PHY407 Lab03

(Yonatan Eyob Q1,Q2), (Kivanc Aykac Q1,Q3)
Student Numbers: 1004253309, 1004326222

October 2^{nd}, 2020

1 More on integrating functions

1.a

1.a.i

In this part, the goal is to evaluate the Dawson function (1) with using the methods of Trapezoidal Rule, Simpson's Rule and Gaussian Quadrature for a range of N slices/sample points between N=8 and N=2048 (for $N = 2^n$, where $n = 3, 4, 5, 6, 7, 8, 9, 10, 11$). (The code is resembling a lot of key points with the code our group had submitted last week for Question 2.a.)

$$D(x) = e^{-x^2} \int_0^x e^{t^2} dt \tag{1}$$

Below are the table for the printed outputs of the calculations:

Number of slices	Trapezoidal Rule	Simpson's Rule	Gaussian Quadrature
8	0.2622478205347951	0.18269096459712164	0.1290106788170698
16	0.16828681895583716	0.13696648509618445	0.12934800119977766
32	0.1395800909267732	0.13001118158375188	0.1293480012360034
64	0.13194038496790617	0.12939381631495048	0.12934800123600465
128	0.1299983024925397	0.12935094166741753	0.12934800123600457
256	0.12951071531441982	0.12934818625504665	0.1293480012360041
512	0.12938868844305074	0.12934801281926103	0.1293480012360052
1024	0.1293581735809614	0.12934800196026489	0.12934800123600546
2048	0.1293505443561975	0.12934800128127608	0.12934800123600487

As can be seen, Gaussian Quadrature method reaches steadiness faster then the other methods.

1.a.ii

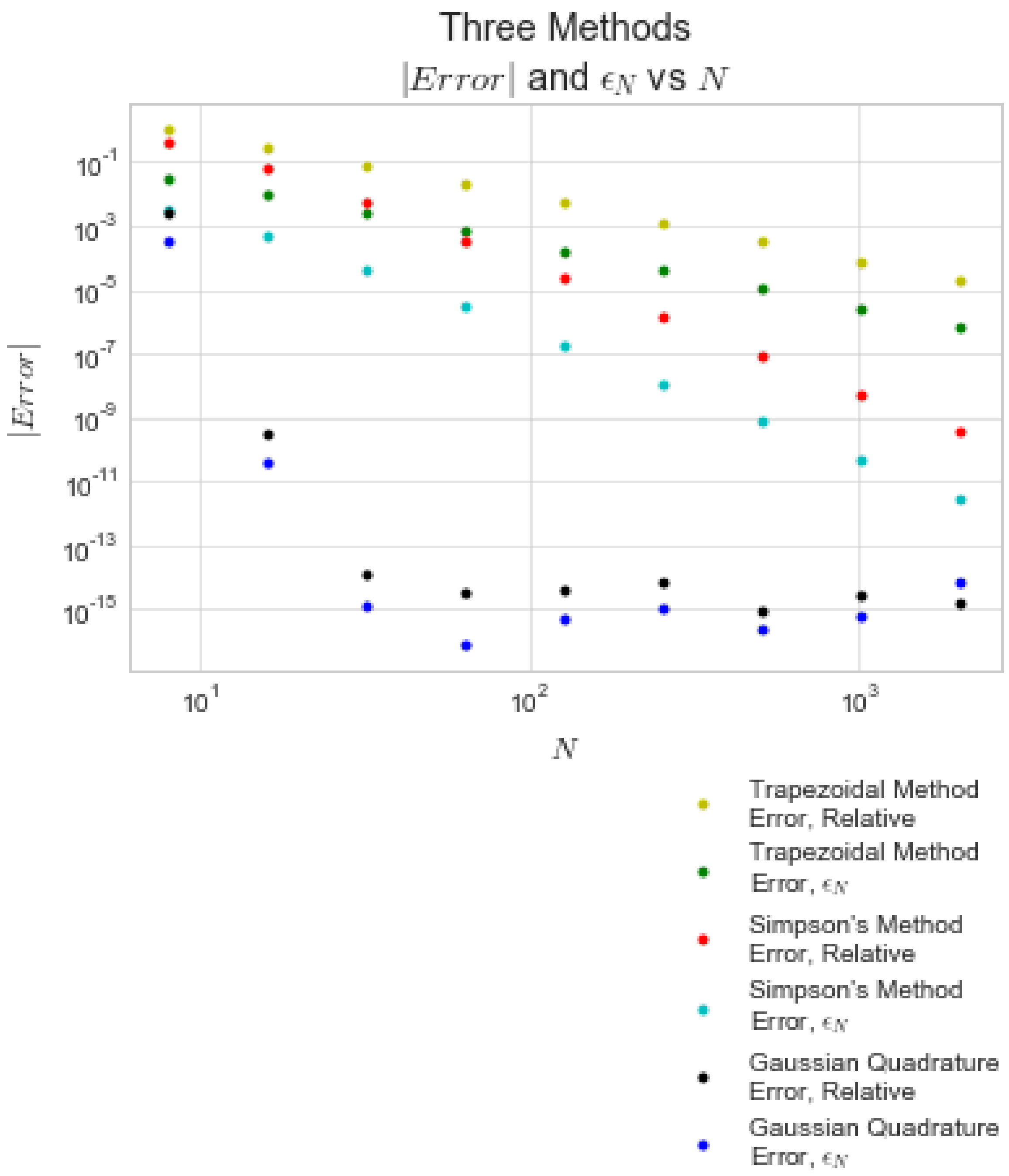
The true value for $D(4)$ (Eq. 1) is taken to be the `scipy.special.dawsn(4.0)` as it was done in Lab 2. From this "true value", the relative errors for each method. And as it is done in *Mark Newman, Computational Physics, Ch.5*, the methods' error can be interpreted as in equation 2.

