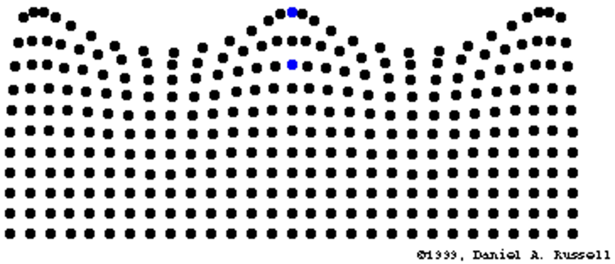
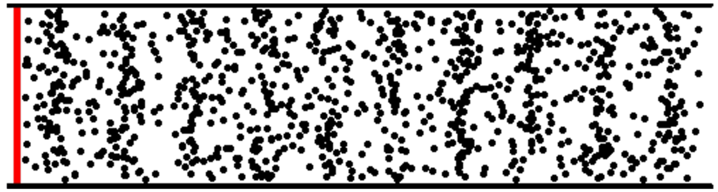
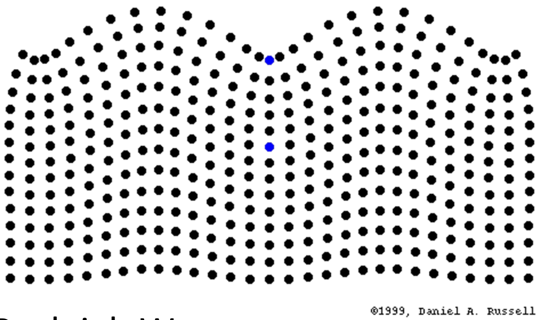


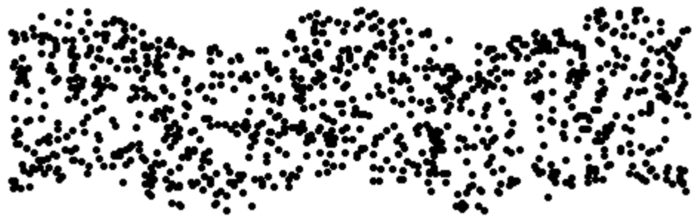
Love Waves


P Waves

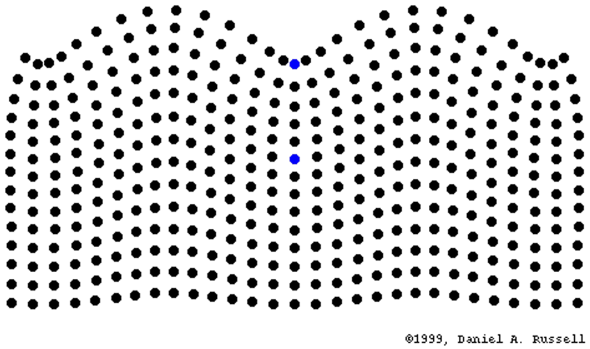
Lecture 4
Fourier Transform and Spectral Analysis



Rayleigh Waves


S Waves

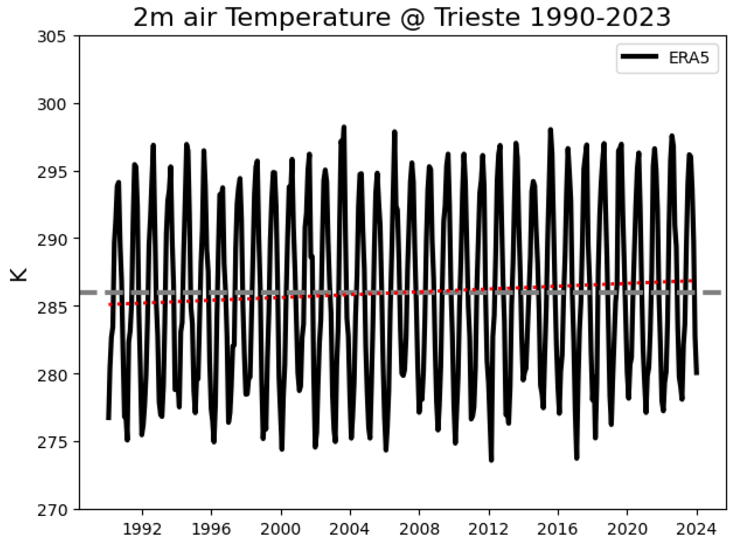
Fourier Transform



Rayleigh Waves (seismic Surface Saves)

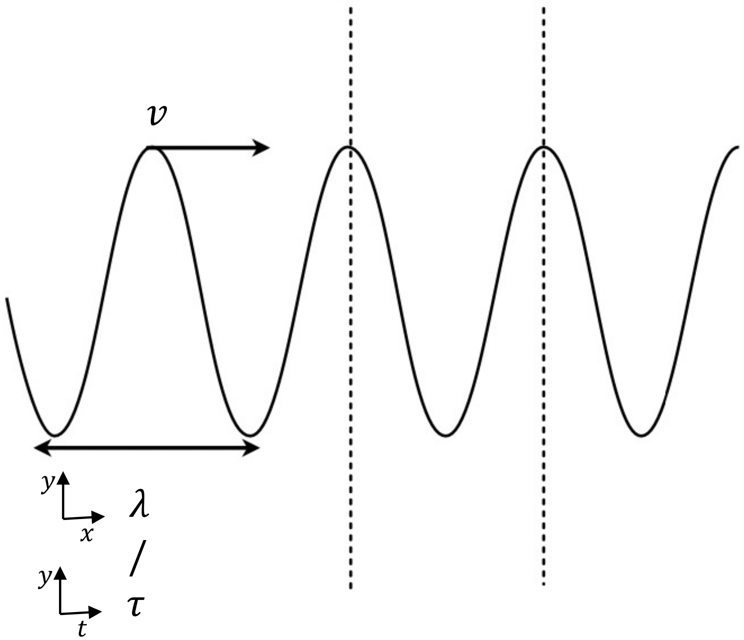
Atmospheric Gravity Waves (Internal waves)

Not only waves:



Fourier Transform

"It's a wave!"



Return period of the oscillation in time: $\tau\ (s)$ PERIOD
Return period of the oscillation in space: λ (m) WAVELENGTH
Velocity at which the wave travels : $v = \lambda/\tau$ PHASE SPEED

$$\omega = \frac{2\pi}{\tau}$$ (ANGULAR) FREQUENCY (1/s = Hz)

$$k = \frac{2\pi}{\lambda}$$ (ANGULAR) WAVENUMBER (1/m)

Figure 1.8 from C.J.Nappo book (Academic Press). Modified.

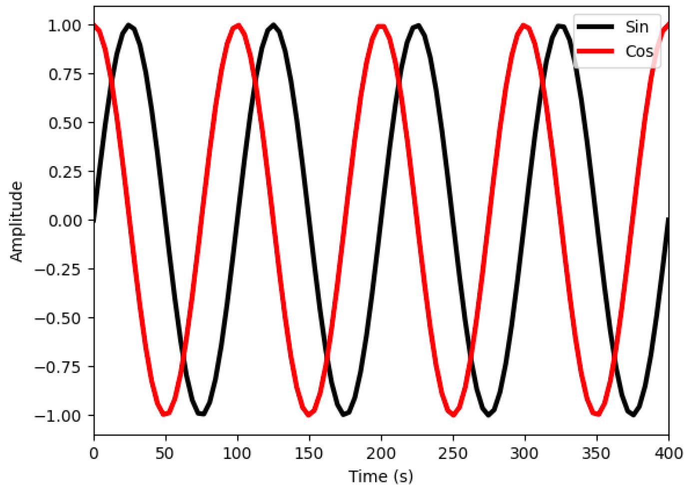
Fourier Transform

A generic wave function:

$$f(x,t) = f(x \mp vt)$$

$\xrightarrow{-}$

$\xleftarrow{+}$



A cosine* wave that travels in space x and time t with an amplitude A and a phase φ:

$$f(x,t) = A\,cos(kx - \omega t + \varphi)$$

Move to complex notation using the Euler's formula: $e^{ix} = \cos(x) + isin(x)$

$$\underbrace{f(x,t) = \Re e\; A\; e^{i(kx-\omega t+\varphi)}}_{SINUSOID}$$

(where we discard the imaginary part of the solution)

*NB: cosine and sine waves are both sinusoids , sinusoids are also called 'harmonics'

Fourier Transform

Fourier's theorem: **<u>a periodic function</u>** can be represented as a sum of sinusoids added together. Each sinusoid has a frequency that is an integer multiple of the fundamental frequency ν_0 (i.e. the frequency of the studied signal).

For a generic function that depends on time only, with a period τ, **<u>the Fourier series</u>** can be written as:

$$f(t) = \frac{C_0}{2} + C_1 \cos\left(\frac{2\pi}{\tau}t + \varphi_1\right) + C_2 \cos\left(\frac{2\pi}{\tau/2}t + \varphi_2\right) + \cdots$$

Where $C_{0,1,2\ldots}$ are constants.

Using the trigonometric addition formulas*, and $\nu_0 = \frac{1}{\tau}$, we can more conveniently write:

$$f(t) = \frac{C_0}{2} + \sum_{n=1}^{\infty} A_n \cos(2\pi n \nu_0 t) \quad + \sum_{n=1}^{\infty} B_n sin(2\pi n \nu_0 t) \quad (1)$$

A SUM OF COSINES AND SINES

where:

$A_n = C_n \cos(\varphi_n)$, $B_n = -C_n \sin(\varphi_n)$

$\boxed{* \cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)}$

Fourier Transform

The process by which the values of the C_0, A_n and B_n coefficients are determined is **known as Fourier Analysis**. This can be done by integrating both sides of (1) between 0 and τ to find:

$$C_0 = \frac{2}{\tau}\int_0^{\tau} f(t)dt$$

(Note thus that the first term of the Fourier series in (1) is the average value of the input function)

$$A_n = \frac{2}{\tau}\int_0^{\tau} f(t)\cos(2\pi n \nu_0 t)\,dt$$

(all other coefficients are used to determine the remaining terms in the series, i.e. the different frequencies that combine to form the studied signal)

$$B_n = \frac{2}{\tau}\int_0^{\tau} f(t)\sin(2\pi n \nu_0 t)\,dt$$

Fourier Transform

It is common to write the Fourier series in complex notation making use of the Euler's formula:

$$f(t) = \sum_{-\infty}^{+\infty} C_n e^{i2\pi n\nu_0 t} \qquad (2)$$

Where the complex coefficients C_n are:

