Recurrence Analysis of Climate Variation in the Northern Hemisphere

Toni Schmidt, Nikolaos Kordalis, Kathia Ivett Cuevas Martinez

March 26, 2020

[1]:

```python
# import libraries
import numpy as np
from scipy import signal
import nda.recurrence as rc
import nda.recurrence_quantification as rq
import matplotlib.pyplot as pl
from mpl_toolkits.mplot3d import Axes3D

# configure plot appearance
pl.rcParams.update({'font.size': 18,
                    'lines.linewidth': .7,
                    'grid.linestyle': ':',
                    'xtick.minor.visible': True,
                    'ytick.minor.visible': True,
                    'font.sans-serif': 'Helvetica'})
```

[2]:

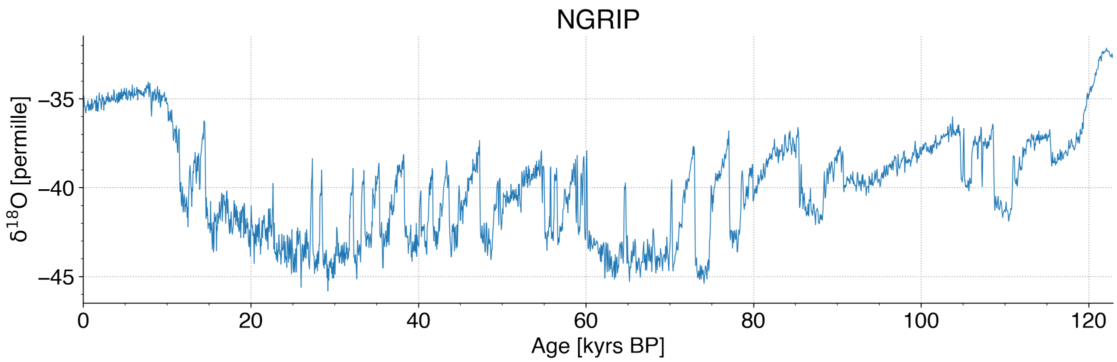
```python
# set up function that computes tau-recurrence rate
def rrt(rm, s=1.):
    """
    Equation from Marwan et al., 2007
    (doi:10.1016/j.physrep.2006.11.001)

    Input:
        rm  [recurrence matrix]

    Output:
        tau [tau delay]
        rrt [tau-recurrence rate]
    """
    # length 'n' of data series
    n = len(rm)

    # delay 'tau'
    tau = np.arange(-n+1, n)
```

```
    # tau-recurrence rate 'rrt'
    rrt = []

    for t in tau:
        # tau-th diagonal 'd' in the rp
        d = np.diag(rm, k=-t)

        # compute recurrence rate for respective diagonal
        rrt.append(sum(d)/len(d))

    return tau*s, rrt

# set up function that computes the power spectrum of a time series
def ps(t, x, s=1):
    """
    After Mathworks: Power Spectral Density Estimates Using FFT
    (https://de.mathworks.com/help/signal/ug/
 ↪power-spectral-density-estimates-using-fft.html)

    Input:
        t  [time array]
        x  [data array]
        s  [scaling factor, i.e. sampling interval]

    Output:
        p  [sampling period]
        ps [power spectrum]
    """
    # if time array 't' is odd then remove last time point
    if len(t)%2 == 1:
        t = t[:-1]

    # time range 'tr'
    tr = t.max()-t.min()

    # length 'n' of time series
    n = len(x)

    # derive sampling period 'p'
    p = np.arange(0, tr/2+s, s)

    # one-dimensional discrete Fourier transform 'ft'
    ft = np.fft.fft(x)
    ft = ft[:n//2+1]

    # power spectrum 'ps'
    ps = abs(ft)**2
```

```
    ps[1:-1] = 2 * ps[1:-1]

    return p, ps
```

