MUSIC Method for Spectral Estimation

Introduction

The **MU**tiple **SI**gnal **C**lassifier (MUSIC) method is an approach for spectral estimation that offers higher frequency resolution than Fourier based approaches. This technique is particularly appropriate for signals that consists of multiple sinusoids polluted with white (i.e. Gaussian) noise.

This application generates a noisy sinusoidal data set, and then applies the MUSIC method to identify the frequencies used to generate the signal.

Consider a signal x(n) that is the sum of r sinusoids, with N samples. If the signal is polluted with white noise w(n), then

$$x(n) = \sum_{i=1}^{r} A_i e^{j\omega_i n + \phi_i} + w(n)$$

where A_i and ω_i is the amplitude and frequency of the i^{th} sinusoid, i = 1 ... r and n = 1 ... N

The autocorrelation matrix of the signal has N eigenvalues and N eigenvectors.

- the r largest eigenvalues represent eigenvectors associated with the subspace of the signal plus noise
- the remaining eigenvectorsr are only assoicated with the noise subspace

Theoretically, the projection of the eigenvectors of the noise subspace onto the signal frequencies is 0 (since these eigenvectors are orthogonal to the signal). Hence this function is theoretically infinite at a signal frequency ω.

$$P(\omega) = \frac{1}{\sum_{i=r+1}^{N} \left| a(\omega)\, q_i \right|^2}$$

where $a(\omega) = \left[1, e^{j\omega}, e^{2j\omega} \ldots e^{(M-1)j\omega}\right]$ and q_i are the smallest N - r eigenvalues.

P(ω) is known as a "pseudospectrum". The peaks in this function only identify the location of

frequencies, and are not a true spectrum.

The value of r (i.e. the size of the signal subspace) is a choice of the analyst.

- a large r gives a detailed pseudospectrum that might contain spurious spectral peaks
- a small r gives a smooth pseudospectrum that might miss weak spectral peaks

Since $P(\omega)$ can be evaluated at any ω, the MUSIC method offers a form of *superesolution* - that is, frequencies smaller than one sample.

```
> restart:
  with(SignalProcessing):
  with(Statistics):
  with(LinearAlgebra):
```