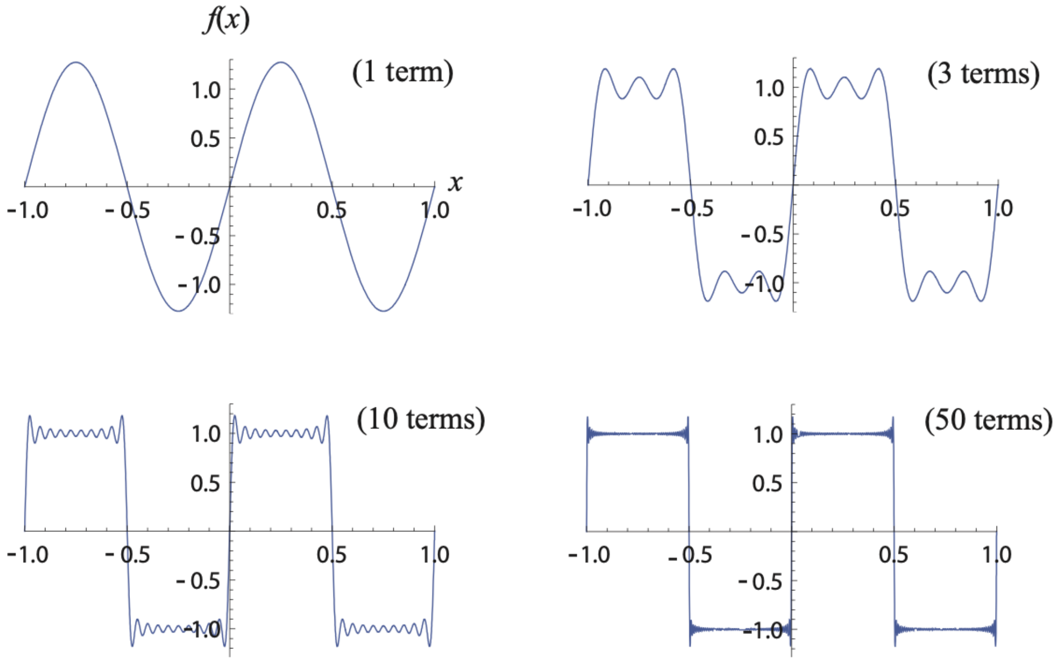
$$C_n = \frac{1}{\tau}\int_o^{\tau} f(x)e^{-i2\pi n\nu_0 t}\, dt \qquad (3)$$

Note this expression holds also for C_0

See for example https://scholar.harvard.edu/files/davidmorin/files/waves_fourier.pdf for full derivation.

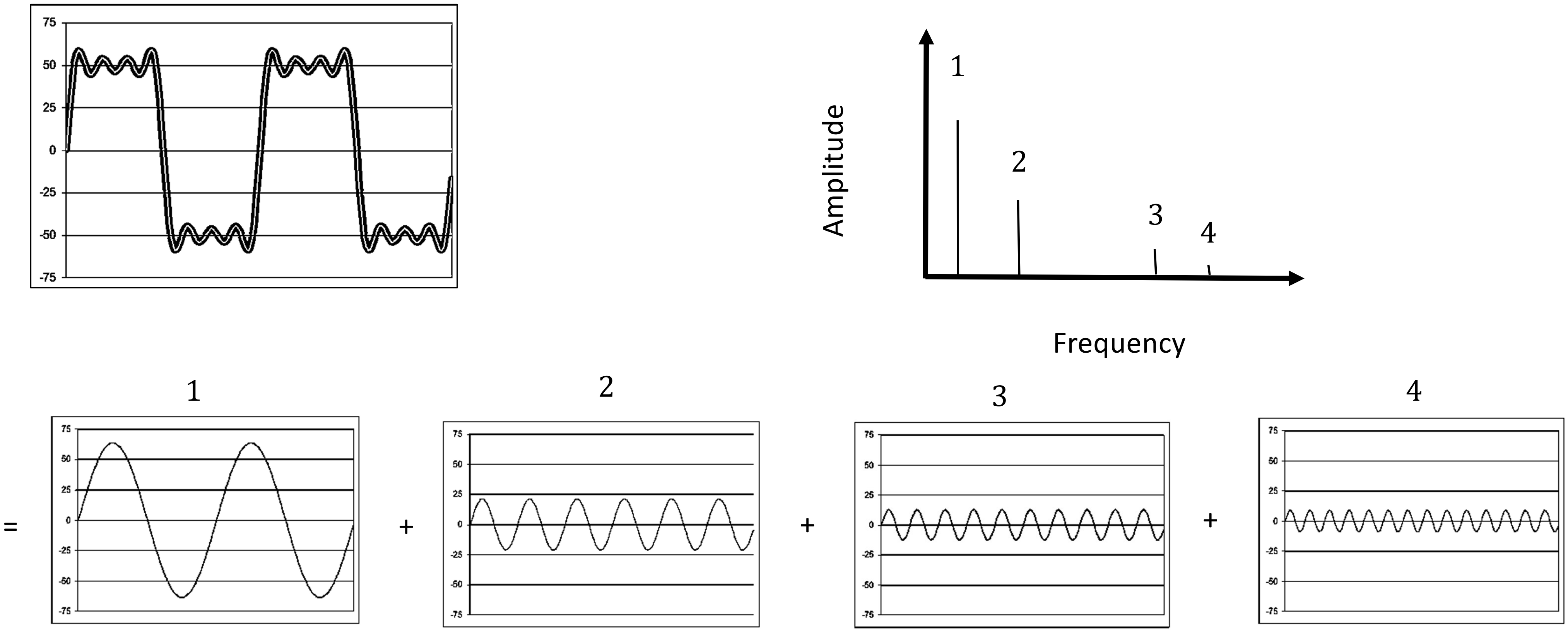
Fourier Transform

A few examples of periodic functions:



Fourier Transform

Harmonic decomposition of a square wave (with 4 terms) and example spectrum:



Fourier Transform

A function can be expanded to a Fourier series only if periodic. ***The majority of times we deal with signals that are non-periodic***, we can still apply the Fourier analysis to a aperiodic signal noting that:

$\tau \rightarrow \infty$ aperiodic signals can be thought as of periodic signals with an infinite period

In the limit $\tau \rightarrow \infty$ we can write (1) as:

$$f(t) = \int_{-\infty}^{\infty} A(\nu)\cos(2\pi\nu t)d\nu + \int_{-\infty}^{\infty} B(\nu)\sin(2\pi\nu t)d\nu$$

The Summation over all integer multiples of the fundamental frequency (ν_0) in the Fourier Series **becomes an Integral** because we sum across all possible frequencies from $-\infty$ to $+\infty$

Using the complex notation (as done for (2) and (3)), and $\omega = 2\pi\nu$, we can write:

$$f(t) = \int_{-\infty}^{+\infty} C_\omega e^{i\omega t}\, d\omega$$

Where:

$$\underbrace{C(\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\, dt}_{FOURIER\ TRANSFORM\ (often:\ F(\omega))}$$

Fourier Transform

Fourier transform: **<u>Any function</u>** (not necessarily periodic) can be written as a continuous integral of sines an cosines or exponential functions.

The Fourier Transform of $f(t)$ in time is:

$$F(\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,dt$$

The Inverse Fourier Transform in time is:

$$f(t) = \int_{-\infty}^{+\infty} F(\omega)e^{i\omega t}\,d\omega$$

The Fourier Transform of $f(x)$ in space is (where $k = 2\pi/\lambda$):

$$F(k) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} f(x)e^{-ikx}\,dx$$

The Inverse Fourier Transform in space is:

$$f(x) = \int_{-\infty}^{+\infty} F(k)e^{ikx}\,dk$$

The Double (2D) Fourier Transform in space and time is:

$$F(k,\omega) = \frac{1}{2\pi}\iint_{-\infty}^{+\infty} f(x,t)e^{-i(kx+\omega t)}\,dxdt$$

and its inverse:

$$f(x,t) = \iint_{-\infty}^{+\infty} F(k,\omega)e^{i(kx+\omega t)}\,dkd\omega$$

Fourier Transform

The Double (2D) Fourier Transform in space and time is:

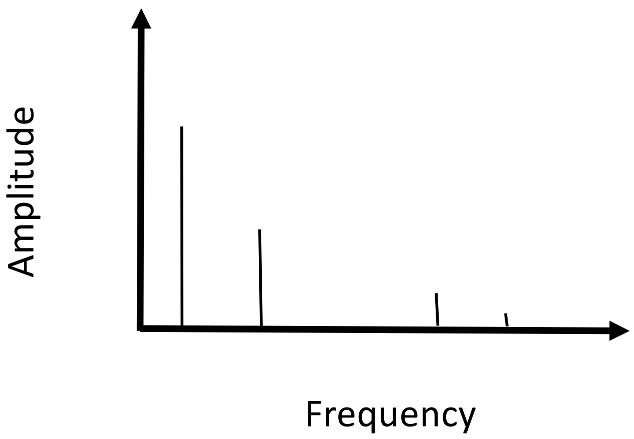
$$F(k,\omega) = \frac{1}{2\pi}\iint_{-\infty}^{+\infty} f(x,t)e^{-i(kx+\omega t)}\,dxdt$$

$$f(x,t) \xrightarrow[\textit{To the frequency domain}]{\text{1D}} F(x,\omega) \xrightarrow[\textit{To the wavenumber domain}]{\text{1D}} F(k,\omega)$$

SEPARABILITY OF THE 2D FT

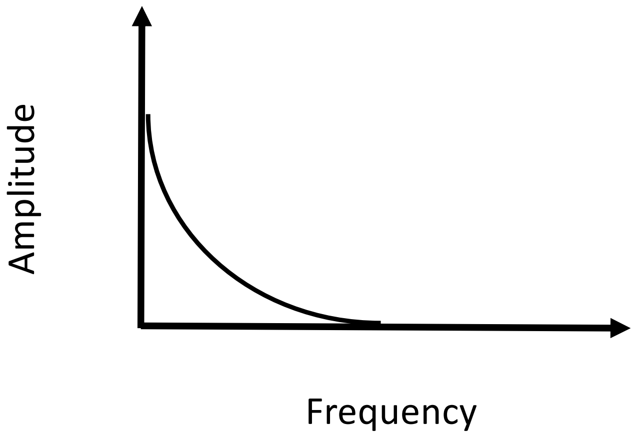
Fourier Transform

Discrete Spectrum for a periodic function:



The signal is the sum of harmonics with a frequency multiple of a fundamental frequency. The spectrum is discrete, a ‘line spectrum’.

Continuous Spectrum for an aperiodic function:



The signal can be represented as the sum of harmonics with all frequencies. The spectrum is continuous. Note as $\tau \to \infty$ frequencies get closer to each other, the function looks more and more like a ‘pulse function’.

