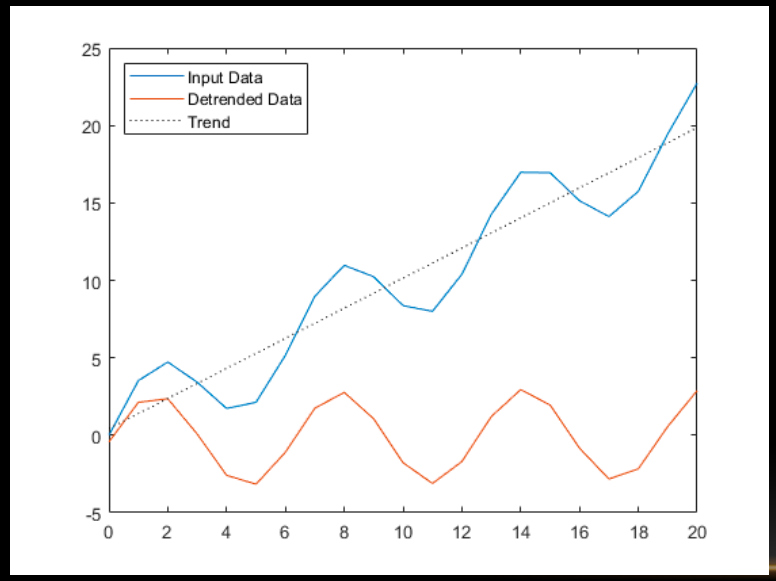
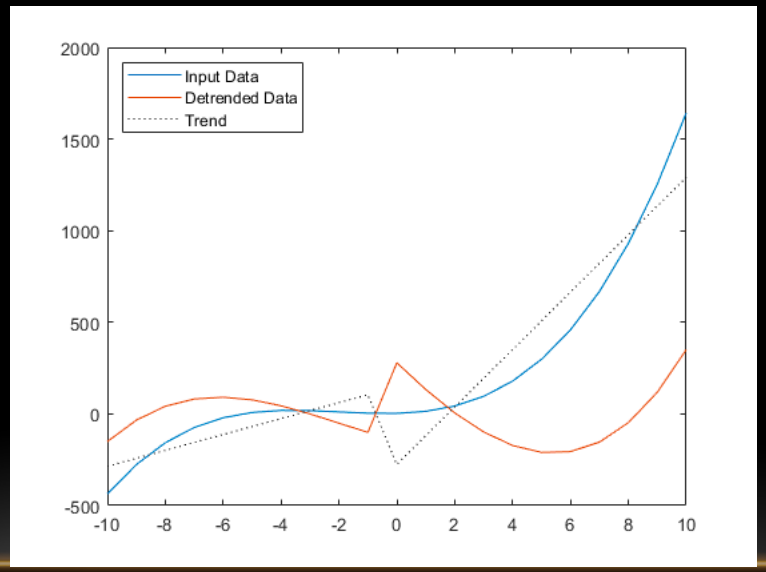
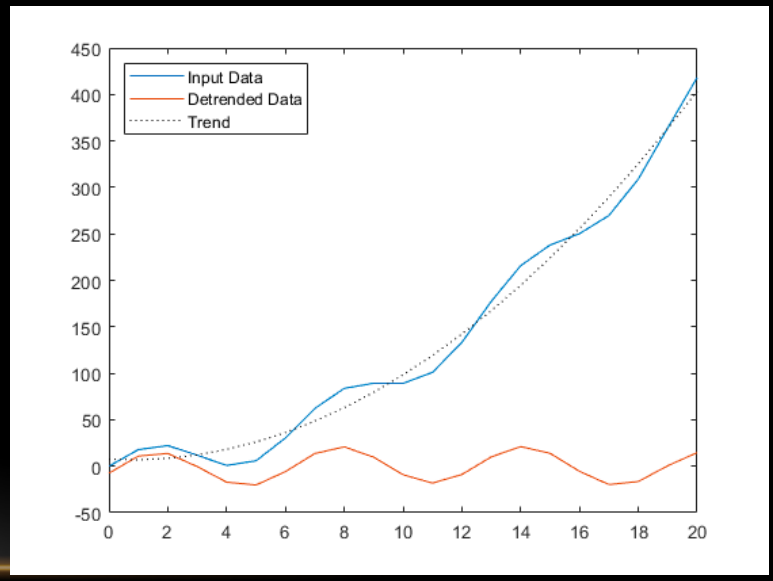
Trend and Detrending

- In meteorology, we often need to detrend data.
  - Continuous Linear trend: using a linear least-squares fit (left)
  - Discontinuous Linear trend: remove piecewise linear trends (center)
  - Continuous Nonlinear trend: for example, remove the continuous quadratic trend (right)







https://www.mathworks.com/help/matlab/ref/detrend.html

Linear Detrend Using Python

scipy.signal.detrend

`scipy.signal.detrend(data, axis=- 1, type='linear', bp=0, overwrite_data=False)` [source]

Remove linear trend along axis from data.

