vaughan-lewis-survey

July 29, 2024

[1]:
```
from datetime import datetime
import numpy as np
import pandas
import matplotlib.pyplot as plt
import pyproj
import rasterio
```

Load in our survey data.

[2]:
```
survey1 = pandas.read_csv("Vaughan-Lewis Icefall Day 2.csv")
survey2 = pandas.read_csv("Vaughan-Lewis Icefall Day 7.csv")
```

Extract the lat/lon coordinates of the survey points.

[3]:
```
lat1, lon1 = survey1["Latitude"], survey1["Longitude"]
lat2, lon2 = survey2["Latitude"], survey2["Longitude"]
```

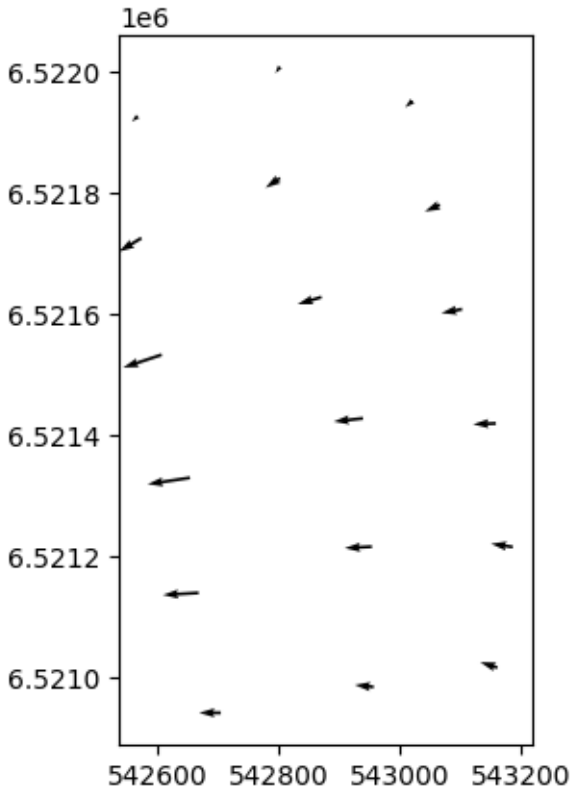
Convert the coordinates to UTM zone 8.

[4]:
```
transformer = pyproj.Transformer.from_crs(4326, 32608)
x1, y1 = transformer.transform(lat1, lon1)
x2, y2 = transformer.transform(lat2, lon2)
```

Compute and plot the displacements.

[5]:
```
dx, dy = x2 - x1, y2 - y1
```

[6]:
```
fig, ax = plt.subplots()
ax.set_aspect("equal")
ax.quiver(x1, y1, dx, dy);
```



Get all the timestamps for when the points were surveyed and compute the elapsed time between the first and second surveys.

[7]:
```
time_strings1 = survey1["Averaging start"]
time_strings2 = survey2["Averaging start"]
fmt = "%Y-%m-%d %H:%M:%S"
length = 19  # Cut off some unnecessary data
times1 = [datetime.strptime(timestr[:length], fmt) for timestr in time_strings1]
times2 = [datetime.strptime(timestr[:length], fmt) for timestr in time_strings2]
dt = np.array([(t2 - t1).total_seconds() for t1, t2 in zip(times1, times2)])
```

Compute the velocities.

[8]:
```
seconds_per_year = 365.25 * 24 * 60 * 60
vx = dx / dt * seconds_per_year
vy = dy / dt * seconds_per_year
vx, vy
```

[8]:
```
(array([ -18.83769635,  -69.05829705, -119.56198825, -131.58849532,
        -111.20230755,  -67.24839916,  -58.80982963,  -84.23566758,
         -91.00408294,  -74.0524971 ,  -44.70863633,  -14.89870474,
         -23.7421618 ,  -46.99914602,  -64.76201893,  -70.73992211,
```

```
        -67.71968712,  -53.42793177]),
 array([-16.7594527 , -42.06337596, -39.78283345, -20.05219762,
        -6.44917284,   0.15495326,   6.27346445,  -4.60792164,
       -10.15237215, -22.03855476, -30.27140326, -20.10034641,
       -20.0055315 , -21.0507362 , -12.66738101,  -2.58992119,
        10.21071782,  17.49502191]))
```