Spectral Analysis

Example aperiodic signal and its spectrum:

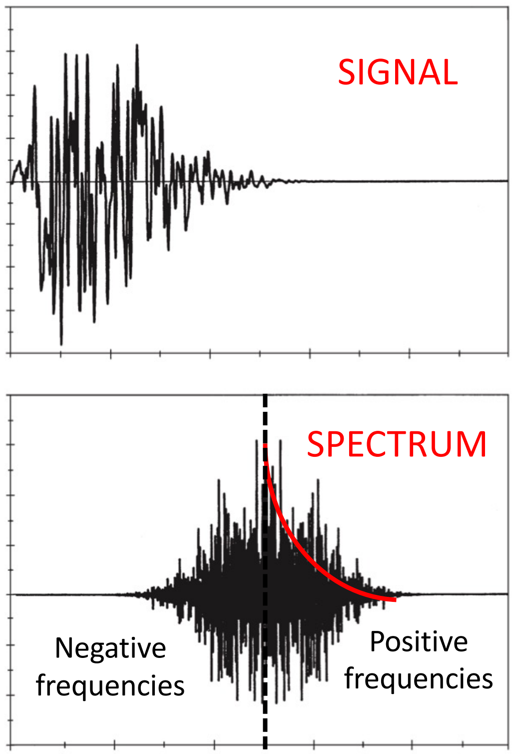


Fig. 1.2. The spectrum of a crash: all frequencies are present.

From: *"A Student's Guide to Fourier Transforms"* by J.F. James

The Power Spectrum for each frequency is defined as the square of the module of the Fourier Transform:

$$S(\omega) = F(\omega)F^*(\omega) = |F(\omega)|^2$$

$S(\omega)$: *power spectrum of the signal*

$F(\omega)$: *Fourier Transform of the signal*

$F^*(\omega)$: *Complex conjugate of the FT*

Let's consider a complex number c :

$c = a + ib$

its conjugate:

$c^* = a - ib$

its module: $|c| = \sqrt{a^2 + b^2}$

$$c \cdot c^* = a^2 - aib + aib - i^2b^2 = a^2 + b^2 = |c|^2$$

Spectral Analysis

- By looking at $S(\omega)$ we can identify what portion of the spectrum (i.e. what frequencies/wavenumbers) carry the most energy, which is the majority of times what we are interested in!

- Note that, because of how it is defined (i.e. complex numbers with a real ($cosines$) and an imaginary ($sines$) part) , the Fourier transform of a given signal will always return a sum of positive f^+ and f^- negative frequencies*:
  each f^+ , f^- pair represents two waves moving in opposite direction

(*remember the integral goes from $-\infty$ to $+\infty$ so to reconstruct our original signal we have to consider *all* frequencies).

- For purely real signals, the information carried by the negative frequencies is redundant (i.e. negative frequencies are simply the complex conjugates of positive ones). **The spectrum is symmetric around the zero frequency***, so we can discard negative frequency when showing the spectrum and analyzing results.
- ***this property is known as the symmetry of the Fourier Transform; remember the $\cos$ function is an even function (i.e. symmetrical) and the Fourier transform of a real signals is made up of cosines only.**

Spectral Analysis

A=numpy.fft.fft(a) https://numpy.org/doc/stable/reference/routines.fft.html

Python finds the Fourier coefficients and returns an array containing the transformed signal:
A[0]: the first element in the array is the zero frequency term, note in python this is the sum of all the signal because of how the DFT is defined (see documentation).
A[1:n/2]: contains the positive-frequency terms,
A[n/2+1:]: contains the negative-frequency terms.

numpy.fft.ifft() : Compute the inverse discrete Fourier Transform

numpy.fft.fft2() : Compute the 2-dimensional discrete Fourier Transform

numpy.fft.ifft2() : Compute the inverse 2-dimensional discrete Fourier Transform

Spectral Analysis

A=numpy.fft.fft(a) https://numpy.org/doc/stable/reference/routines.fft.html

np.fft.fftfreq(): returns the frequencies of transformed signal (i.e. our x axis in a 1d power spectrum).

np.fft.fftshift(): shifts the transformed signal and its frequencies to put the zero-frequency components in the middle.

To compute the Power Spectrum of a signal ‘a’:

```
A=numpy.fft.fft(a)
S=np.abs(A)**2
```

Spectral Analysis

Note that `python` compute the Discrete Fourier Transform (DFT), as per documentation. Discretizing the Fourier Transform is needed as we deal with signals that were sampled with a given sampling frequency (f_s).

Note also that the sampling frequency sets *the highest frequency* that we will be able to detect in the sampled signal, this is called the *Nyquist frequency* f_{Ny} :

$$f_{Ny} = \frac{f_s}{2}$$

It will not be possible to detect frequencies that are higher than half the sampling frequency.

Spectral Analysis

Harmonic decomposition of a square wave (with 4 terms):

τ_s

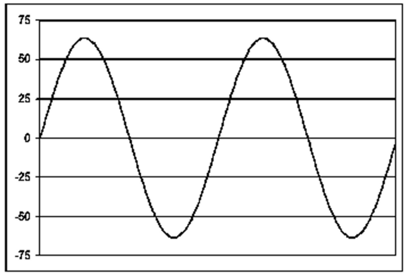
$f_s = 2f_{Ny}$

For gridded datasets, this is equivalent to say that one always need at least* two grid-cells to represent a wave (*in practice many more are needed).

The highest frequency in the signal:

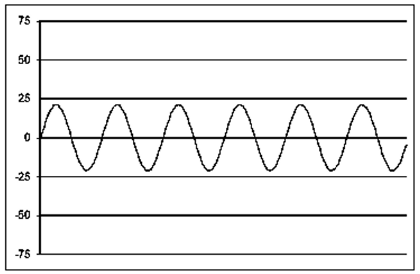
=

1



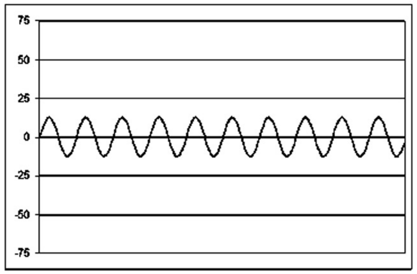
+

2



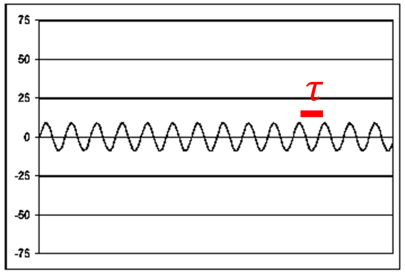
+

3



+

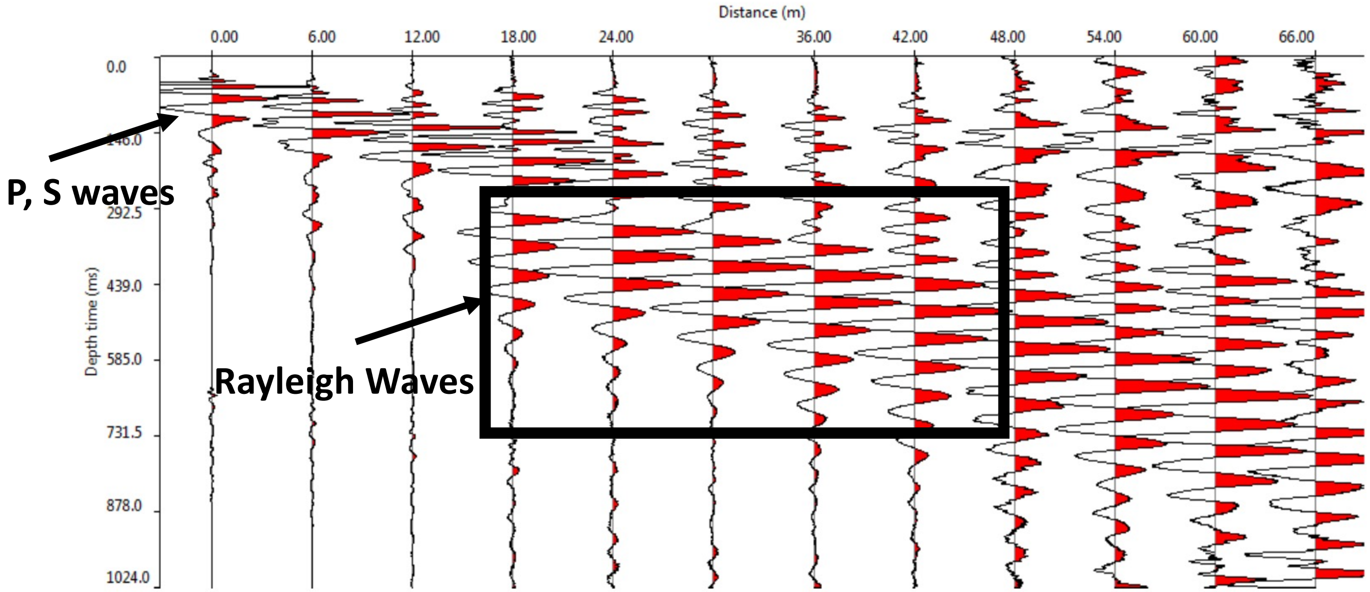
4 $f_{Ny} = 1/\tau$



Spectral Analysis

Example of Fourier Transform applications in seismic: Surface Waves Analysis

Geophones: ★ .... ★



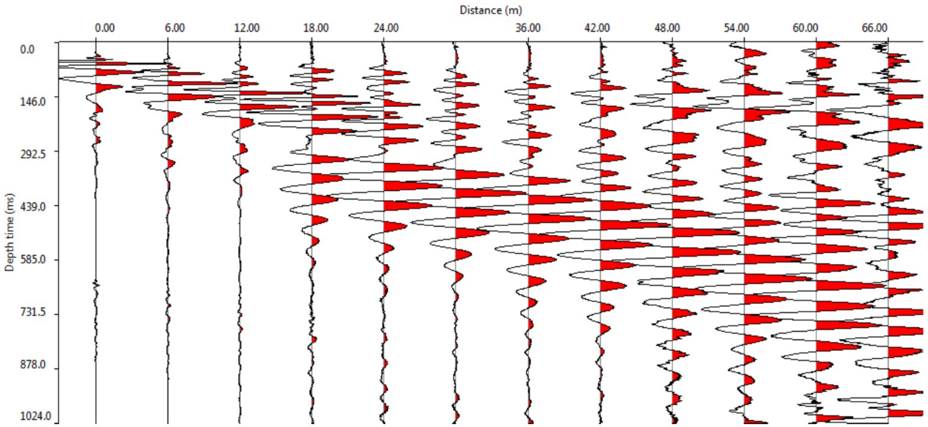
MASW is a seismic method that measures the shear-wave velocity distribution. It can be used to: determine the depth to bedrock, map sinkholes, assessing earthquake resilience ("seismic characterization").