$$\begin{gathered} TrapezoidalRule:\ \epsilon_N = \tfrac{1}{3}(I_{2N} - I_N) \\ Simpson'sRule:\ \epsilon_N = \tfrac{1}{15}(I_{2N} - I_N) \\ GaussianQuadrature: \epsilon_N = I_{2N} - I_N \end{gathered} \tag{2}$$

where I_N denote the method's output with N slices.

As can be seen from the 'Three Methods' graph below, although Simpson's Rule method is giving results with lower magnitude of error than Trapezoidal method, the Gaussian Quadrature method for integration is superbly more accurate than the other two methods as it reaches a $\sim 10^{-15}$ of error in only 3-4 steps of changing N (NOT "3-4 slices"!!!). The ϵ_N errors gotten from equation 2 are always smaller than the relative errors of the methods. One important thing to notice is that some of the errors in the Gaussian Quadrature method start to fluctuate very much as N increases. This is because the y-axis got so small that now the rounding errors are significant. The fact that this fluctuation happens around 10^{-16} is not a surprise because that is the error constant, $C \simeq 10^{-16}$. Therefore, it is wise to set the desired error to $\sim 10^{-15}$ and keep on increasing N until this value is reached and breaking the operation at that point.

The plot with logarithmic axis is as follows:



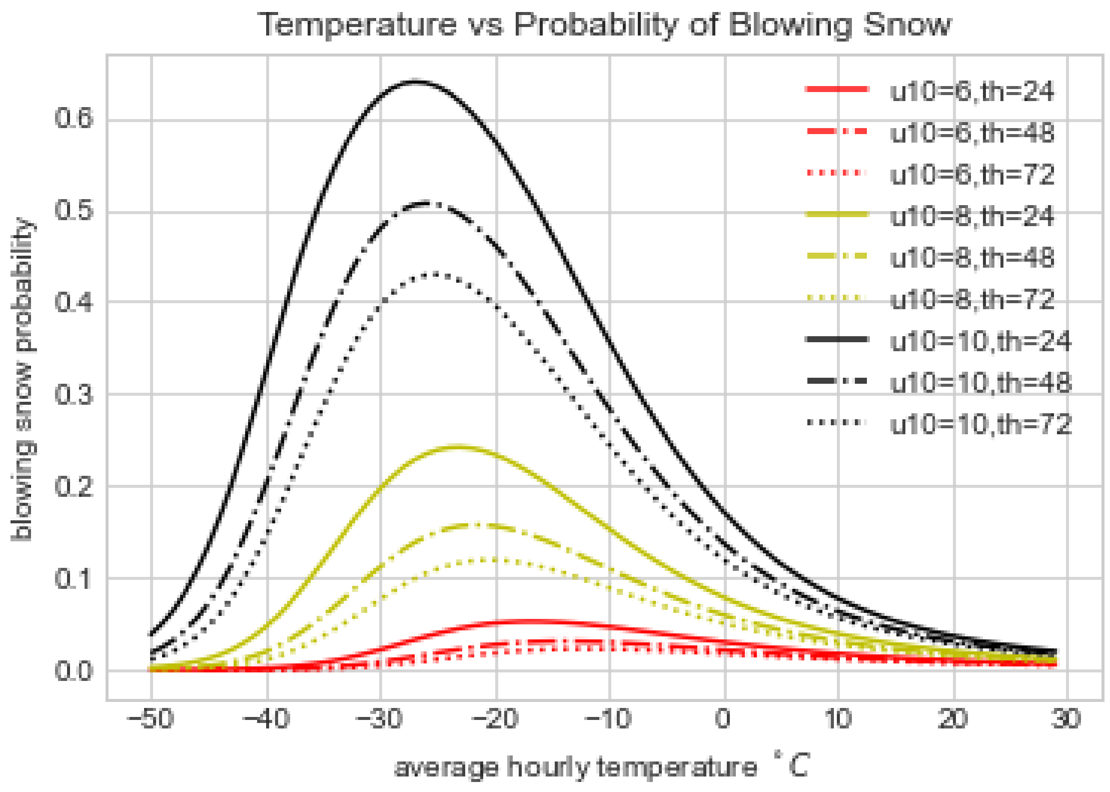
Magnitude of the errors of the three methods on logarithmic axis. ϵ_N refers to the error estimates (eq.2). Relative error refers to the method's error compared