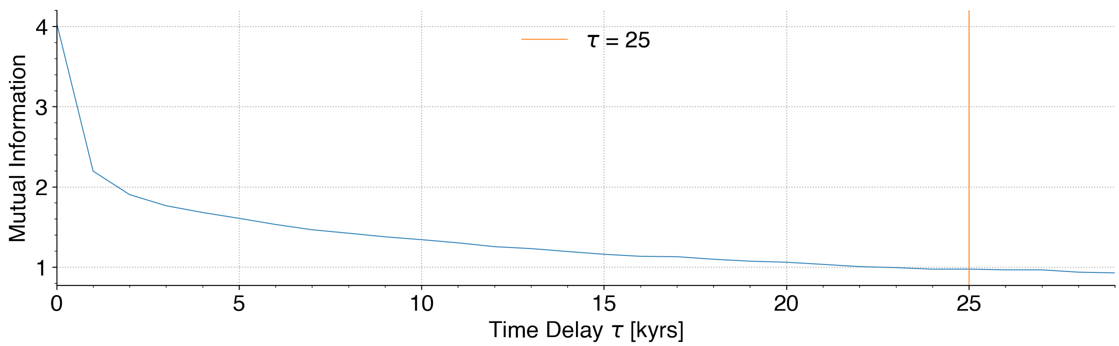
1 A frequently used climate proxy for the northern-hemisphere climate is NGRIP data. Download the data and preprocess the data if necessary.

[3]:
```
# load data 'd'
d = np.loadtxt('../1_data/41586_2004_BFnature02805_MOESM1_ESM.txt')

# slice data 'd' using every second row
d = d[::2]

# assign variables time 't' [myrs BP] and d180 'x' [permille]
t = d[:,0]
x = d[:,1]

# remove data 'd' from memory
del(d)

# convert unit of time 't' from [myrs BP] to [kyrs BP]
t /= 1e6

# plot data
pl.figure(figsize=(15,4), dpi=300)
pl.plot(t, x)
pl.xlim(0, t.max())
pl.title('NGRIP')
pl.xlabel('Age [kyrs BP]')
pl.ylabel(r'$\rm\delta^{18}$O [permille]')
pl.gca().spines['top'].set_visible(False)
pl.gca().spines['right'].set_visible(False)
pl.grid();
```