Last Traces are too noisy
(signal to noise ratio is too low, waves are more effectively attenuated)

Seismogram from **MASW (Multichannel Analysis of Surface Waves)** acquisition. Data processed using the 'SWAN' software.

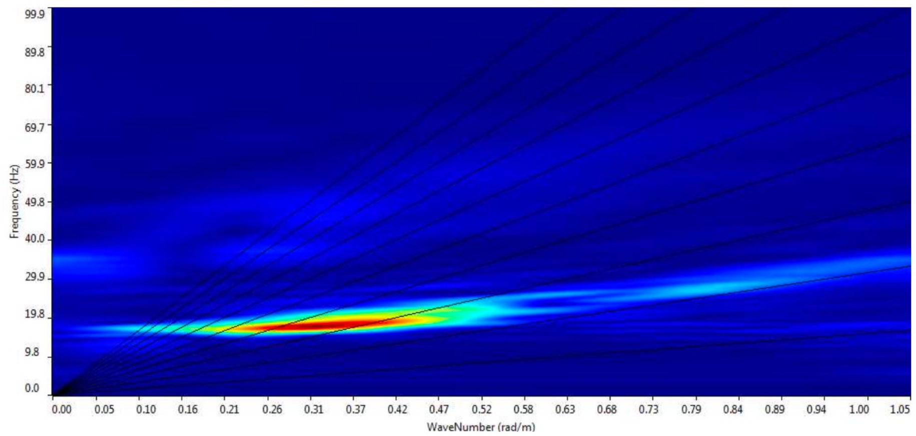
Spectral Analysis

From Time/Space domain ….



2D FFT →

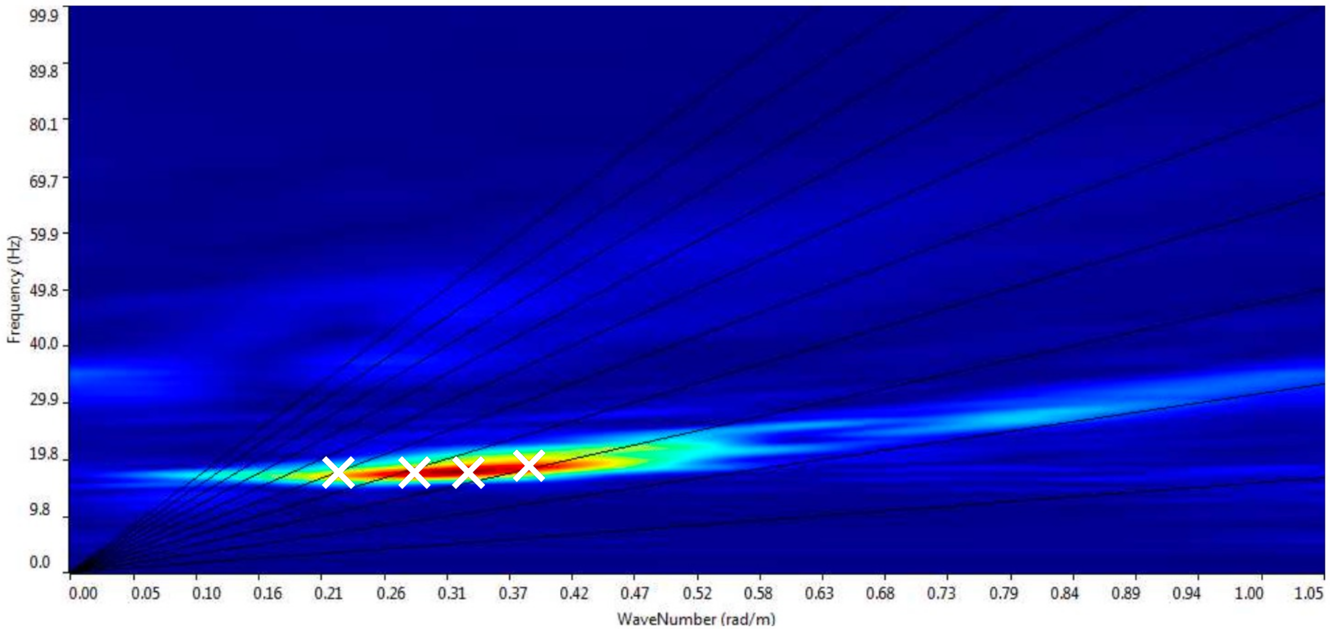
… to Frequency/Wavenumber domain



$$f(x,t) \xrightarrow[\text{To the frequency domain}]{\text{1D}} F(x,\omega) \xrightarrow[\text{To the wavenumber domain}]{\text{1D}} F(k,\omega)$$

Spectral Analysis

What information can we get from the (f, k) spectrum?

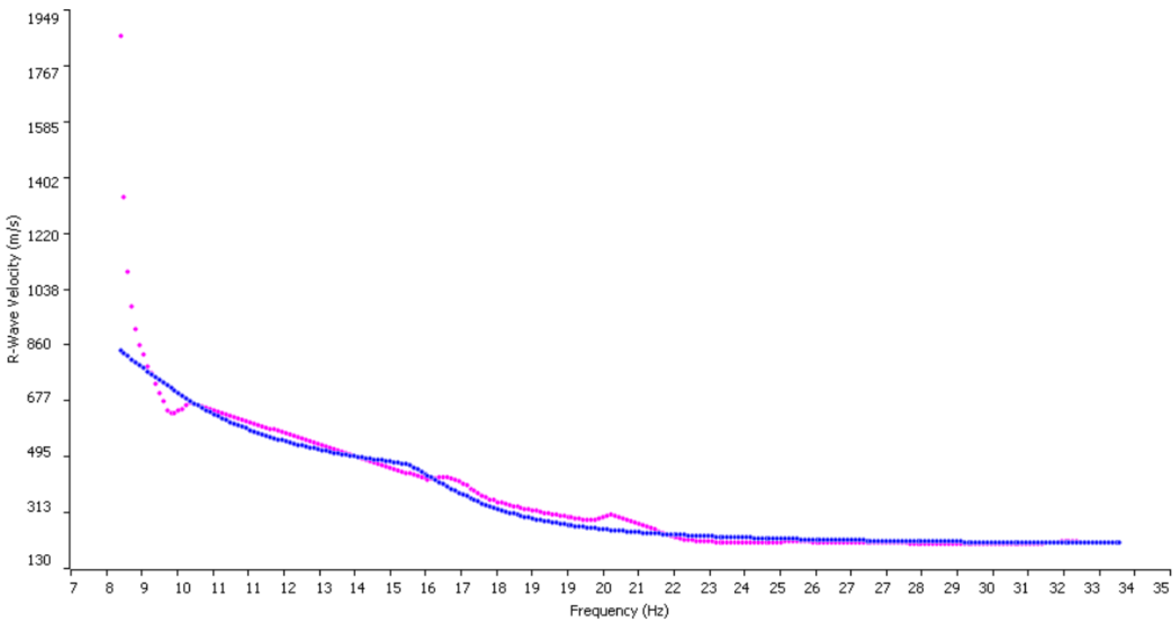


- Pick the f, k pairs with the most energy (we are after Rayleigh waves, carrying the most energy in the seismogram).
- Use them to estimate the phase velocity (V_f) of the Rayleigh waves:

$$V_f = \frac{f}{k}$$

Spectral Analysis

The Dispersion Curve:



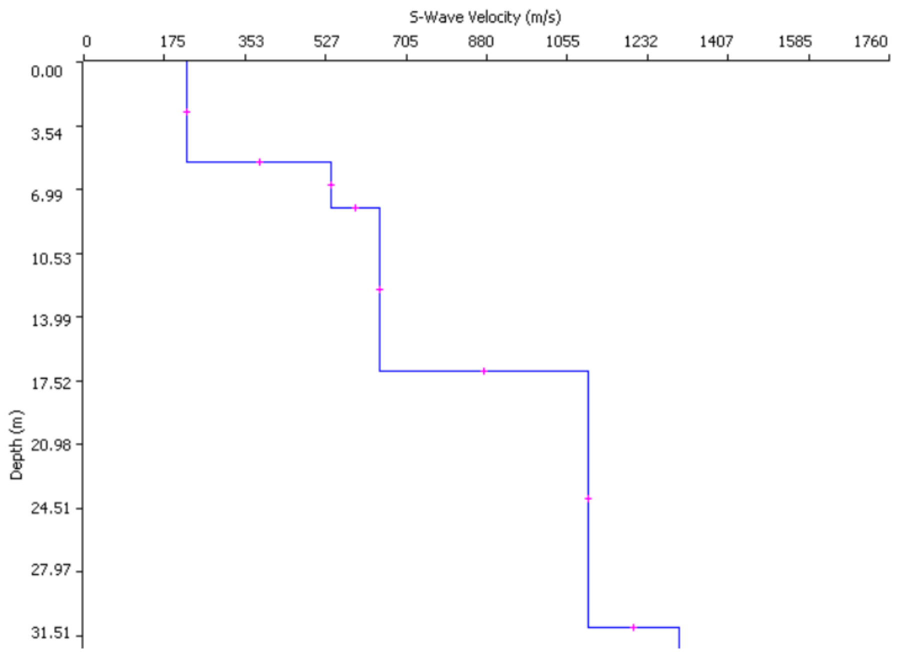
- Pick the f, k pairs with the most energy (we are after Rayleigh waves, carrying the most energy in the seismogram).
- Use them to estimate the phase velocity (V_f) of the Rayleigh waves:

$$V_f = \frac{f}{k}$$

- Plot V_f as a function of f to get the Dispersion Curve.
- The Experimental Dispersion Curve (pink line) that we obtained from our data is fitted with a theoretical one (blue line)