to the real value D(4).
ϵ_N for the Gaussian quadrature seems to be the most accurate error estimate compared to the Trapezoidal error estimate and the Simpson error estimate because, of the three error estimates, the Gaussian quadrature error estimate was closest to it's own relative error as shown in the graph above.

1.b

The equation for the probability of blowing snow is: P(u_{10},T_a,t_h)=$\frac{1}{\sqrt{2\pi}S}\int_0^{u_{10}} e^{\frac{-(\bar{u}-u)^2}{2S^2}}du$
u_{10}- average hourly wind speed
T_a- average hourly temperature
t_h- surface age of snow

The "Temperature vs Probability of Blowing Snow" graph below indicates that a higher wind speed and younger snow is most likely to produce blowing snow. Older snow or slower wind speed lowers the probability of blowing snow to occur. This dependence makes sense because the older the snow is, the more likely it is to harden (in cold temperatures) or melt (in warm temperatures) so younger snow should be most likely to create blowing snow. Stronger wind speeds should also be more likely to create blowing snow because the wind is what's responsible for lifting the snow, so a larger wind speed results in a larger pushing force from the wind.



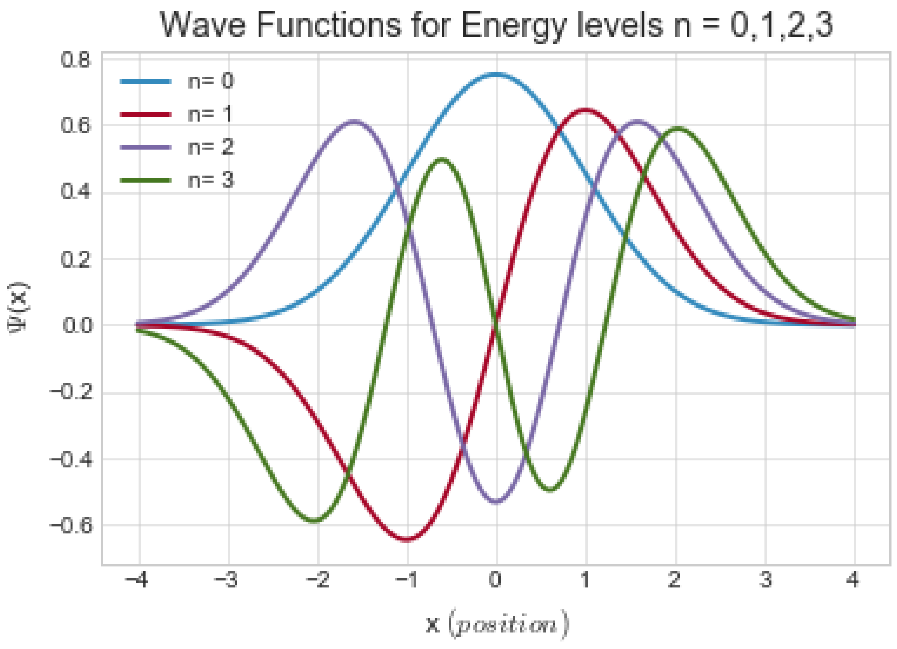
The temperature at which blowing snow is most likely to occur becomes lower as the wind speed increases as indicated by the graph above (the peak of the graph shifts to the left as the wind speed increases).

