Project 7 (week 2.3): Signal Processing

Answers to task 3 and task 10 (including Python code).

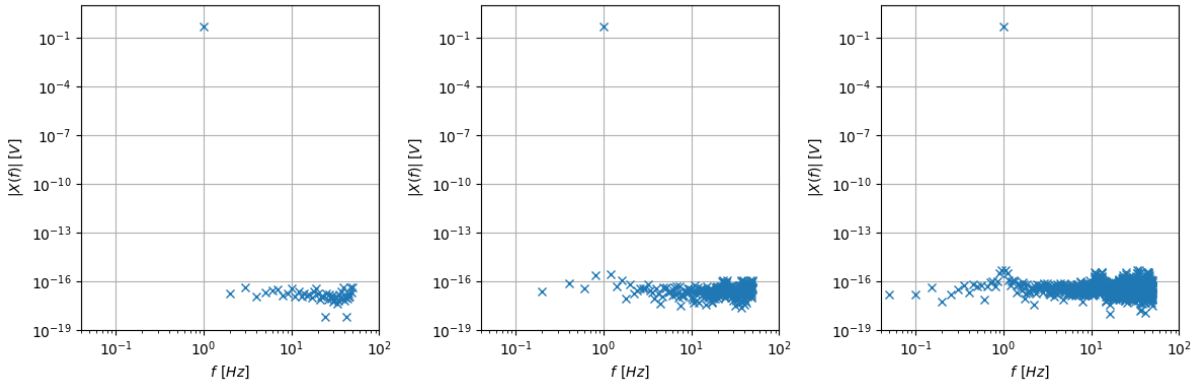
Answer to the other tasks will be provided in the form of feedback on your reports, if applicable, since the code and analysis in Task 3 is identical to the tasks that are not included here.

Task 3:

- Repeat Task 2 with different measurement times T_{meas} for the signal. Use measurement times such that the oscillation fits *exactly* 1 time, 5 times and 20 times.
- Plot the amplitude spectrum for all three measurement times, **only for positive frequencies**, in separate graphs (log-log scale) with the same domains and answer to the following questions:

  1. What is the effect of changing T_{meas} on the frequency range in the amplitude spectrum? Does the highest analysis frequency change?
  2. Does the frequency resolution change?
  3. Does the magnitude of at the peaks change?

```
# Create a list which contains the T_meas for the 3 cases
T_lst = [1, 5, 20]

# Set the sampling rate
f_s = 100

# Create a figure and loop over the 3 cases
plt.figure(figsize=(12,4))
for i, T_meas in enumerate(T_lst):

    # Set the time vector, amplitude, carrier frequency, phase shift, and then
the sinusoidal signal
    t_vec = np.arange(0, T_meas, 1/f_s)
    A = 1
    f_c = 1
    phi = 5 * np.pi / 180
    x = A * np.sin(2 * np.pi * f_c * t_vec + phi)

    # The number of samples and the computation of the DFT with the time-di-
mension restored
    N = len(x)
    X_cont = np.fft.fft(x) / N

    # The frequency resolution and the frequency vector with the analysis fre-
quencies
    f_0 = f_s / N
    f_vec = np.arange(0, f_s, f_0)

    # The result of the DFT for positive frequencies up to half of the sam-
pling rate f_s
    X_cont = X_cont[:N//2]
    f_vec = f_vec[:N//2]

    # Plotting
    plt.subplot(1, 3, i+1)
    plt.plot(f_vec, np.abs(X_cont), 'x')
    plt.loglog()
    plt.xlim(0.04, 100)
    plt.ylim(10**(-19), 10)
    plt.xlabel('$f \: \: [Hz]$')
    plt.ylabel('$|X(f)| \: [V]$')
    plt.grid()
    plt.tight_layout()
```



What is the effect of changing T_{meas} on the frequency range in the amplitude spectrum? Does the highest analysis frequency change?

No, the highest analysis frequency stays the same (as it is related to the sampling frequency, which we did not change). The analysis frequencies run until $f_s/2$. A two-sided spectrum was computed, however only the positive frequencies are shown.

Does the frequency resolution change?

Yes, the frequency resolution becomes better / finer (because $f_0 = \Delta f = 1/T_{meas}$ gets smaller).

Does the magnitude of $X(f)$ at the peaks change?

No, because we already divide by N (we already account for the measurement duration). The magnitude in the above plots is $\frac{A}{2}$ (no dependence on T_{meas}) .