Spectral Analysis

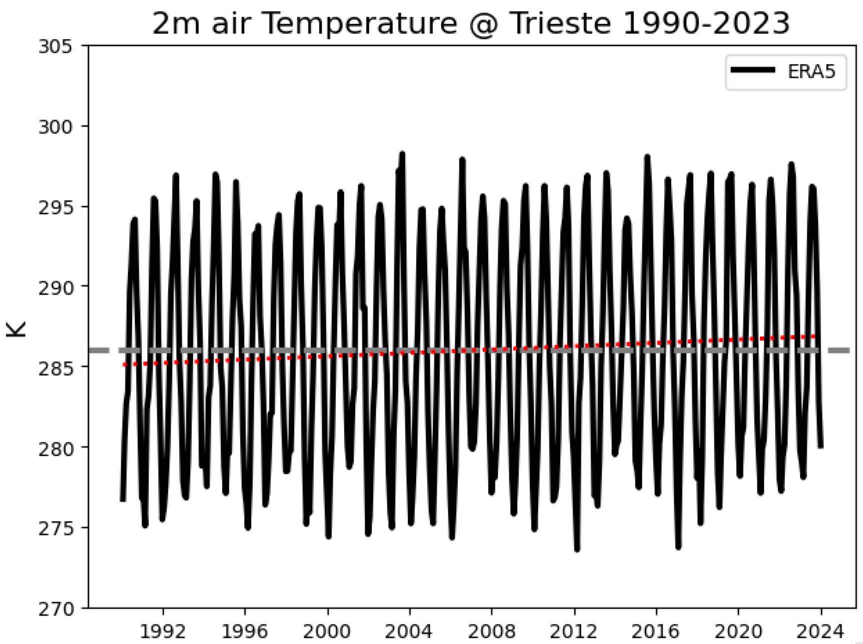
A stratigraphic model for the terrain based on the best match of the Experimental Dispersion Curve:



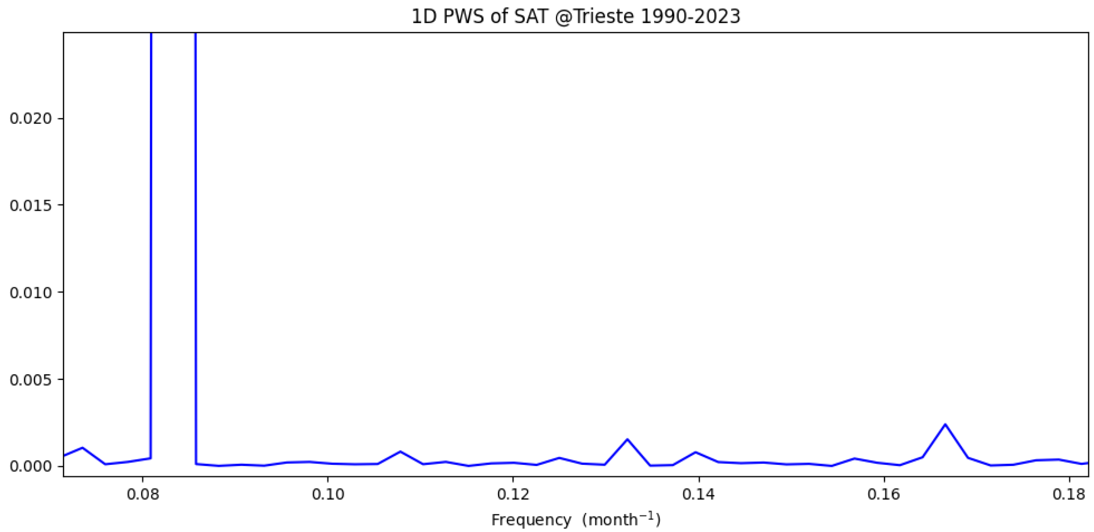
- When a best matching theoretical curve is found, the corresponding model of terrain (i.e. numbers of layers and thickness) is assumed to be a close representation of reality.

Spectral Analysis

Example of Fourier Transform applications in Climate science: dominant frequencies in time-series (1D Power Spectrum)



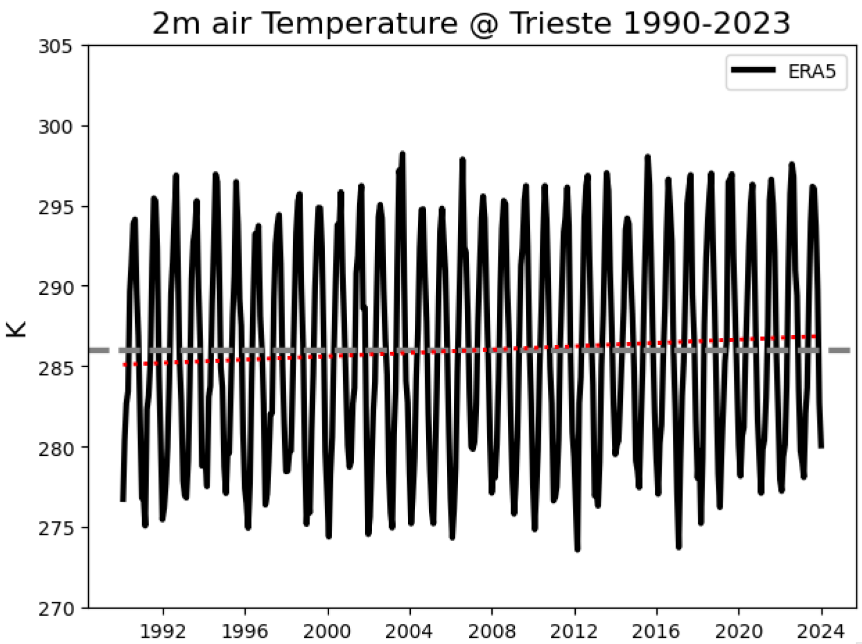
Normalized 1D Power Spectrum (note we show only positive f)



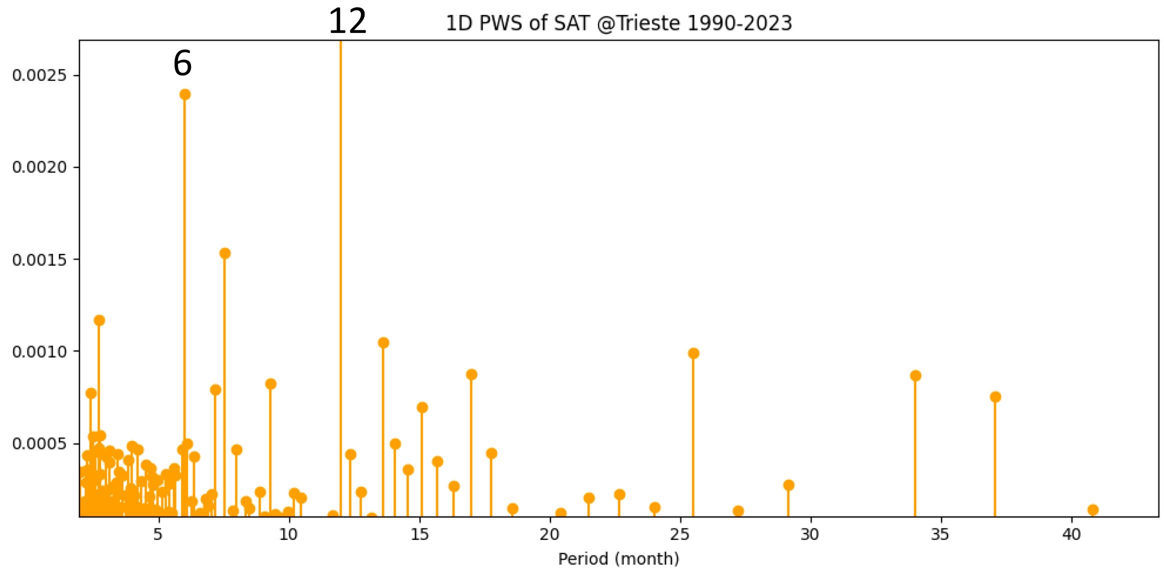
Peak at f= 0.0836 ($month^{-1}$), t=1/0.0836 ≃ 12 (month)

Spectral Analysis

Example of Fourier Transform applications in Climate science: dominant frequencies in time-series (1D Power Spectrum)