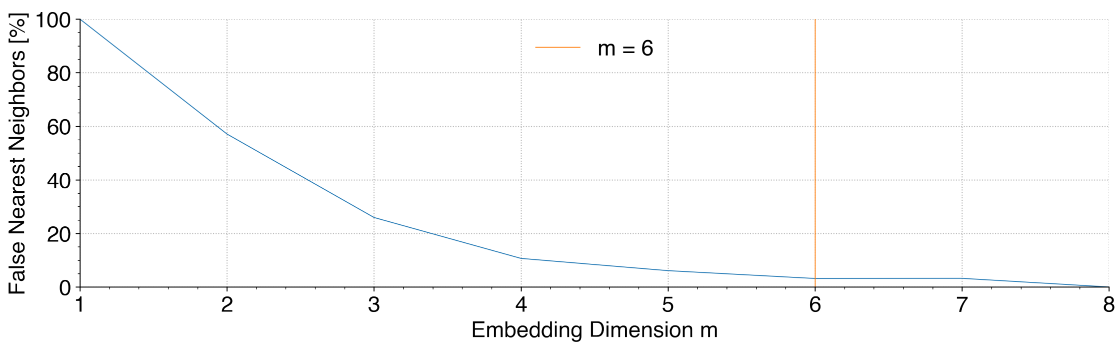


2 Estimate embedding parameters and reconstruct and present the phase space trajectory.

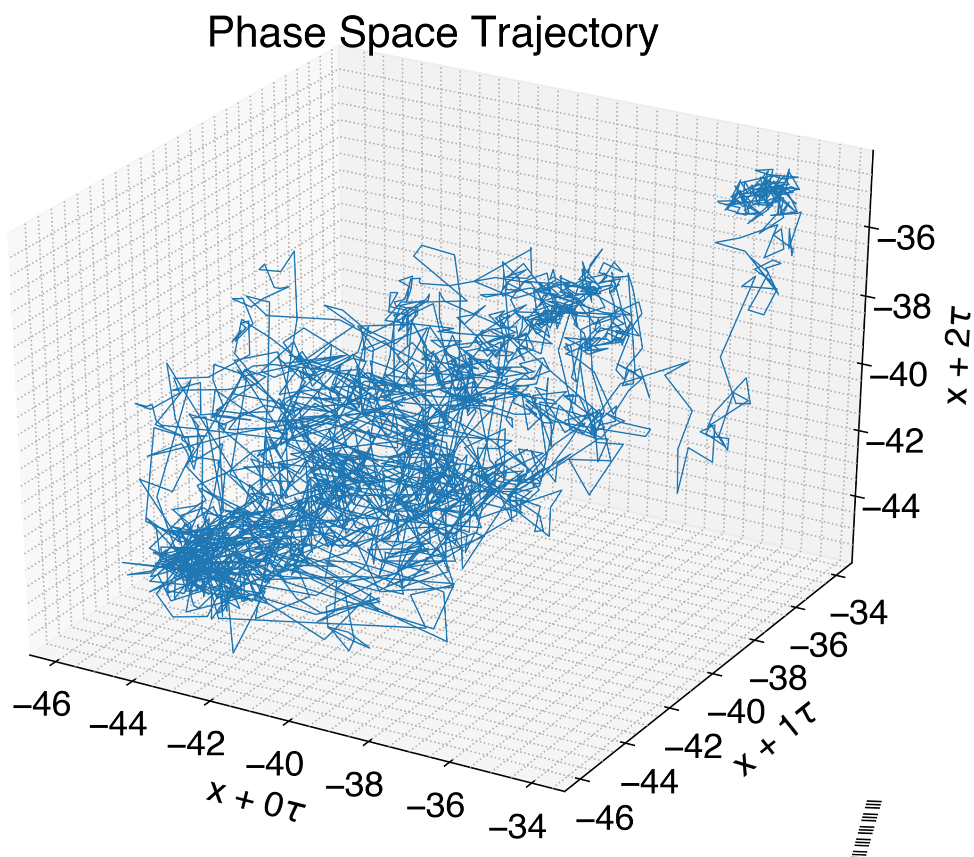
[4]:
```
# calculate mutual information 'mi' per time delays 'taus'
mi, taus = rc.mi(x, 30)

# estimate time delay 'tau'
tau = rc.first_minimum(mi)

# plot mutual information
pl.figure(figsize=(15,4), dpi=300)
pl.plot(taus, mi)
pl.axvline(tau, c='C1', label=r'$\tau$ = ' + str(tau))
pl.xlim(0, taus.max())
pl.xlabel(r'Time Delay $\tau$ [kyrs]')
pl.ylabel('Mutual Information')
pl.legend(loc='upper center', frameon=False)
pl.gca().spines['top'].set_visible(False)
pl.gca().spines['right'].set_visible(False)
pl.grid();
```



[5]:
```
# calculate fraction of false nearest neighbors 'fn' per embedding dimensions␣
↪'ms'
fn, ms = rc.fnn(x, tau, 8, r=1)

# estimate suitable embedding dimension 'm'
m = rc.first_minimum(fn)

# plot mutual information
pl.figure(figsize=(15,4), dpi=300)
pl.plot(ms, fn*100)
pl.axvline(m, c='C1', label='m = ' + str(m))
pl.xlim(1, ms.max())
pl.ylim(0, 100)
pl.xlabel('Embedding Dimension m')
pl.ylabel(r'False Nearest Neighbors [%]')
pl.legend(loc='upper center', frameon=False)
pl.gca().spines['top'].set_visible(False)
pl.gca().spines['right'].set_visible(False)
pl.grid();
```



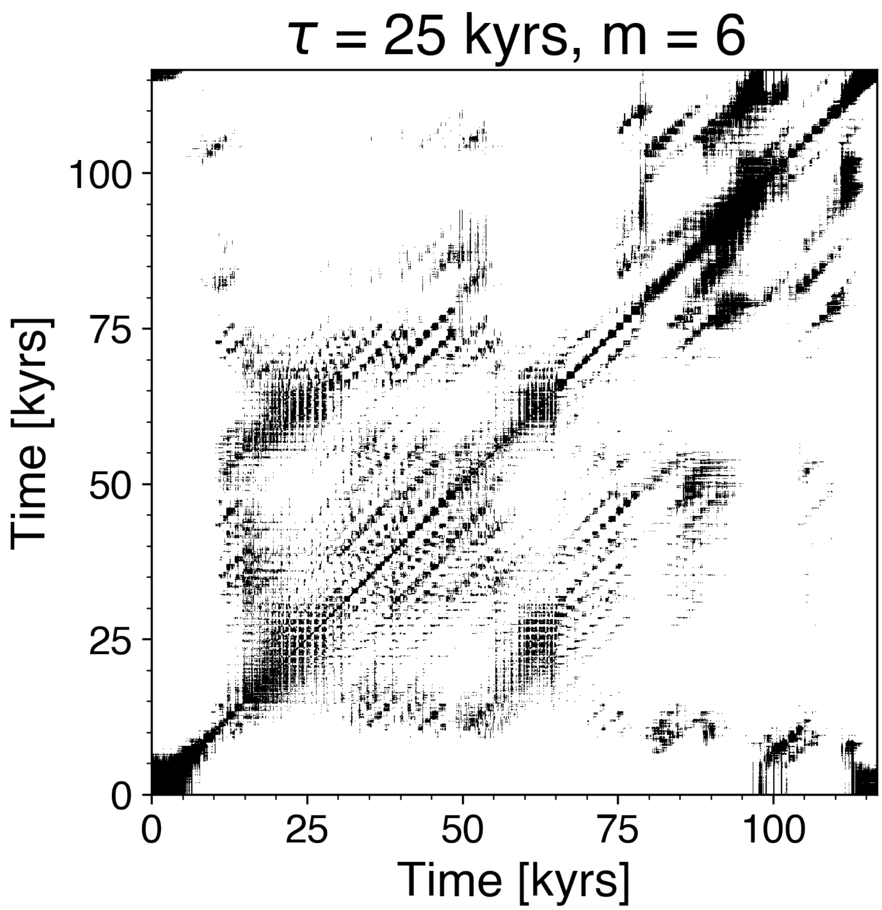
[6]:
```
# embed the time series 'xe'
xe = rc.embed(x, m, tau)

# plot first 3 dimensions of phase space trajectory
pl.figure(figsize=(10,8), dpi=300)
ax = pl.gca(projection='3d')
ax.plot(xe[:,0], xe[:,1], xe[:,2])
ax.set_title('Phase Space Trajectory')
ax.set_xlabel('\n' + r'x + 0$\tau$')
ax.set_ylabel('\n' + r'x + 1$\tau$')
ax.set_zlabel('\n' + r'x + 2$\tau$');
```