2 Calculating quantum mechanical observables

In this section we will observe the wave functions $\Psi_n(x)$ for energy levels n = 0,1,2,3, and 30, then we will discuss the relationship between the momentum uncertainty and the position uncertainty, and then we will try to find a simple rule to determine the total energy in each energy level.

2.a

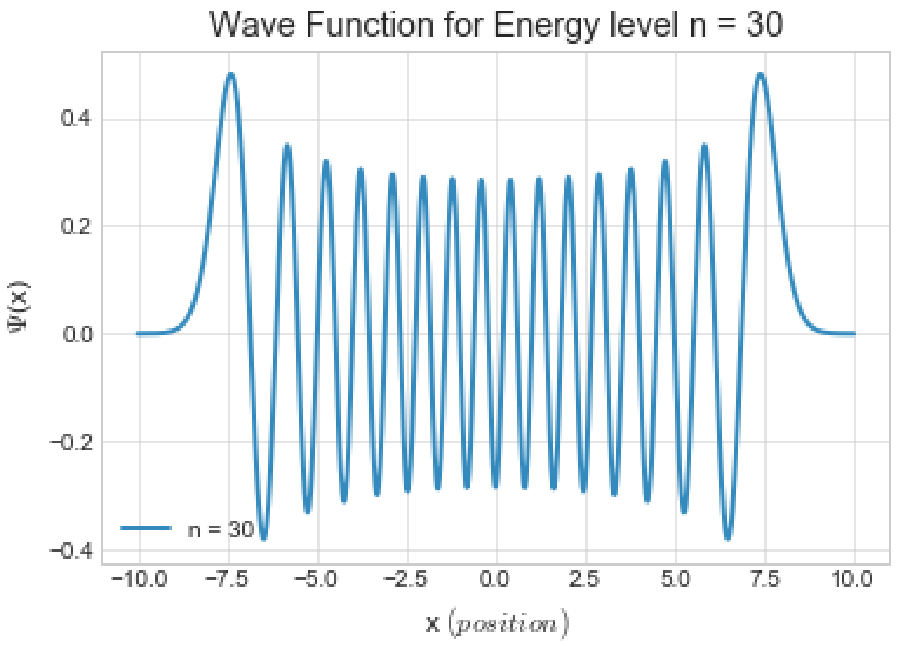
first we will be graphing the wave functions of the energy levels n=0,1,2,3:



The wave functions look very sinusoidal, with the number of anti-nodes in each wave function equal to n + 1. Also, the wave function is symmetric on x=0 (or in other words, is an even function about the y-axis on x=0) when n is an even number and the wave function is an odd function about the y-axis on x=0 when n is an odd number. The peak amplitude of the wave function decrease as n increases but the wave function converges to zero slower as n increases as well (you can see that $\Psi_0(x)$ converges to zero as $\| x \|$ increases sooner than $\Psi_3(x)$ does).

2.b

This next graph will be of the wave function of energy level n=30:



As previously mentioned, the peak amplitude for $\Psi_{n=30}(x)$ is less than the peak amplitude for $\Psi_{n=0,1,2,3}(x)$ and it's clear that $\Psi_{30}(x)$ converges to zero much later than $\Psi_0(x)$ or $\Psi_3(x)$ (when I say 'later' I mean for a larger value of $\| x \|$). The hypothesis that the number of anti-nodes is equal to n+1 proved to be correct here as well, because there are 31 anti-nodes in the graph of $\Psi_{30}(x)$.