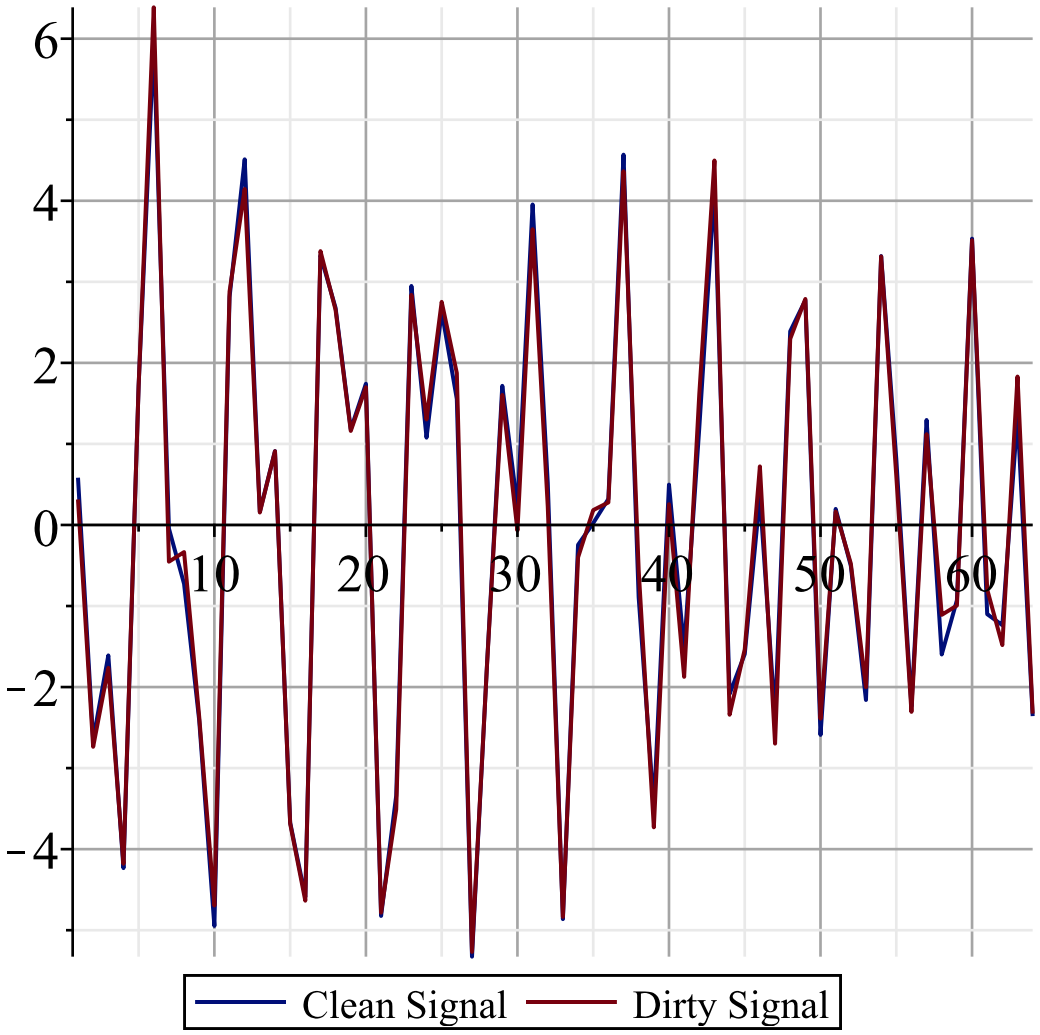
Generate a Noisy Signal

Frequencies, amplitudes and number of samples. Note that two of the sinusoids have very similar frequencies

```
> omega1 := 1.0:
  omega2 := 1.05:
  omega3 := 2.2:

  A1     := 1.5:
  A2     := 2.5:
  A3     := 2.5:

  N      := 64:

  clean_signal := Vector(N, i -> A1 * cos(omega1 * i) + A2 * cos
  (omega2 * i) + A3 * cos(omega3 * i), datatype = float[8]):
```

Add Gaussian noise to the signal

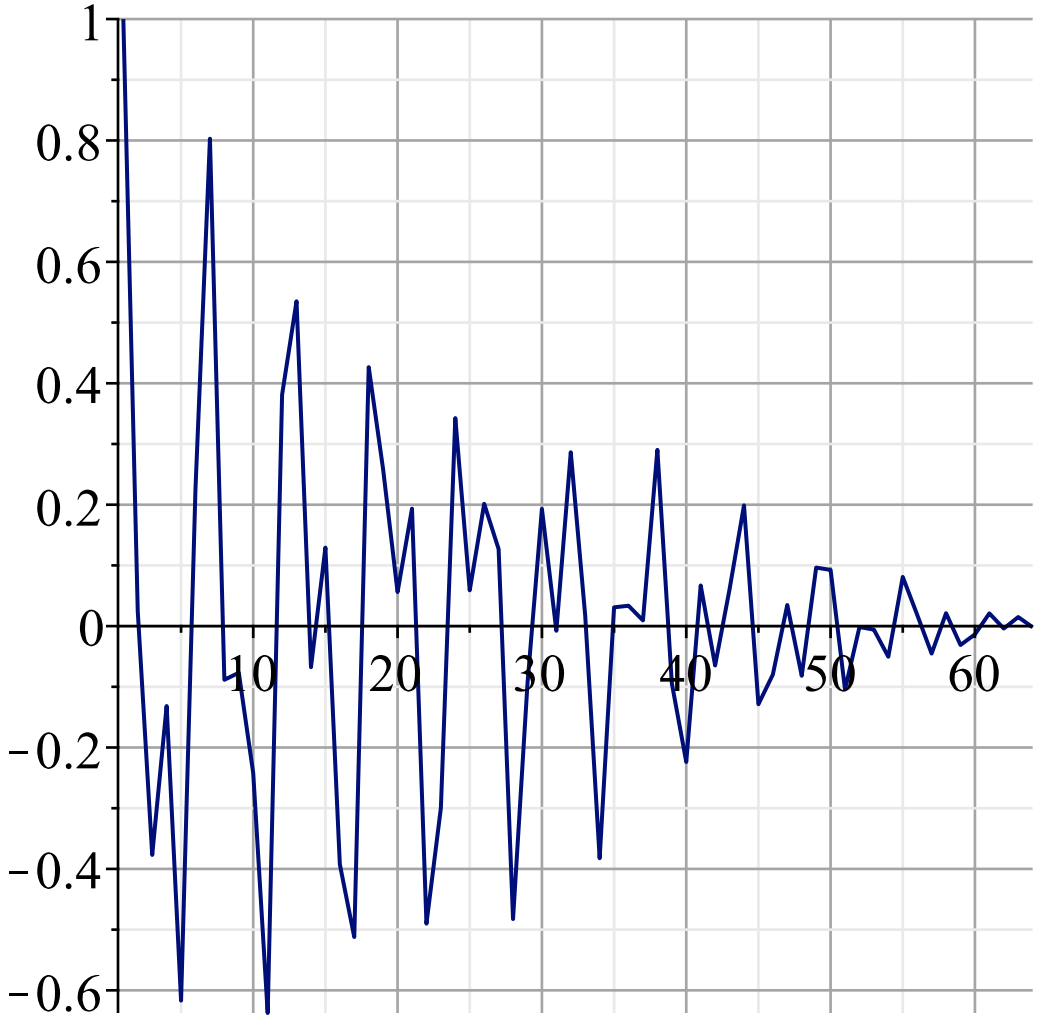
```
> Nmean := 0.0:
  Nstd  := 0.25:

  dirty_signal := clean_signal +~ Sample(Distribution(Normal
  (Nmean, Nstd)), N):

> dataplot([clean_signal, dirty_signal], legend = ["Clean
  Signal", "Dirty Signal"],style = line, gridlines)
```