Parameters:

data : *array_like*

The input data.

axis : *int, optional*

The axis along which to detrend the data. By default this is the last axis (-1).

type : *{'linear', 'constant'}, optional*

The type of detrending. If `type == 'linear'` (default), the result of a linear least-squares fit to *data* is subtracted from *data*. If `type == 'constant'`, only the mean of *data* is subtracted.

bp : *array_like of ints, optional*

A sequence of break points. If given, an individual linear fit is performed for each part of *data* between two break points. Break points are specified as indices into *data*. This parameter only has an effect when `type == 'linear'`.

overwrite_data : *bool, optional*

If True, perform in place detrending and avoid a copy. Default is False

Returns:

ret : *ndarray*

The detrended input data.

https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.detrend.html

Linear Detrend Using Python (cont'd)

Examples

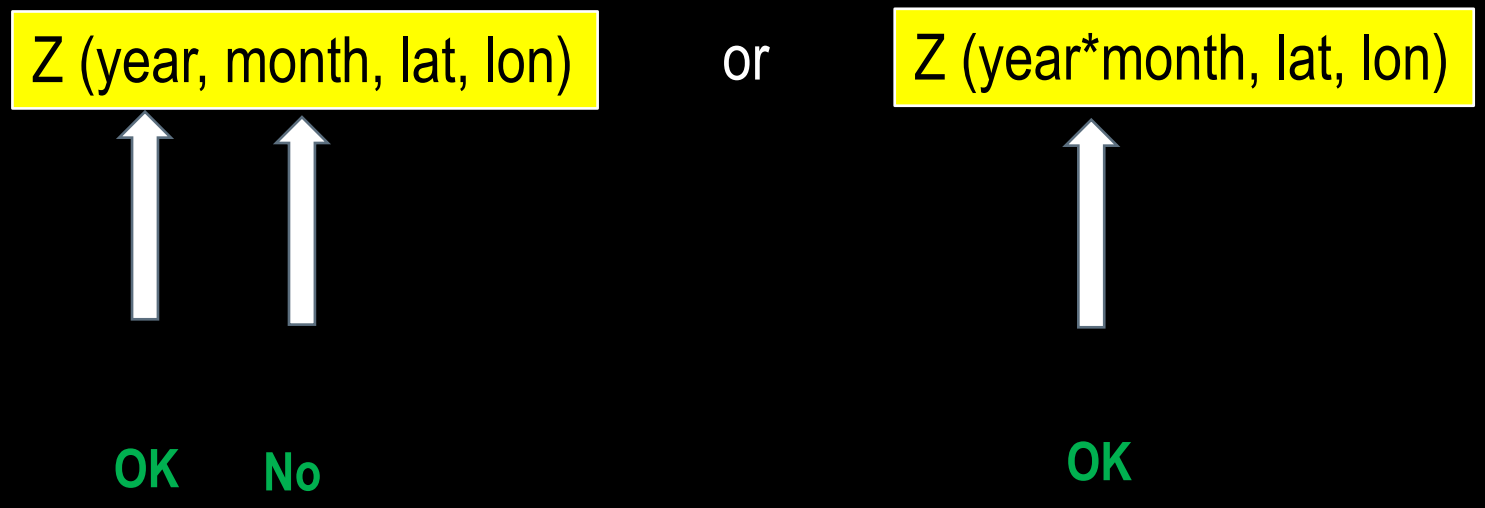
```
>>> from scipy import signal
>>> from numpy.random import default_rng
>>> rng = default_rng()
>>> npoints = 1000
>>> noise = rng.standard_normal(npoints)
>>> x = 3 + 2*np.linspace(0, 1, npoints) + noise
>>> (signal.detrend(x) - noise).max()
0.06  # random
```

Construct a testing data array

Detrend X

https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.detrend.html

How to detrend a dataset?



Significance Test: Mann-Kendall Trend Test

- The Mann-Kendall Trend Test can be used to determine whether a consistently increasing or decreasing trend (monotonic trends) exists in time series data .
- It is a non-parametric test, meaning that data doesn't have to meet the assumption of normality, but data should have no serial correlation.
- The hypotheses for the test are as follows:
  - H0 (null hypothesis): No trend is present in the data.
  - HA (alternative hypothesis): A trend is present in the data. (This could be a positive or negative trend)
- If the p-value of the test is lower than some significance level (e.g., 0.05, and 0.01), then there is a statistically significant trend present in the time series data.

https://www.statology.org/mann-kendall-test-python/

Significance Test: Mann-Kendall Trend Test (cont'd)

```
#create dataset
data = [31, 29, 28, 28, 27, 26, 26, 27, 27, 27, 28, 29, 30, 29, 30, 29, 28]

#perform Mann-Kendall Trend Test
import pymannkendall as mk

mk.original_test(data)
```

```
Mann_Kendall_Test(trend='no trend', h=False, p=0.422586268671707,
                  z=0.80194241623, Tau=0.147058823529, s=20.0,
                  var_s=561.33333333, slope=0.0384615384615, intercept=27.6923076
```

- **trend**: This tells the trend. (Possible output includes increasing, decreasing, or no trend)
- **h:** True if trend is present. False if no trend is present.
- **p:** The p-value of the test.
- **z:** The normalize test statistic.
- **slope:** Theil-Sen estimator/slope
- **intercept:** Intercept of Kendall-Theil Robust Line

https://www.statology.org/mann-kendall-test-python/