2.c

In order to recognize a relationship between the momentum uncertainty and the position uncertainty, and to find a simple rule to calculate the total energy at each energy level, we will list the total energy, position uncertainty, and momentum uncertainty for energy levels 0 to 15 below:

Energy Level	Total Energy	Position Uncertainty	Momentum Uncertainty
0	0.49999999999998673	0.7071067811865381	0.7071067811865381
1	1.5000000000036098	1.2247448713930533	1.224744871393072
2	2.5000000000597233	1.5811388301053693	1.5811388301007825
3	3.4999999993291766	1.8708286932386666	1.8708286931767049
4	4.499999986345653	2.121320340012266	2.121320340670298
5	5.499999998440318	2.3452078737858177	2.3452078853725613
6	6.500000839948106	2.549509922378741	2.5495099206687857
7	7.500002775286724	2.7386136080453207	2.738612980397639
8	8.499980879868204	2.9154735796163544	2.9154717570738717
9	9.4998875365148	3.0821820851471116	3.0821954297266423
10	10.500125202013178	3.2403545427157727	3.2404247933762655
11	11.501905806042211	3.3914934063563136	3.391398544365742
12	12.501629150524163	3.5363391918530813	3.5351893048622194
13	13.483844689962707	3.672389154188237	3.671681805402807
14	14.465493989165875	3.798448863035311	3.808250807554238
15	15.564184298625545	3.9362504543943655	3.954023388591833

From the list above, it's clear that as the energy level increases, both the momentum and position uncertainties increase as well. Momentum and position uncertainties are almost identical through energy levels 0 to 15 so it's fair to assume that $\sqrt{<x^2>} \approx \sqrt{<p^2>}$. The Total Energy of the system seems to rely on a simple rule that E= $n + \frac{1}{2}$, where n is the energy level and E is the total energy.

3 Generating a relief map

3.a

In this part, goal is to set mind path for generating the illumination and gradient graphs for the region of Lake Geneva at the border between France and Switzerland. The data is going to be retrieved from NASA Space Shuttle Radar Topography Mission (SRTM).[1]
Pseudo code for this part as follows:

- Find the coordinates of Lake Geneva
- Import `numpy`, `matplotlib.pyplot`, `struct`
- Define the destination of the *.hgt* file on the computer
- Read the file into a buffer, two bytes at a time with a loop
- Reshape the data array into a (1201,1201)

1. NASA Space Shuttle Radar TopographyMission, http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/.

- Define the 2D-gradient function, use slicing to treat the interiors and the edges differently
- Define the illumination function
- Calculate gradients and illuminations
- set ϕ and h
- Plot height, illumination and gradients using `plt.imshow`

After doing the coding along the lines of the above pseudo-code, results are in section 3.b.

3.b

In this part two things are going to be treated as constants: $h = 83$ and $\phi = \pi/6$. Although h(physical distance between each of the indices) is theoretically not a constant because it is a geoid.
Since $\phi = \pi/6$, the sun is hitting from the south-west direction.
Right after getting the height data from the `N46E006.hgt` file, when calculating the partial derivatives, slicing had been used (for more info:[2]). One way to check if the borders had been treated is to check the dimension of the $\frac{\partial w}{\partial x}$ and $\frac{\partial w}{\partial y}$ arrays. After writing an if-statement that prints out the result of the check, what is gotten is:

```
No value went missing => borders were treated correctly
```

This means that the `.shape` of the original data that carries the height information matches the `.shape` of the gradient arrays. Therefore, borders were treated correctly.
Below are the graphs of *Height*, *Intensity of Illumination*, $\frac{\partial w}{\partial x}$ and $\frac{\partial w}{\partial y}$ graphs for the Lake Geneva.

Figure 1 shows that the topography around the south eastern side of Lake Geneva is quite rocky, reaching to altitudes as high as ~ 3200 metres. Despite these altitudes, there are also sea level altitudes (0 metres, marked with blueish colors) implying the existence of a body of water, which is the Lake Geneva in this case.

Although the illumination of the sides of the mountains might mislead the viewer to think that the Sun is shining from south-east in Figure 1 because of the way the mountains stretch, when observed carefully, it is visible that the sun is shining from $\phi = \pi/6$ (south-west). One other way to prove is would be to exaggerate the ϕ value and set it to π -sunset- and observe that the Sun is in fact shining from west which would mean that the way ϕ is coded is correct.

2. Greg Hewgill, *Understanding slice notation*, https://stackoverflow.com/questions/509211/understanding-slice-notation.