Autocorrelation of the Noisy Signal

Calculate the autocorrelation of the mean zeroed data

```
> corr_matrix := AutoCorrelation(dirty_signal -~ Mean
  (dirty_signal)):
  dataplot(corr_matrix, style = line, gridlines)
```



Eigenvalues and Eigenvalues of the Correlation Matrix

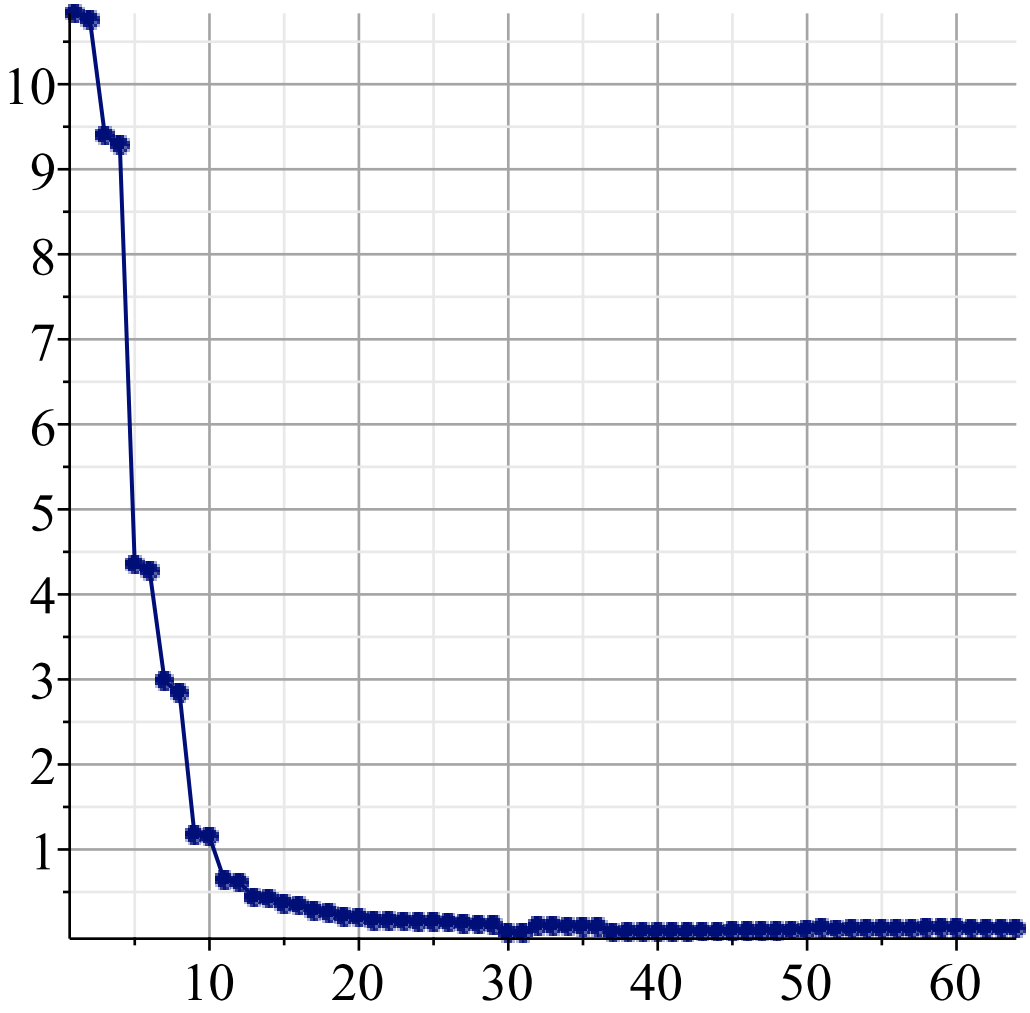
The autocorrelation vector is the first row of a circulant matrix (a specialized kind of Toeplitz matrix)

```
> toep_mat := Matrix(N, N, (i, j) -> corr_matrix[abs(i - j) + 1],
  datatype = float[8])
```