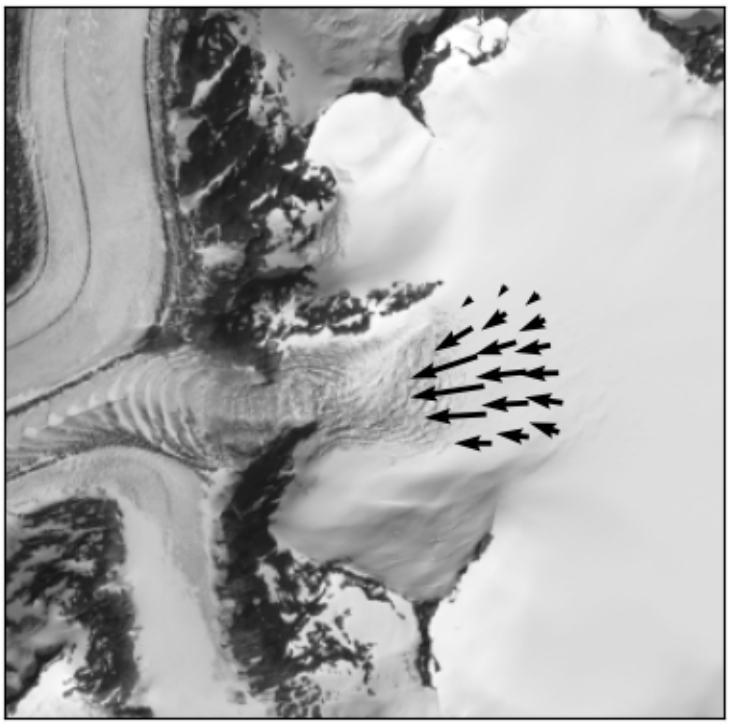
Load in some satellite imagery.

[9]:
```
xmin, xmax = 539400.0, 544280.0
ymin, ymax = 6519080.0, 6523900.0
with rasterio.open("LC08_L1TP_058019_20230703_20230717_02_T1_B8.TIF", "r") as␣
 ↪src:
    transform = src.transform
    window = rasterio.windows.from_bounds(
        left=xmin, right=xmax, bottom=ymin, top=ymax, transform=transform
    )
    image = src.read(indexes=1, window=window)
```

Plot the velocities on top of the satellite imagery.

[10]:
```
extent = (xmin, xmax, ymin, ymax)
vmin, vmax = 6500, 46550

fig, ax = plt.subplots()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
ax.set_xlim((xmin, xmax))
ax.set_ylim((ymin, ymax))
ax.imshow(image, extent=extent, vmin=vmin, vmax=vmax, cmap="Greys_r")
ax.quiver(x1, y1, vx, vy);
```



[11]:
```
v = np.column_stack((vx, vy))
X1 = np.column_stack((x1, y1))
X2 = np.column_stack((x2, y2))
```

Compute the extensional strain rate by evaluating the magnitude of the velocity difference divided by the distance between points 2 and 14, which are respectively in the survey lines closest to and furthest from the icefall.

[12]:
```
dv = v[2] - v[14]
dx = X1[14] - X1[2]
extensional_strain_rate = np.sqrt(np.sum(dv**2)) / np.sqrt(np.sum(dx**2))
print(f"Extensional strain rate: {1000 * extensional_strain_rate:.2f} (m / yr) /
  ↪ km")
```

```
Extensional strain rate: 121.98 (m / yr) / km
```

Same but for the lateral strain rate and points 0 (closest to the nunatak) and 2 (centerline of the flow).

[13]:
```
dv = v[2] - v[0]
dx = X1[2] - X1[0]
lateral_strain_rate = np.sqrt(np.sum(dv**2)) / np.sqrt(np.sum(dx**2))
print(f"Lateral strain rate: {1000 * lateral_strain_rate:.2f} (m / yr) / km")
```

```
Lateral strain rate: 261.25 (m / yr) / km
```

Now compute some components of the stress tensor using Glen's flow law and assuming that $A \approx 150; \mathrm{yr}^{-1}\ \mathrm{MPa}^{-3}$ for temperate ice.

[14]:
```
A = 150.0  # 1 / yr * MPa^{-3}
n = 3
_extensional = 1000 * (extensional_strain_rate / A) ** (1 / n)
print(f"Extensional stress: { _extensional:.4f} kPa")
```

```
Extensional stress: 93.3384 kPa
```

[15]:
```
_lateral = 1000 * (lateral_strain_rate / A) ** (1 / n)
print(f"Lateral stress: { _lateral:.4f} kPa")
```

```
Lateral stress: 120.3161 kPa
```

Some day maybe we'll compute a strain rate map using the Delaunay triangulation.

[16]:
```
import scipy
triangulation = scipy.spatial.Delaunay(X1)
```

[17]:
```
fig, ax = plt.subplots()
ax.set_aspect("equal")
ax.triplot(X1[:, 0], X1[:, 1], triangulation.simplices);
```