3 Construct a RP of this embedded time series. Check different options of recurrence criteria and discuss your final choice. Interpret the visual appearance of the RP.

[7]:
```
# derive recurrence matrix 'rm'
rm = rc.rp(x, m, tau, 0.1, threshold_by='fan')

# plot recurrence plot
pl.figure(figsize=(5,5), dpi=300)
pl.imshow(rm, cmap='binary', origin='lower')
pl.title(r'$\tau$ = %s kyrs, m = %s' %(tau, m))
ax_is = pl.gca().get_xticks()[1:-1]
ax_be = [int(ax_is[i] * 50 * 1e-3) for i in range(len(ax_is))]
pl.gca().set_xticks(ax_is)
pl.gca().set_xticklabels(ax_be, fontsize=15)
pl.gca().set_xlabel('Time [kyrs]');
pl.gca().set_yticks(ax_is)
```

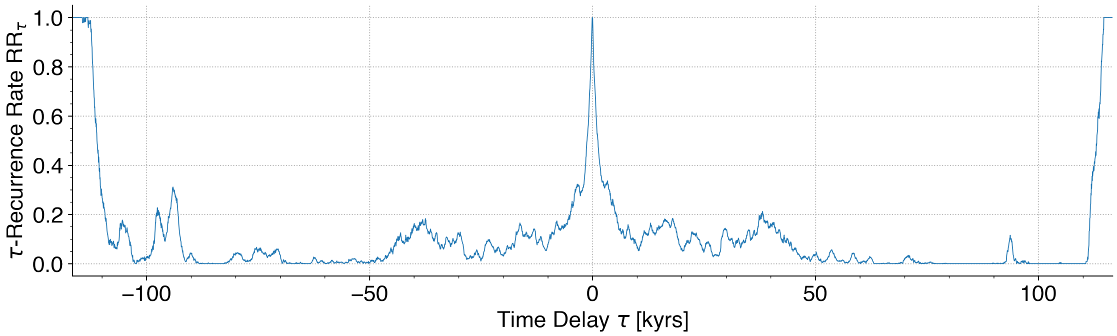
```
pl.gca().set_yticklabels(ax_be, fontsize=15)
pl.gca().set_ylabel('Time [kyrs]');
```



4 Compute the -recurrence rate as a function of and display the results. Check different filtering options (low-pass, high-pass, band-pass filter) and their impact on the -recurrence rate.