$$toep\_mat := \begin{bmatrix} 1. & 0.0242069504986711 & -0.376969027540246 & -0.13165801 \\ 0.0242069504986711 & 1. & 0.0242069504986711 & -0.37696902 \\ -0.376969027540246 & 0.0242069504986711 & 1. & 0.024206950 \\ -0.131658019491073 & -0.376969027540246 & 0.0242069504986711 & 1. \\ -0.616961223508436 & -0.131658019491073 & -0.376969027540246 & 0.024206950 \\ 0.226895070136685 & -0.616961223508436 & -0.131658019491073 & -0.37696902 \\ 0.802908325555154 & 0.226895070136685 & -0.616961223508436 & -0.13165801 \\ -0.0884621207562059 & 0.802908325555154 & 0.226895070136685 & -0.61696122 \\ -0.0768932116893337 & -0.0884621207562059 & 0.802908325555154 & 0.22689507 \\ -0.242287644772093 & -0.0768932116893337 & -0.0884621207562059 & 0.80290832 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Eigenvalues and eigenvectors of the autocorrelation matrix

```
> eig_vals, eig_vecs := Eigenvectors(toep_mat):

  dataplot(eig_vals, gridlines);
```



Note that the largest eigenvalues indicate the presence of sinusoids in the signal, while the smallest eigenvalues are associated with the noise.

Music Estimator and Pseudospectrum

The r largest eigenvalues represent those eigenvectors associated with the signal (the other eigenvectors are associated with noise)