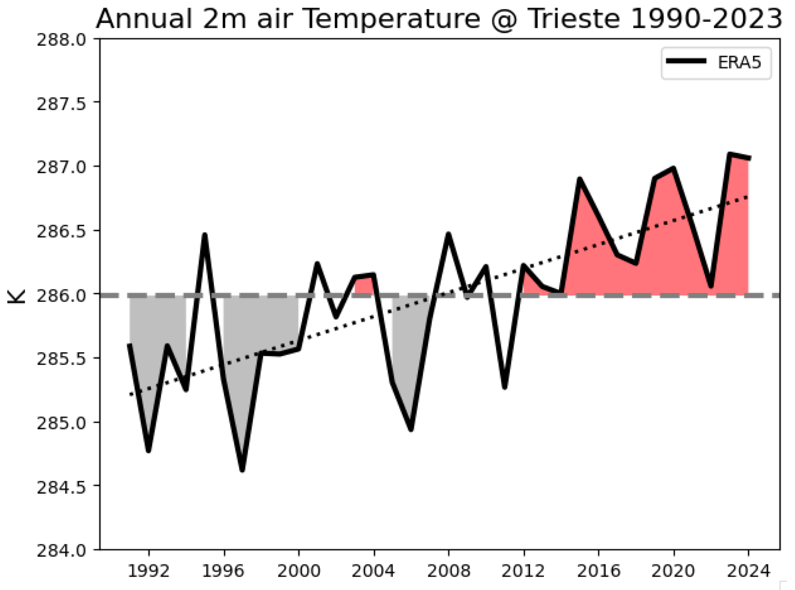


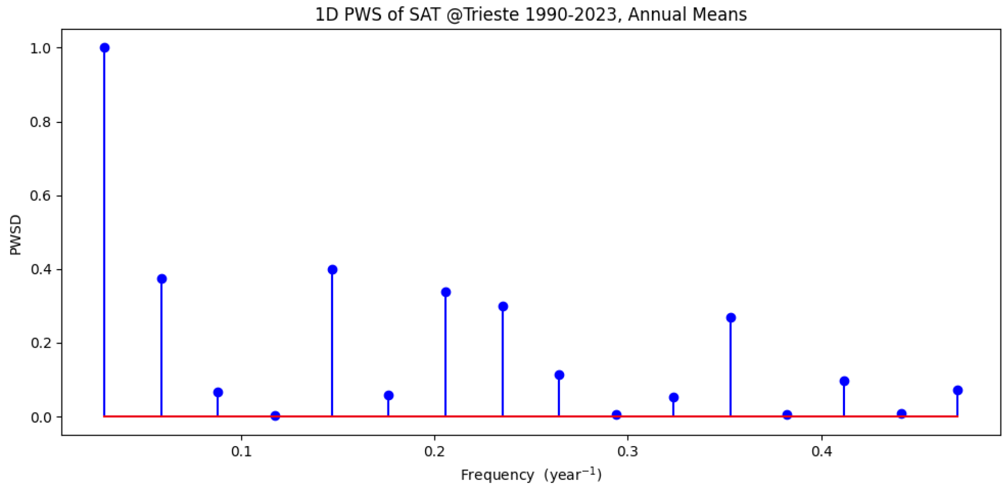
Normalized 1D Power Spectrum

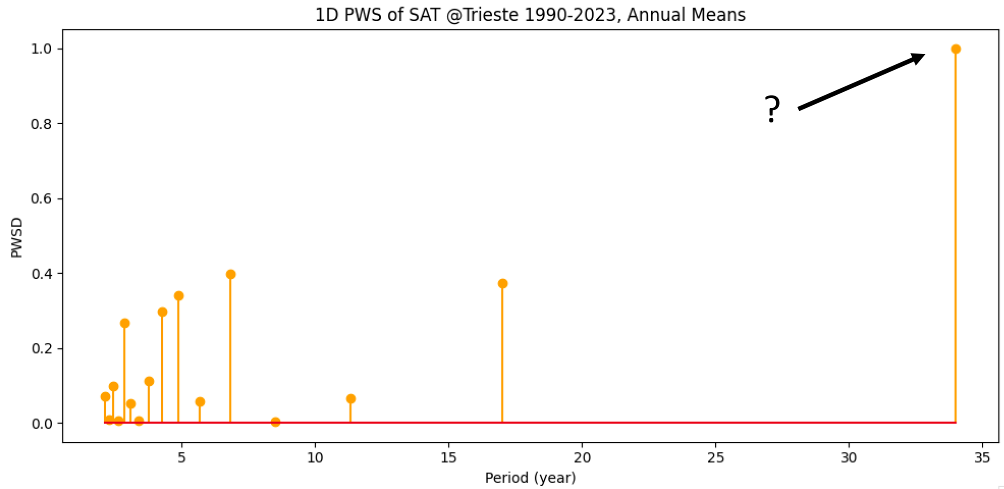


Peak at f= 0.0836 ($month^{-1}$), t=1/0.0836 ≃ 12 (month)

Spectral Analysis



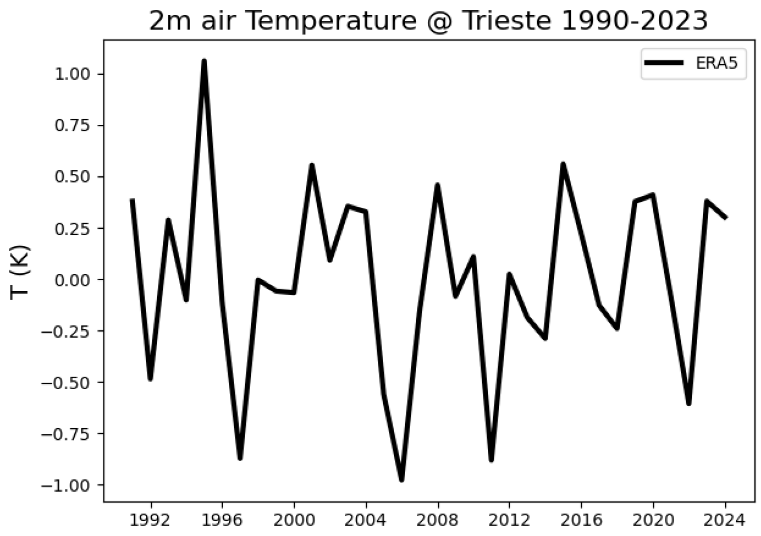


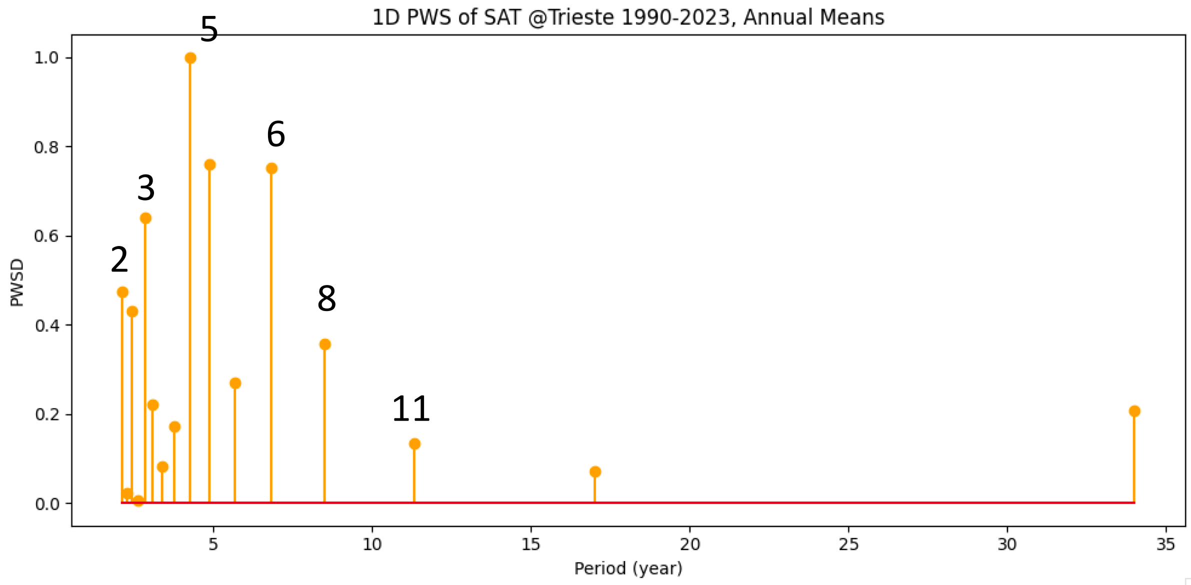


The largest peak in the spectrum has now a period of 34 years, this is the length of our input time-series.
It is an ‘artefact’ of the strong positive trend!

Spectral Analysis

Always detrend time-series before applying Fourier Transform!



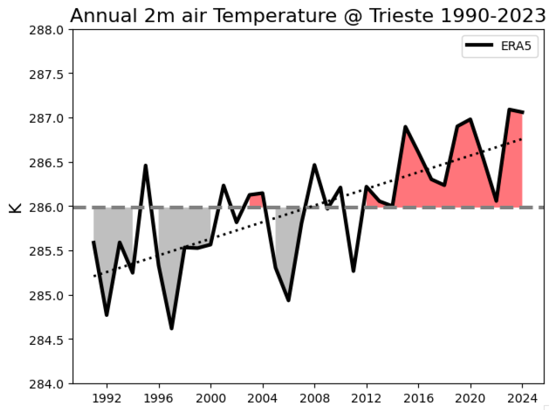


Detrended time-series (left) and its 1D PWS (right).

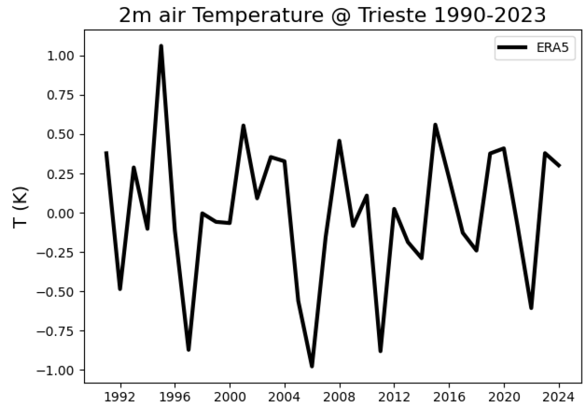
Spectral Analysis

To detrend a time-series remove the best fitting straight-line:

```
#Load txt Trieste SAT time-series
data_in=np.loadtxt('ERA5_2m_SAT_TS_1990_2023.txt', usecols=2)

#compute annual mean
data_in=compute_annual(data_in)

#find best fitting line using linear regression
time=np.arange(0,len(data_in),1)
slope, intercept, r_value, p_value, std_err
=stats.linregress(time,  data_in)
#define regression line
line_fit=slope*time +intercept
#detrend data by subtracting the regression line
data_in=data_in-line_fit

#plot detrended time-series
plot_tseries(data_in, label=' ', color='black', linestyle='-
', xlabel=' ', ylabel='T (K)', title='2m air Temperature @
Trieste 1990-2023 Detrended', date=True)
```



data_in=data_in-line_fit



Exercises

Example of how to use numpy.fft.fft() to compute a normalized power spectrum:

```
from numpy.fft import *

def FFT_1D(data_in):

 ##Call function to compute the 1D FFT
 n=len(data_in)
 FT1D=fft(data_in, n)
 #print (FT1D)

 ##Call function to compute the associated frequencies
 dt=1. #this is our sampling rate = 1 year, 1 month etc..
 nk=fftfreq(len(FT1D), dt) # Natural frequencies associated to each Fourier coefficient
 #print (nk)

 ##Call function to shift the frequencies so that the zero frequency is in the middle of the
array
 FT1D = fftshift(FT1D)
 nk = fftshift(nk)
```

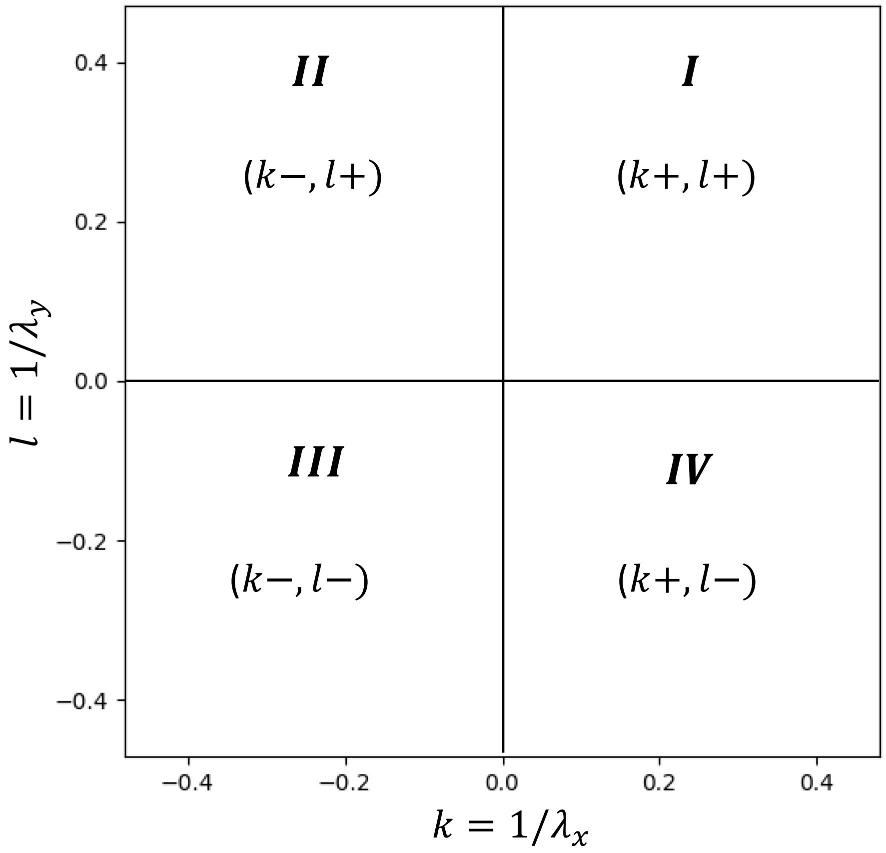
Exercises

Example of how to use numpy.fft.fft() to compute a normalized power spectrum:

```
##Now select only positive (>0) frequencies. Note in this way we also exclude from the final
array the zero-frequency term.

 positives= np.where(nk>0)
 FT1D=FT1D[positives] #take only positive freq (FFT for real input is symmetric)
 nk =nk[positives]


 ##Compute the power spectrum
 Spec_1D= np.absolute(FT1D)**2

 ##Find its maximum value
 maximum=np.amax(Spec_1D)
 ##Normalize the power spectrum relative to its maximum value
 Spec_1D_norm=Spec_1D/maximum



 return(Spec_1D_norm, nk)
```