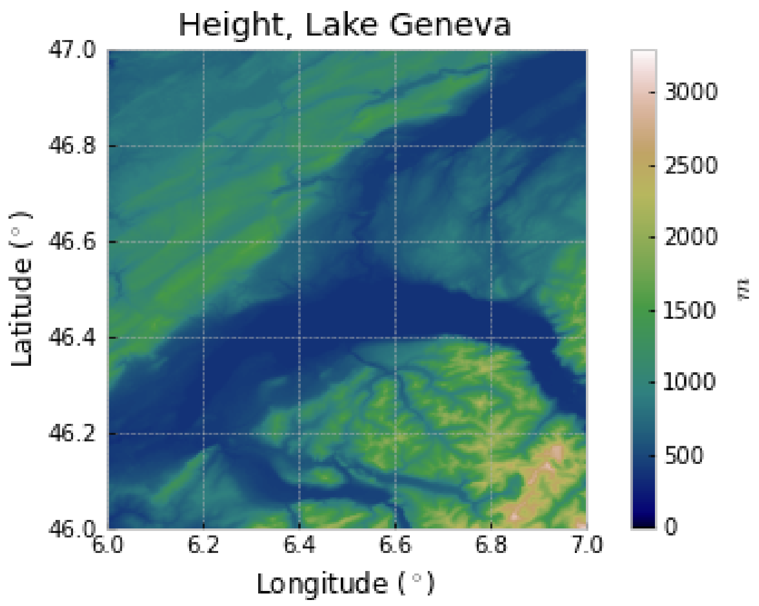


Figure 1: Height of the topography around Lake Geneva

But it is not necessary to do this check since this exaggerated plot would give a result similar in Figure 3 since the way $\frac{\partial w}{\partial y}$ is coded is going from west to is just like the rays from the sun would shine on the region at sunset.
Below are the graphs of the partial derivatives:

From the graphs of the partial derivatives (Figures 3 & 4), it can be seen that these graphs are only good at reflecting the north-south and west-east properties. The intensity of illumination plot. (Figure 2) was doing better at showing the formation of the ridges.
Next four plots (Figures 5,6,7,8) show the zoomed in topographies of 4 of the well known towns around Lake Geneva (Geneva, Lausanne, Évian and CERN):[3]

3. *Latitude and Longitude Finder*, https://www.latlong.net/.