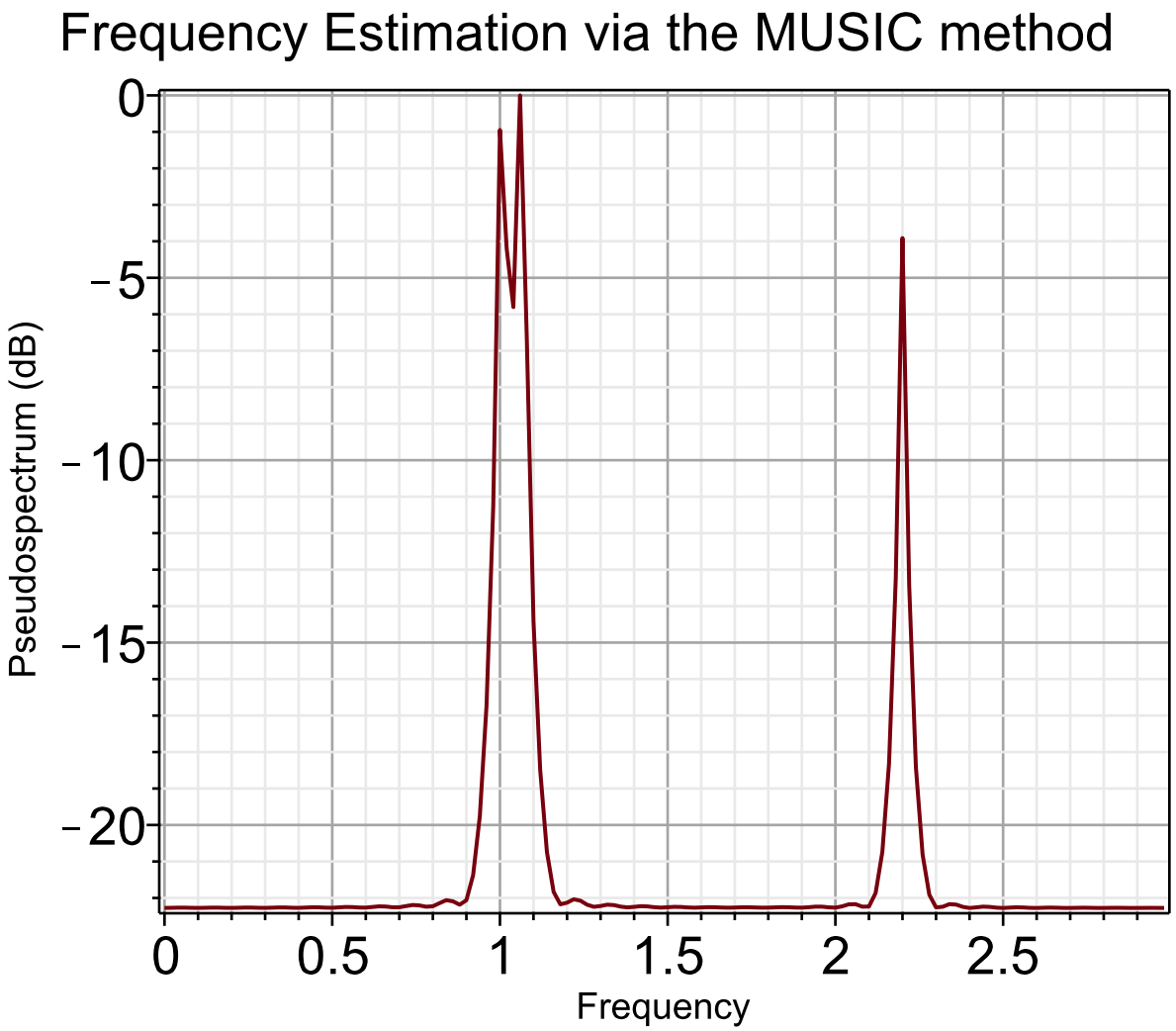
```
> r := 6:
```

```
> a_omega := Vector([seq(exp((i - 1.0) * I * omega), i = 1 .. N)])
  :

  p := proc(freq)
    local i, A_omega:
    A_omega := eval(a_omega, omega = freq):
    1 / add(abs(A_omega . eig_vecs[.., i]) ^ 2, i = r + 1 .. N);
  end proc:
```

Evaluate the music estimator at a range of angles, and plot the results.

```
> max_search_angle := 3.0:
  num_points       := 150:
```