Spectral Analysis

Example of a 2D Fourier Transform: getting information about directionality



Since the Fourier transform of a purely real signal **is symmetric**, in a 2D power spectrum all the information is contained in **the first two quadrants of the (k, l) plane.**

The third ($k < 0$, $l < 0$) and fourth ($k > 0$, $l < 0$) quadrants are just mirrored images of the first ($k > 0$, $l > 0$) and second ($k < 0$, $l > 0$) quadrants.

$$(k+, l+) = (k-, l-)$$

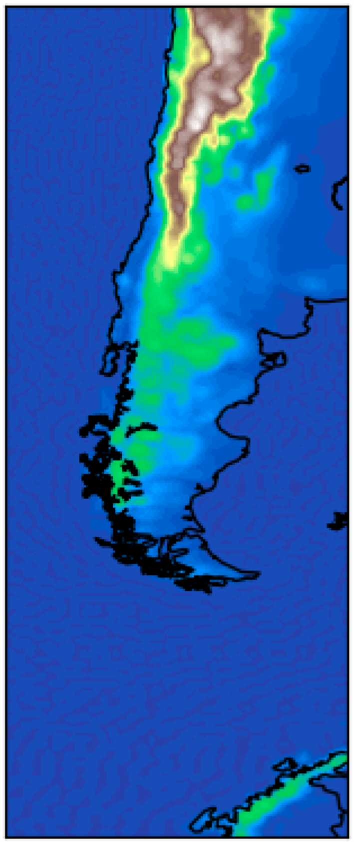
$$(k-, l+) = (k+, l-)$$

The first two quadrants are enough to give us information about the directionality of the data, i.e. in what direction the most energetic wavenumbers lie?

Spectral Analysis

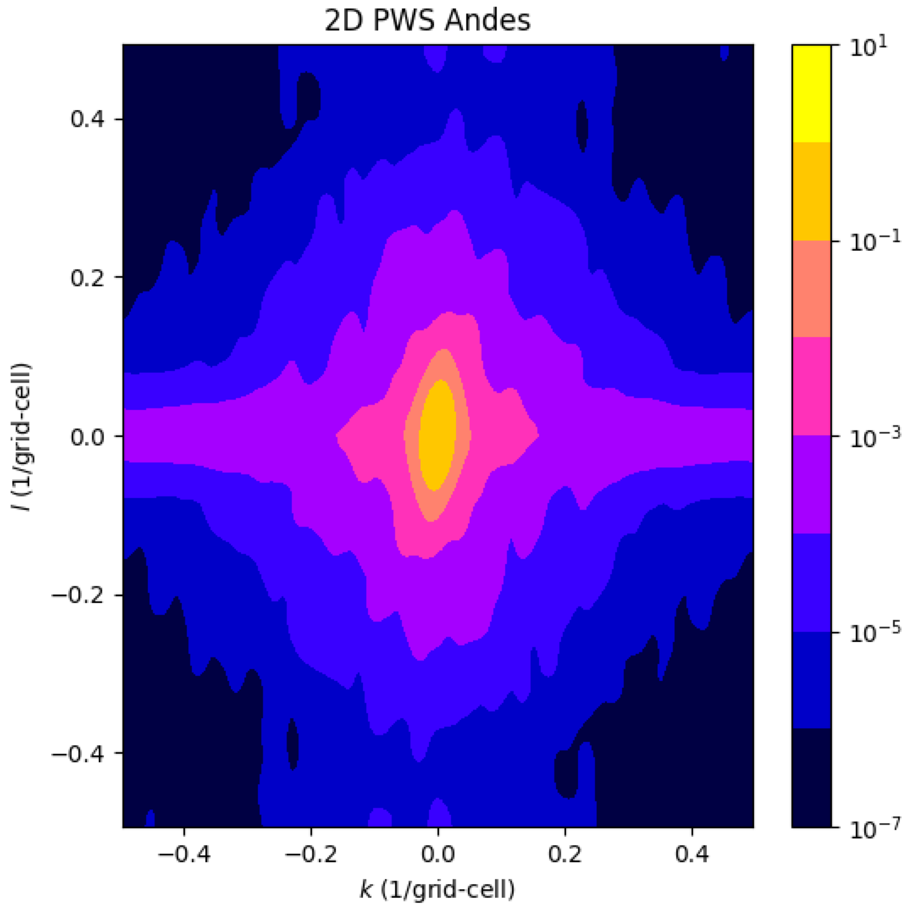
Example of a 2D Fourier Transform: getting information about directionality

Andes



2D FFT

→

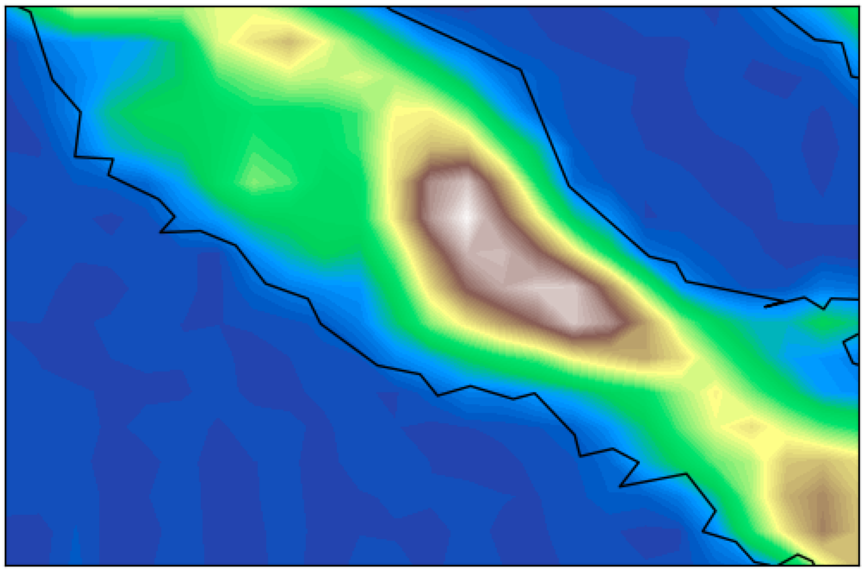


South/North direction of the main orography

Spectral Analysis

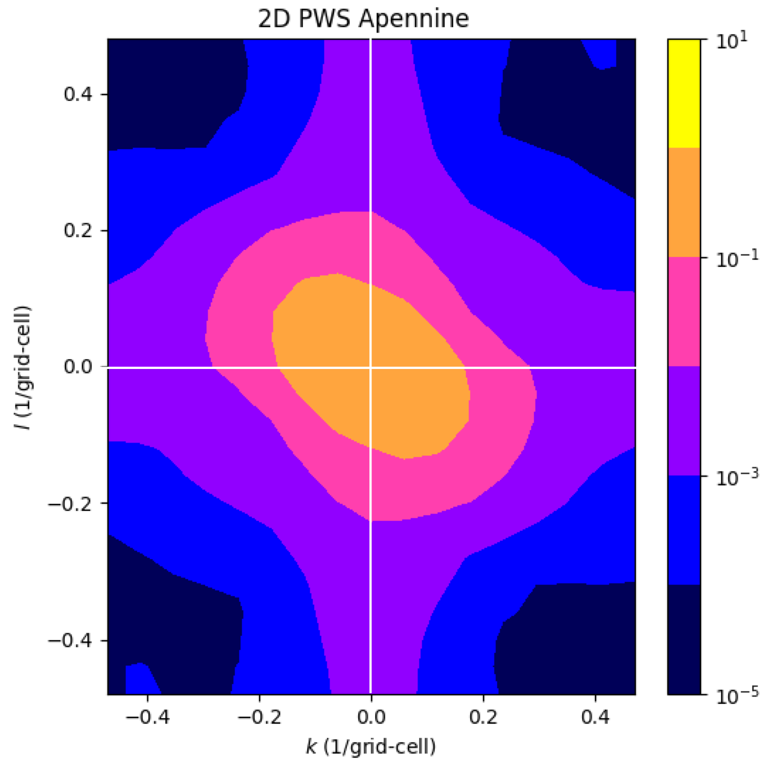
Example of a 2D Fourier Transform: getting information about directionality