Global Mean Sea-Level (GMSL) data (optional)

Task 10:

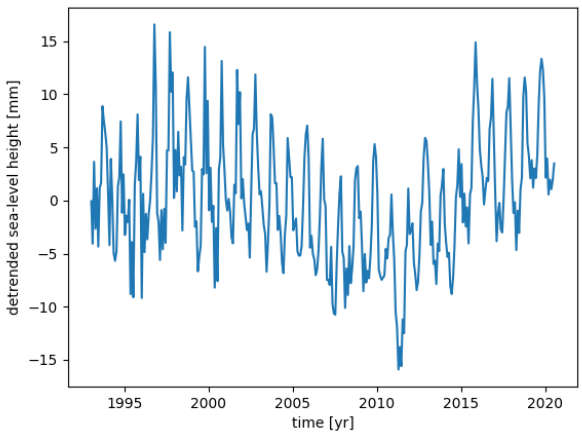
Detrend the data.

Estimate and plot power spectral density (PSD), hence the periodogram, for the (detrended) global mean sea-level data.

Identify the largest peak in the spectrum, what is the frequency, and can you come up with a physical explanation of this behaviour?

The code for reading the data was already given (defining arrays ‘data.iloc’, ‘t’, ‘T’, ‘y’, and variables ‘N’ and ‘dt’ as well).

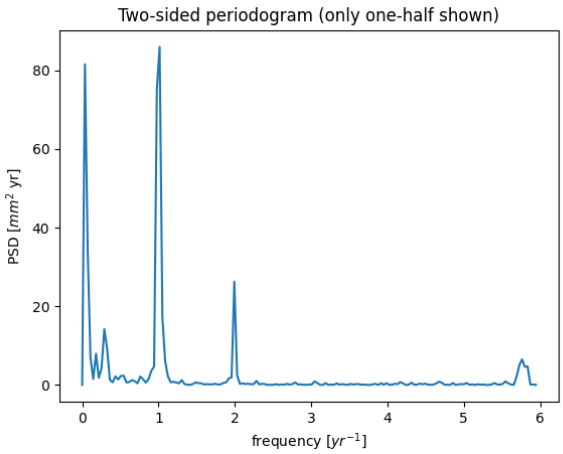
```
############################### Detrend the data ############################

# Prepare for detrending the data, we'll estimate an offset and a slope (two
unknown parameters)
A = np.ones((N, 2)) # Note that here A is the design matrix and not an ampli-
tude
A[:,1] = t

# (Unweighted) least-squares estimation of the offset and slope (both con-
tained in the 2D vector xhat)
xhat = (np.linalg.inv(A.T @ A) @ A.T) @ y

# Estimated observations (fitted by a straight line with offset and slope)
yhat = A @ xhat

# Difference of observed value and estimated observation (least-squares resid-
uals)
ehat = y - yhat

# Hence observed time-series but detrended, this will act as our signal x(t),
or x_0,...,x_{N-1}
plt.figure()
plt.plot(data.iloc[:,0], ehat)
plt.xlabel('time [yr]')
plt.ylabel('detrended sea-level height [mm]')
```



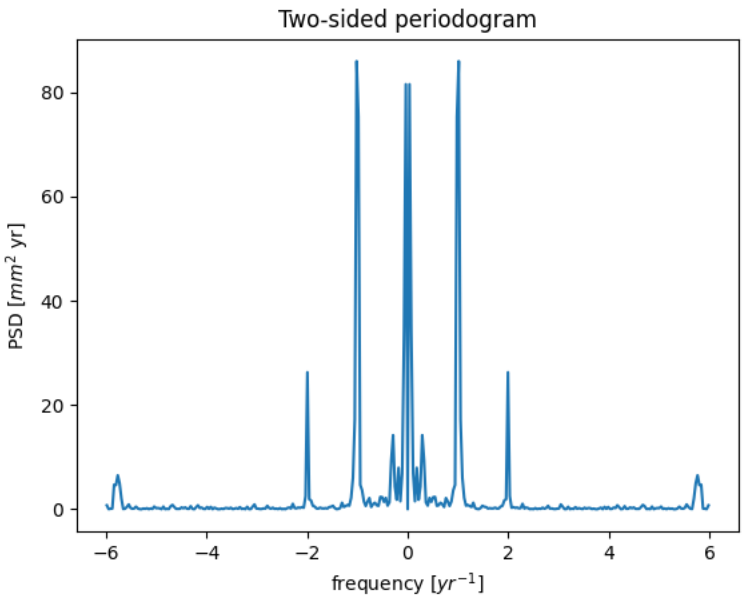
```
# Sampling frequency [Hz]
f_s = 1 / dt

# Discrete Fourier Transform (DFT) by fft
# and divide by N (as to maintain analogy with continuous-time Fourier trans-
form)
X_cont = np.fft.fft(ehat)/N

# frequency resolution
f_0 = f_s / N
```

Below *three* flavours of the (same) periodogram are presented. The axes are in linear scale (not logarithmic).