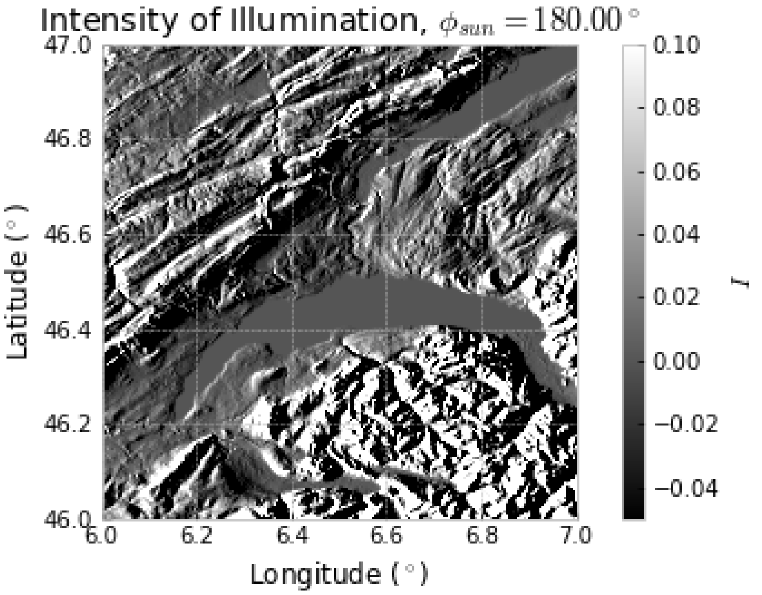


Figure 2: Intensity of illumination of the topography around Lake Geneva

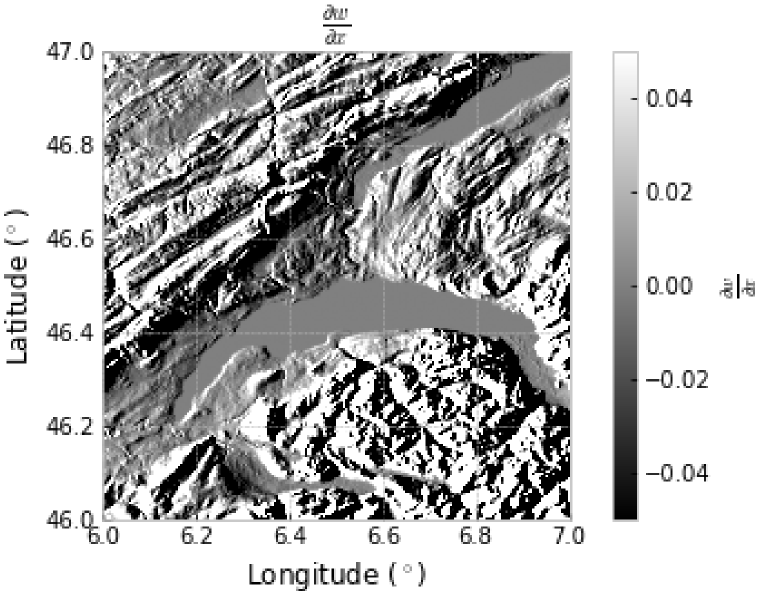


Figure 3: $\frac{\partial w}{\partial x}$ of the topography around Lake Geneva

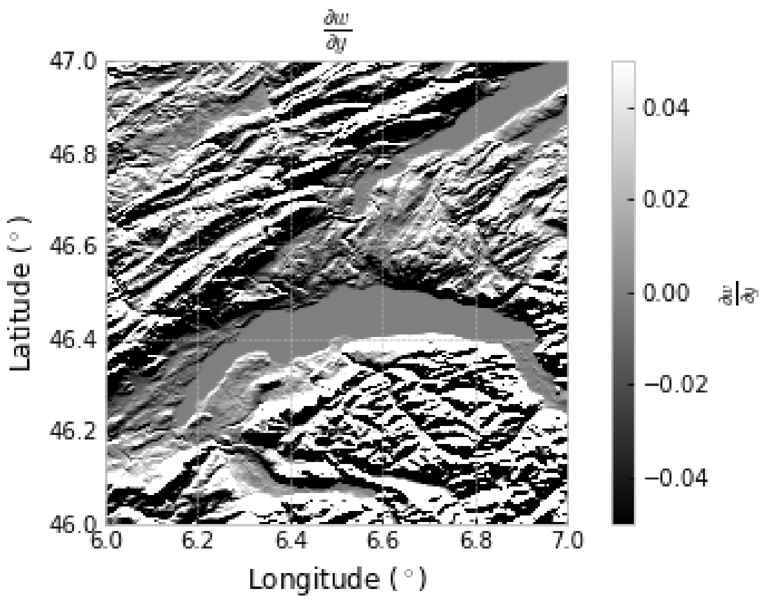


Figure 4: $\frac{\partial w}{\partial y}$ of the topography around Lake Geneva