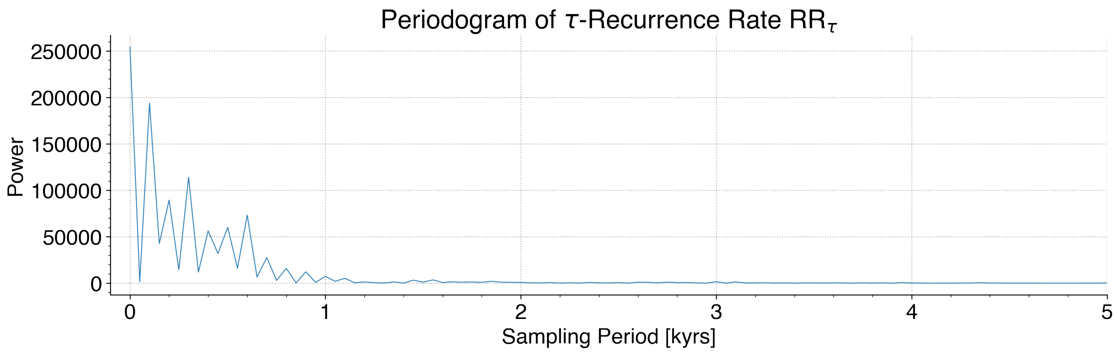
[8]:
```
# compute tau-recurrence rate 'rr' as a function of tau 'rt'
rt, rr = rrt(rm, 50*1e-3)

# plot tau-recurrence rate
pl.figure(figsize=(15,4), dpi=300)
pl.plot(rt, rr)
pl.xlim(rt.min(), rt.max())
pl.xlabel(r'Time Delay $\tau$ [kyrs]')
```

```
pl.ylabel(r'$\tau$-Recurrence Rate RR$_\tau$')
pl.gca().spines['top'].set_visible(False)
pl.gca().spines['right'].set_visible(False)
pl.grid();
```



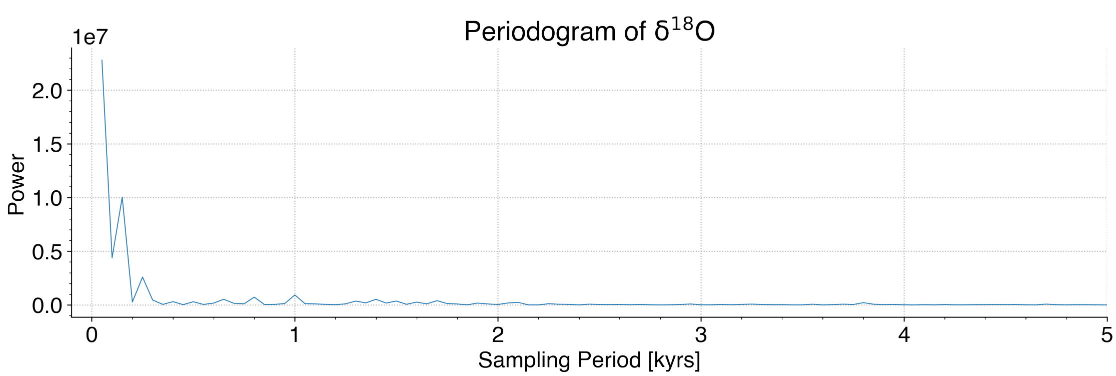
5 Make a fourier transform of all the -recurrence rate and show the squared absolute values of these transforms as a function of the sampling period (recurrence based powerspectrum).

[9]:
```
# compute power spectrum of tau-recurrence rate 'pse'
# for different sampling periods 'pe'
pe, pse = ps(rt, rr, 50 * 1e-3)

# plot power spectrum of tau-recurrence rate
pl.figure(figsize=(15,4), dpi=300)
pl.plot(pe, pse)
pl.xlim(-.1, 5)
pl.title(r'Periodogram of $\tau$-Recurrence Rate RR$_\tau$')
pl.xlabel('Sampling Period [kyrs]')
pl.ylabel('Power')
pl.gca().spines['top'].set_visible(False)
pl.gca().spines['right'].set_visible(False)
pl.grid();
```



6 Make a Fourier Analysis of the original time series and compare and discuss the powerspectrum with the recurrence based powerspectrum.

[10]:
```
# compute power spectrum of tau-recurrence rate 'psx'
# for different sampling periods 'px'
px, psx = ps(t, x, 50 * 1e-3)

# plot power spectrum of tau-recurrence rate
pl.figure(figsize=(15,4), dpi=300)
pl.plot(px[1:], psx[1:])
pl.xlim(-.1, 5)
pl.title(r'Periodogram of $\rm\delta^{18}$O')
pl.xlabel('Sampling Period [kyrs]')
pl.ylabel('Power')
pl.gca().spines['top'].set_visible(False)
pl.gca().spines['right'].set_visible(False)
pl.grid();
```



References

Andersen, K., Azuma, N., Barnola, J. et al. High-resolution record of Northern Hemisphere climate extending into the last interglacial period. Nature 431, 147–151 (2004). https://doi.org/10.1038/nature02805