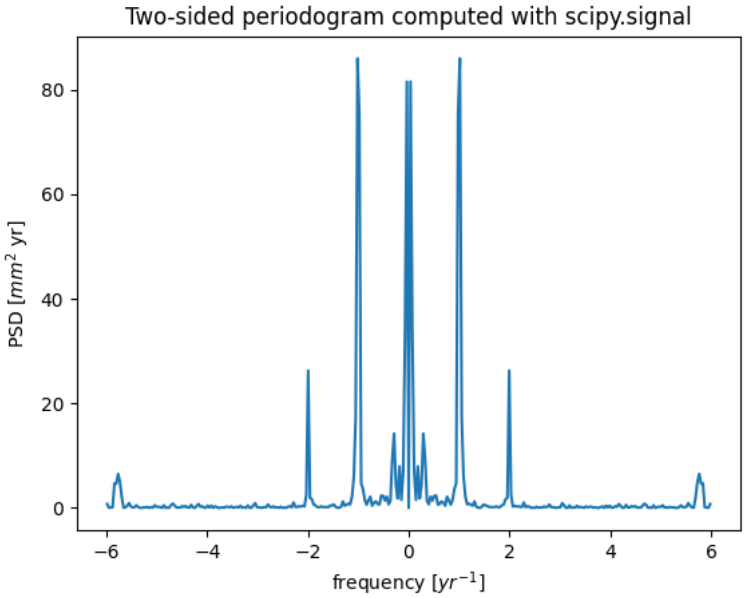
```
######## Plot the Double-Sided Periodogram based on on |Xk|^2 / T for only the
positive frequencies ##########
X_cont_half = X_cont[:N//2]

f_vec = np.arange(0, f_s, f_0)
f_vec_half = f_vec[:N//2]


plt.figure()
plt.plot(f_vec_half, np.abs(X_cont_half)**2 / T)

plt.xlabel(r'frequency [$yr^{-1}$]')
plt.ylabel(r'PSD [$mm^2$ yr]')

plt.title(r'Two-sided periodogram (only one-half shown)')
```



```
########## Compute and Plot the Periodogram based on |Xk|^2 / T ############


# frequency array (centered at f=0, and conform fftshift)
# for NFFT even, covers interval [-f_s/2,f_s/2)
# f_vec = np.concatenate((np.arange(-f_s/2, 0, f_0), np.arange(0, f_s/2, f_0)))

# for NFFT odd, use instead: f_vec=np.concatenate((np.arange(-f_s/2 + f_0/2, 0, f_0), np.arange(0, f_s/2, f_0))); covers interval (-f_s/2, f_s/2)
f_vec = np.concatenate((np.arange(-f_s / 2 + f_0 / 2, 0, f_0), np.arange(0, f_s / 2 , f_0)))

#|Xk|^2 / T periodogram (centered, with f=0 in the center), division by T, which is actual data record length
plt.figure()
plt.plot(f_vec, np.fft.fftshift((np.abs(X_cont))**2 / T))
plt.xlabel(r'frequency [$yr^{-1}$]')
plt.ylabel(r'PSD [$mm^2$ yr]')
plt.title(r'Two-sided periodogram')
```



```
############ Compute the Periodogram using scipy.signal library ############

fp, Xp = signal.periodogram(ehat, fs=f_s, window='boxcar', nfft=NFFT, re-
turn_onesided=False, scaling='density')
# basic periodogram, as a first, rough estimate for Power Spectral Density
(PSD) of signal x
# ehat : detrended observed sea-level height
# fs : sampling frequency [Hz]
# 'boxcar' : rectangular window on data
# NFFT : observed time series until length NFFT
# return_onesided=False : show two-sided spectrum with f=0 in the middle,
#                         covering interval [-f_s/2,f_/2) for NFFT even and (-
f_s/2,f_s/2) for NFFT odd

plt.figure()
plt.plot(np.fft.fftshift(fp), np.fft.fftshift(Xp))
plt.xlabel(r'frequency [$yr^{-1}$]')
plt.ylabel(r'PSD [$mm^2$ yr]')
plt.title(r'Two-sided periodogram computed with scipy.signal')
```



Identify the largest peak in the spectrum, what is the frequency, and can you come up with a physical explanation of this behaviour?

The largest peak is at $f = 0.996/year$, hence the annual cycle, related to summer and winter. There is also a peak at $f = 1.992/year$, hence the double frequency (related to a half year cycle), and this one typically shows up if the once per year periodic cycle is not a perfect harmonic (sine or cosine), but instead a bit distorted/skewed. Finally, there is also a large peak at $f = 1/T = 0.0362/year$, a long term effect, which implies a cycle with the duration of the entire data set, which here seems just a coincidence, that a full cycle occurs in $T=27.58\ years$.