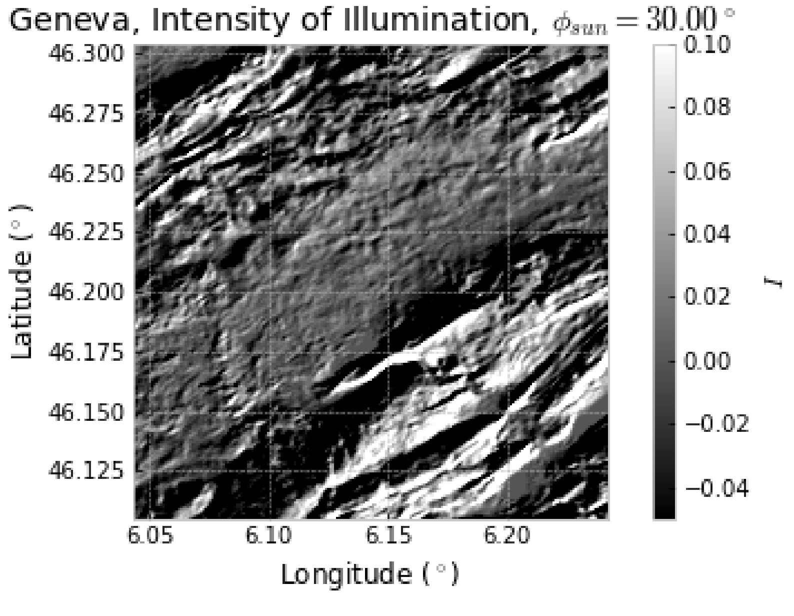


Figure 5: The topography around Geneva

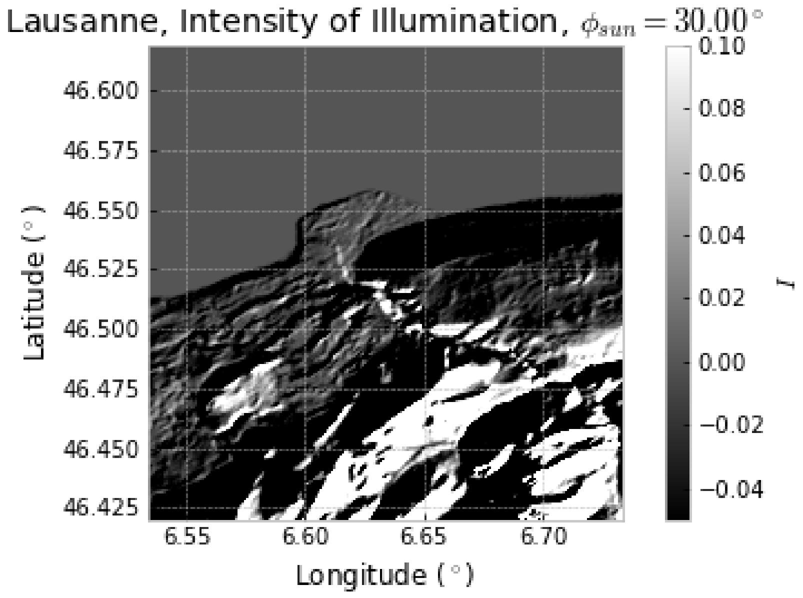


Figure 6: The topography around Lausanne

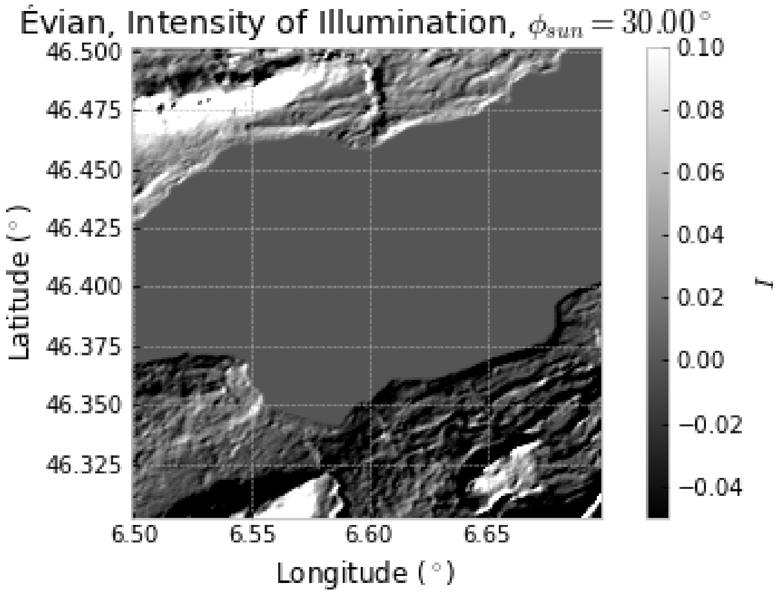


Figure 7: The topography around Évian

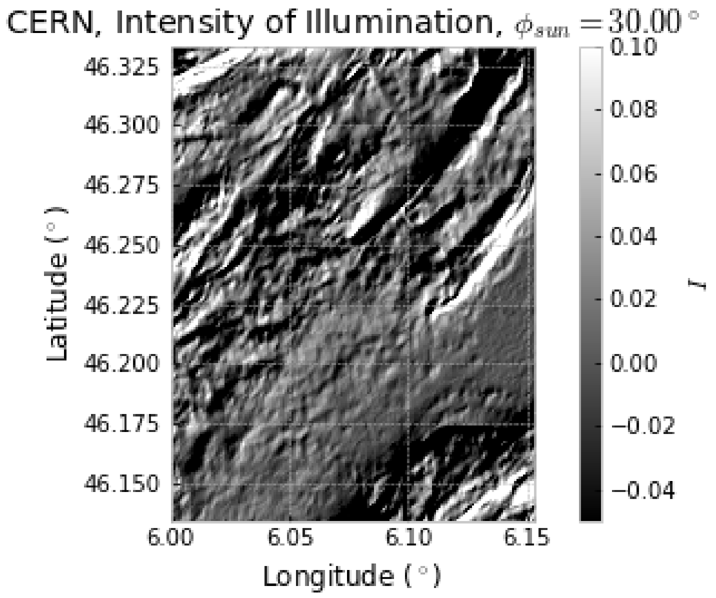


Figure 8: The topography around CERN