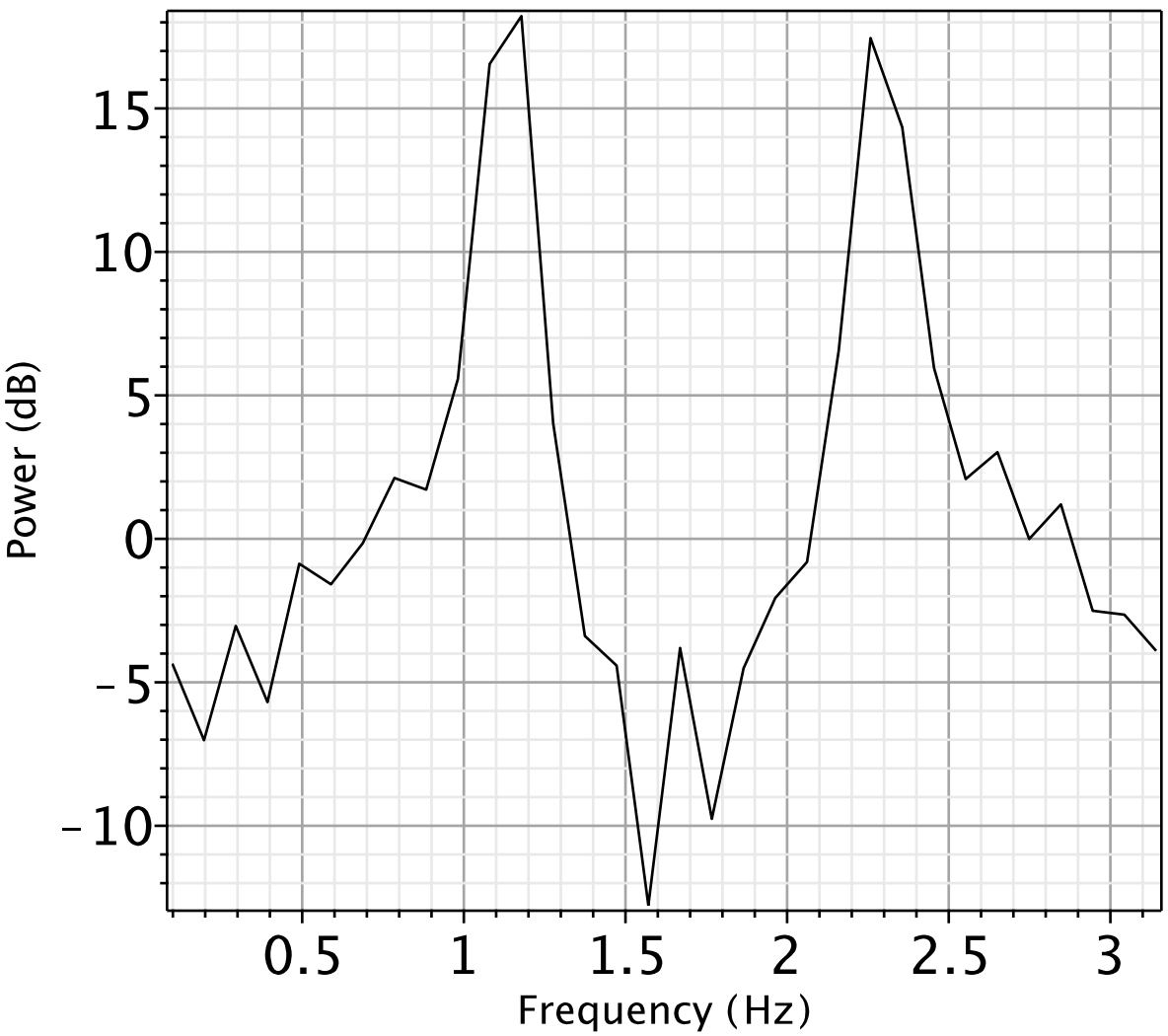
```
thetas           := [seq(  max_search_angle / num_points * (i -
1),  i = 1..num_points)]:
music_estimates  := [seq(p(max_search_angle / num_points * (i -
1)), i = 1..num_points)]:

> plot(thetas, 10*log10~(music_estimates/max(music_estimates)),
  axes = boxed, title = "Frequency Estimation via the MUSIC
  method", labels = ["Frequency","Pseudospectrum (dB)"],
  labeldirections = [horizontal, vertical], font = [Arial],
  titlefont = [Arial, 16], labelfont = [Arial, 11], size = [600,
  400], gridlines)
```



The three peaks correctly identify the location of the frequencies.

Now compare with a periodogram generated with a Fourier approach. The two similar frequencies cannot be discriminated.

```
> Periodogram(dirty_signal, samplerate = 2 * Pi, size = [600,
  400])
```


15
10
5
0
-5
-10
Power (dB)
0.5
1
1.5
2
2.5
3
Frequency (Hz)