Apennine



2D FFT

→

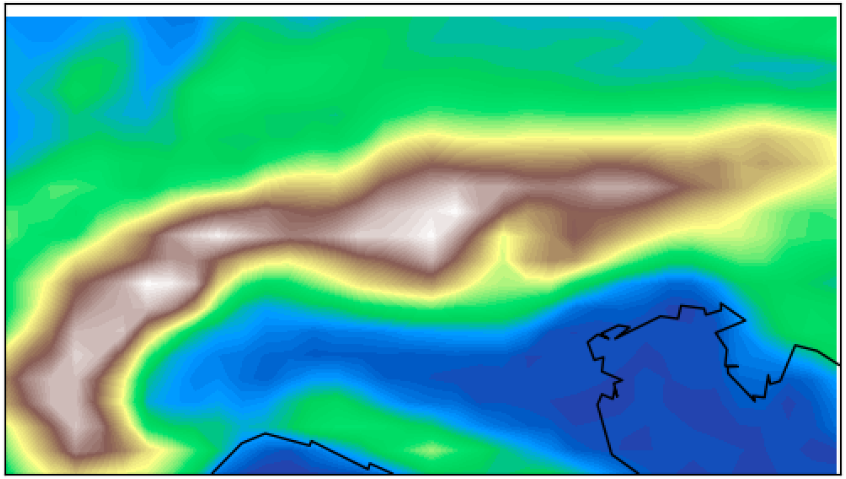


NW/SE direction of the main orography

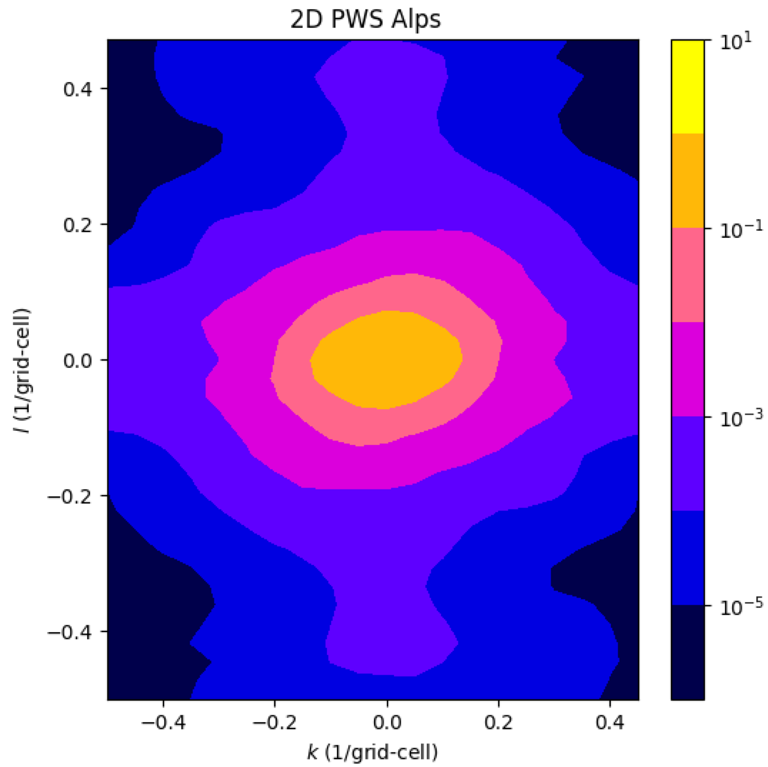
Spectral Analysis

Example of a 2D Fourier Transform: getting information about directionality

Alpes



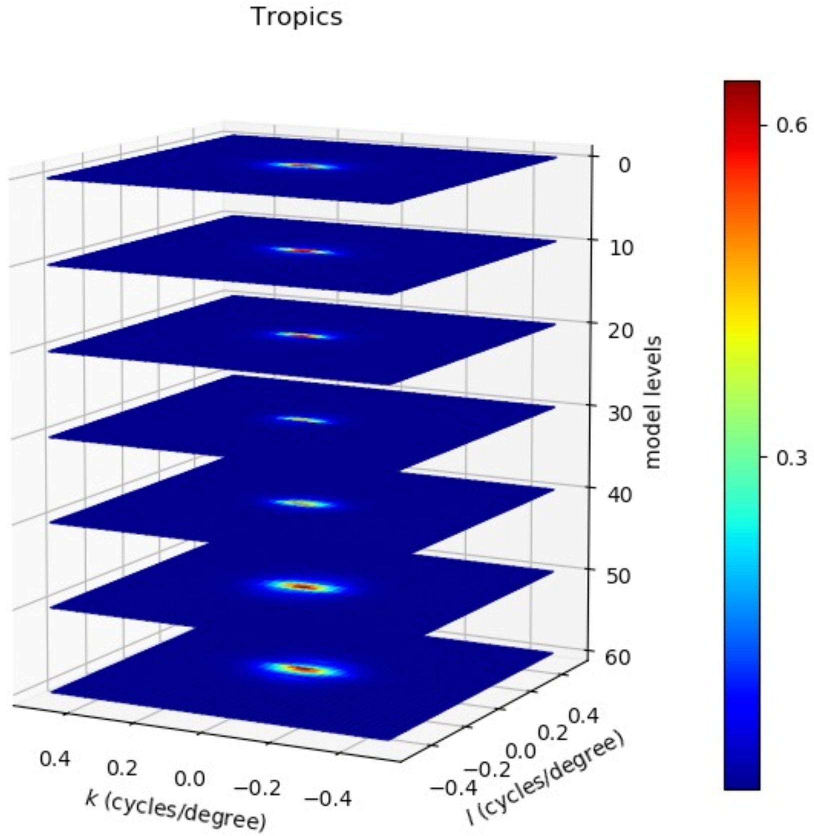
$\xrightarrow{2D\ FFT}$

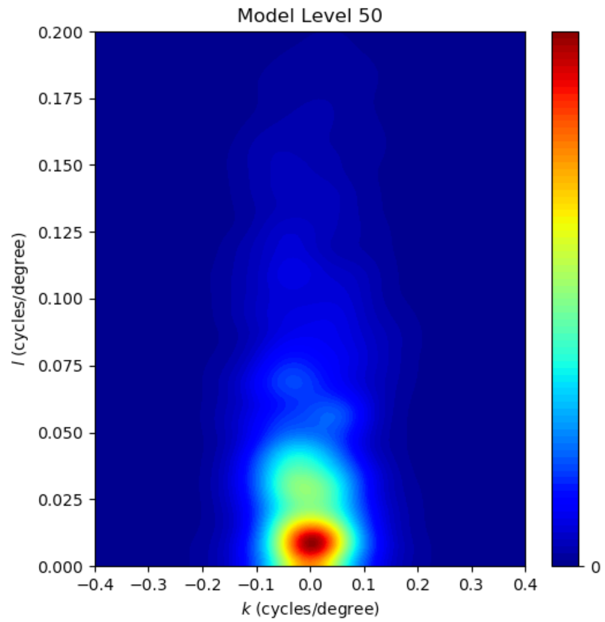


East/West direction of the main orography

Spectral Analysis

Example of a 2D Fourier Transform: dissipation of signal with height





Exercises

Provided dataset: 100 years of global SST data from a Preindustrial control simulation (CMIP6 simulations) run using the UKESM model. File: `PI_CMIP6_HadGEM3_100yrs.nc`

Aim: study the dominant frequency of oscillation in three major ocean basins (North Atlantic, North Pacific and Southern Ocean) under constant preindustrial forcing (i.e. natural variability).

TASKS:

1. Using the provided dataset, extract a regional area of interest. You can pick one of the following three ocean basins (NB: code for this is provided):

North Atlantic: [0-60N], [80W-0] **NA**
North Pacific: [20-70N], [240W-110W] **NP**
Southern Ocean: [50-70S], [0-360] **SO**

2. The frequency of the provided dataset is monthly. First, compute annual means on the data and afterwards compute an area-weighted spatial mean, so to obtain a time-series for the region of interest (NB: code for this is provided).

3. Using `stats.linregress` and following the example provided at slide 29, detrend your timeseries (despite this is constant forcing simulation and strong trends are not expected, it is always good practice to detrend a timeseries before applying an FFT).

Exercises

Example script: `DA_exercises_4.py`

4. Plot your annual timeseries and your detrended timeseries.

5. Write a function to compute the 1D FFT and the 1D Normalized Power Spectrum of the detrended timeseries using the **numpy.fft.fft(), np.fft.fftfreq(), np.fft.fftshift()** packages (see example provided at slides 30-31).

6. Plot the Normalized 1D Power Spectrum and identify what are the main frequencies in your time-series. What is the time-scale of the largest mode of SST variability? (annual, interannual, decadal, multidecadal, etc.) Remember you can plot your Power Spectrum as a function of the period $(1/f)$ to help the interpretation of results. (NB: code for this is provided).

Exercises

Example script: **DA_exercises_4.py**

```
#Load Netcdf dataset
SST=iris.load_cube('PI_CMIP6_HadGEM3_100yrs.nc')
print (SST)


#Extract Ocean Basin
where='SO'   #NA= North Atlantic, NP= North Pacific , SO= Southern Ocean
SST=extract_area(SST, where=where)


#Compute area-weighted annual time series
SST=area_weighted(SST)
print (SST)


#Plot SST time-series
plot_tseries(SST.data, label='PI Control Run', color='black', linestyle='-', xlabel='Years', ylabel='SST
(degC)', title='100 years of SST variability, {where}'.format(where=where), date=False)
```

Exercises

Example script: **DA_exercises_4.py**

```
#Detrend the time-series
SST=my_function_to_detrend(SST)


#Plot the detrended time-series
plot_tseries(SST.data, label='PI Control Run', color='black', linestyle='-', xlabel='Years', ylabel='SST (degC)', title='100 years of SST variability detrended', date=False)


#Call functions to compute 1D Power Spectrum (PWS)
S, nk=my_function_to_compute_FFT_1D(SST.data)


#Function to plot PWS. It takes as arguments the PWS (S), the frequencies (nk), the plot title,
#and a logical switch: period=False for frequencies on x-axis, period=True for period on x-axis.
Plot_S_1D(S, nk, title='PWS {where}'.format(where=where), period=True)


plt.show()
```

Exercises

Example script: **DA_exercises_4.py**

```
def extract_area(data_in, where):

 if where=='NA':
  lat_min=0
  lat_max=60
  lon_min=280
  lon_max=360

 if where=='NP':
…

 if where=='SO':
…

 #Define a geographical constraint based on the input coordinates
 R=iris.Constraint(latitude=lambda  lat: lat_min <= lat <= lat_max,
                  longitude= lambda lon: lon_min <= lon <= lon_max )
 #Extract area
 data_out=data_in.extract(R)

#Plot selected area for visual inspection
bmap=Basemap(projection= 'gall', llcrnrlat= lat_min,  urcrnrlat= lat_max, llcrnrlon= lon_min,  urcrnrlon=
lon_max, resolution='l')
 …………..
```

Exercises

Example script: **DA_exercises_4.py**

```
def area_weighted(data_in):

 #call function to compute annuals mean from monthly means
 data_in=compute_annual_nc(data_in)

 #On a lat,long grid the grid-spacing reduces near the poles, we need to use area weights in
our spatial mean to take into account…
 data_in.coord('latitude').guess_bounds()
 data_in.coord('longitude').guess_bounds()
 cell_area = iris.analysis.cartography.area_weights(data_in)

 data_out= data_in.collapsed(['latitude', 'longitude'],
                                iris.analysis.MEAN,
                                 weights=cell_area)

 return(data_out)
```

Exercises

Example script: **DA_exercises_4.py**

```
################################  Plot 1d PWS  #######################################################

def Plot_S_1D(S, nk, title, period):

  fig=plt.figure(figsize=(10,5))
  ax=plt.gca()

  if period:
   ax.stem(1/nk, S, linefmt='orange', markerfmt='o',label=' ')
   ax.set_xlabel("Period (year)")
  else:
   ax.stem(nk, S, linefmt='blue', markerfmt='o', label=' ')
   ax.set_xlabel('Frequency  (year$^{-1}$)')


  ax.set_ylabel('PWSD')
  ax.set_title(title)


  return()
```