 13010001.
        - You **must** do the gap-filling for each pixel in the catchment along the time-axis. You should then take the mean of the gap-filled data over the catchment.
    - The function <u>must be capable of deriving this information from appropriate MODIS snow cover datasets</u>.
    - You can *optionally* also cache this information in a file, and read the data from that file. If you use a cache, you **must** specify a keyword to switch on- or off- the use of the cached file[1].
    - **Do not** make use of global variables to pass information to a function: you must pass all information required by a function via arguments and/or keywords.
    - This gives **30 marks** in total and is *judged using the generic rubrik for functions* provided.

- **Outside of this function**, in a notebook cell, demonstrate the running of this function for years 2018 and 2019.
    - You **must** demonstrate the running of this function (with and without caching, if you implement that) in a notebook cell and plot graphs showing the snow cover datasets you have generated.
    - This gives **10 marks** (30+10 = 40% for this section)

3.2.2 Model inversion

<u>You should now have access to datasets for</u>

- `T` : mean temperature (`C`) at the Del Norte monitoring station for each day of the year, <u>read from a csv file</u>.
- `Q` : stream flow data (ML/day) for each day of the year, <u>read from a csv file</u>.
- `p` : Catchment snow cover (proportion) *returned* by the function developed above.

for the years 2018 and 2019. You should have access to an implementation of the model in Python in the code `notebooks/geog0111/model.py`. **You should import this code for use in your coursework**. You do not need to develop it yourself.

---

[1] So, if you want to use a cache file, in the function, you might test to see if the cache file exists, and if so, read data from that. If not, or if you set the keyword to ignore the cache file, then you derive `p[t]` from the MODIS data. It is not compulsory to use a cache file. But if you don't then this function **must** derive `p[t]` from the MODIS data.

If, for any reason you have been able to produce these, discuss the matter with your course tutor before completing this section. You must NOT simply use any datasets provided with the notes without consultation. If you do use the datasets with permission, you must acknowledge this in your submission (not doing so constitutes plagiarism).

You will need to use a LUT inversion to provide a calibration and validation of the model described above. You should be familiar with this approach from the material covered in the course and should use `numpy` to implement it.

You must provide:

- **A calibration function** with **argument** `year` (integer) that **returns** a dictionary or Pandas df containing the calibrated model parameters and the goodness of fit metric at the LUT minimum in calibration, along with appropriate datasets that you can use to visualise and verify the calibration results.
    - It should use a LUT approach to calibrate the 2-parameter snowmelt model presented above for `year`.
    - It should read the datasets `T` and `Q` from their CSV files for `year`, performing any necessary filtering or gap-filling (e.g. replace `NaN` values).
    - It should get the snow cover dataset `p` for `year` from the function developed for snow data preparation above.
    - **Do not** make use of global variables to pass information to a function: you must pass all information required by a function via arguments and/or keywords.
    - This gives **35 marks**.

- **A validation function** with **arguments** `year` (integer) <u>and</u> the `output` of the calibration function that **returns** a dictionary or Pandas df containing the goodness of fit metric achieved in validation and other appropriate datasets that you can use to visualise the validation results.
    - It should read the datasets `T` and `Q` from their CSV files for `year`, performing any necessary filtering or gap-filling (e.g. replace `NaN` values).
    - It should get the snow cover dataset `p` for `year` from the function developed for snow data preparation above.
    - It should compare the model-predicted and measured values of `Q` and provide appropriate summary statistics of the goodness of fit for the validation.
    - **Do not** make use of global variables to pass information to a function: you must pass all information required by a function via arguments and/or keywords.
    - This gives **10 marks**.

- **Outside of this function**, in a notebook cell, you must demonstrate the running of these functions for the years 2018 and 2019, using one as calibration and the other validation.
    - You **must** provide appropriate visualization of the measured and modelled data sets.
    - You **must** present the model parameters derived from the calibration.
    - You **must** present summary statistics (goodness of fit) for both the calibration and validation.
    - You **must** present a short paragraph of text describing the calibration and validation results.
    - *Optionally*, you might also illustrate the LUT operation.
    - This gives **15 marks** (35+10+15 = 60%)

3.2.3 Advice on development of the snow cover dataset

You will want to use the MODIS product MOD10A1 for the snow data for the years 2018 and 2019, though you might additionally look into the use of MYD10A1. You should apply the catchment boundary vector dataset you will find in the file `data/Hydrologic_Units/HUC_Polygons.shp` to clip your region of interest, specifying the warp arguments and other parameters as follows:

```
sds      = ['NDSI_Snow_Cover']
   product = 'MOD10A1'
   tile = ['h09v05']

   warp_args = {
      'dstNodata'     : 255,
      'format'        : 'MEM',
      'cropToCutline' : True,
      'cutlineWhere'  : f"HUC=13010001",
      'cutlineDSName' : 'data/Hydrologic_Units/HUC_Polygons.shp'
   }
```

You *can* the use a high-level function such as *modisAnnual* as previously in the notes to gather the dataset for the two years of interest, **but** that if you do, the marks you get <u>for that part of the code development</u> will be limited to a pass grade. You can get more marks by developing, documenting and commenting your own codes for the MODIS processing.

That said, the first time you go through the coursework, we recommend you do use a high-level function such as this to make sure that you can complete this part of the coursework. If can later return and develop your own code, in which case you will have a previous result you can check yours against. You will get more credit for codes that show more coding skills that just calling these high level codes. You should only submit one version of the codes you develop.

You will use one year for model calibration and the other for model validation. As the dataset may have gaps, you might use the regularisation approach we have taken previously for LAI to fill the gaps. The dataset you produce for each year should have 365 days of snow cover data (or 366 for leap years). It should broadly look like the example shown above. If your dataset appears too 'spikey', it may well be that you have forgotten to apply the regularisation/gap filling.

You might find additional visualisations of interest, such as the space-time plots we did for the LAI time series visualisations.

4. Coursework Submission

The due date for Part B (this piece of work) is **12 January 2026** (first Monday of term). Part B represents 50% of final mark for the course. Submission is through the usual Turnitin link on the [course Moodle page](#).

You must develop and run the codes in a single Jupyter notebook, and submit the work in a single notebook as a PDF file.

You must work individually on these tasks. If you do not, it will be treated as plagiarism. By reading [these instructions](#) for this exercise, we assume that you are aware of the UCL rules on plagiarism. You can find more information on this matter in your student handbook. If in doubt about what might constitute plagiarism, ask the course convener.