Department of Civil, Construction, and Environmental Engineering
North Carolina State University

Module 4: Frequency Domain Signal Processing and Analysis

Course, Curriculum, and Laboratory Improvement: Integration of Sensor Technologies in the Civil Engineering Curriculum, DUE0837612

Sensors in the CCEE Curriculum

- Sensors are increasingly used by practicing civil engineers to monitor, measure, and remotely observe the world around them. However, these technologies are not present in traditional CCEE curricula.
- A research team of T.M. Evans (PI), M.A. Gabr (co-PI), Z. Ning (GRA), and C. Markham (URA) received a grant from the National Science Foundation to address this issue by developing educational modules that will be incorporated in a series of CCEE courses.
- Each module will build on the content from previous models, providing both breadth and depth to your studies.

List of Educational Modules

- Module 1: Fundamentals of Geotechnical Sensing and Instrumentation (CE 342)
- Module 2: Sensor Installation and Operation (CE 440)
- Module 3: Data Analysis and Interpretation (CE 443)
- Module 4: Frequency Domain Signal Processing and Analysis (CE 548)

Module 4: Learning Objectives

- At the completion of this module, you will be able to:
  - Explain the need for frequency domain analysis and its principles;
  - Draw conclusions about physical processes based on analysis of sensor data;
  - Combine signals in a meaningful way to gain deeper insight into physical phenomena.

Overview

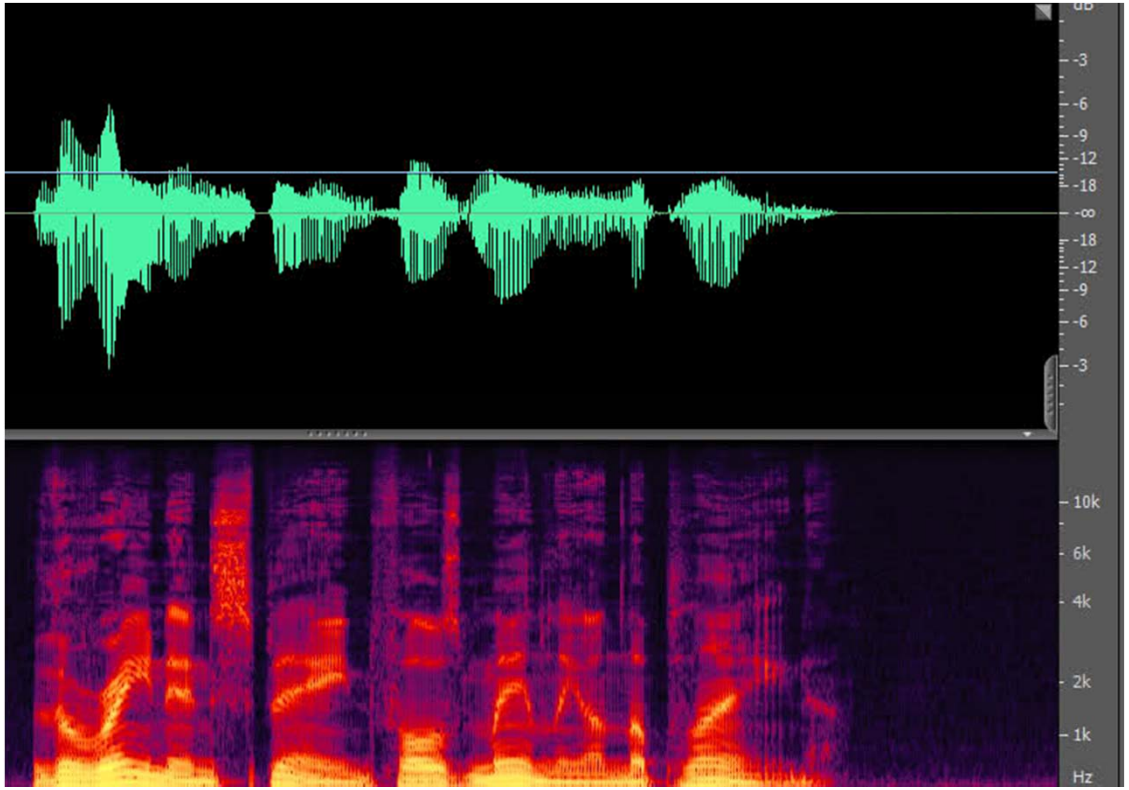
- Basic concepts in frequency domain signal processing and analysis
  - Fourier Transform
  - FFT (Fast Fourier Transform)
- Implementation of FFT in MATLAB and Mathcad
- Example problems
  - Noise reduction with filters
  - Leakage
  - Frequency resolution

Frequency Domain Analysis of Signals

- In engineering and statistics, frequency domain is a term used to describe the analysis of mathematical functions or signals with respect to **frequency**, rather than time.
- The most common purpose for analysis of signals in the frequency domain is the analysis of signal properties. The engineer can study the spectrum to determine which frequencies are present in the input signal and which are missing.
  - Better identify the characteristics of signal
  - Add or subtract frequencies to the original signal
- Signals are converted from time or space domain to the frequency domain usually through the **Fourier transform.**

A voice waveform in time and frequency domain

- **Time domain**
- **Frequency domain**



http://i52.tinypic.com/eqp5df.jpg

Course, Curriculum, and Laboratory Improvement: Integration of Sensor Technologies in the Civil Engineering Curriculum, DUE0837612

Fourier Series

- A continuous periodic function *f(t)* with period *T* can be expressed as a linear combination of sinusoids, which is known as Fourier series:

$$f(t)=\sum_{u=-\infty}^{\infty}\left[a_u\cdot\cos\left(u\frac{2\pi}{T}t\right)+b_u\cdot\sin\left(u\frac{2\pi}{T}t\right)\right] \qquad \textbf{Eq.1}$$

- Recall Euler's formula,

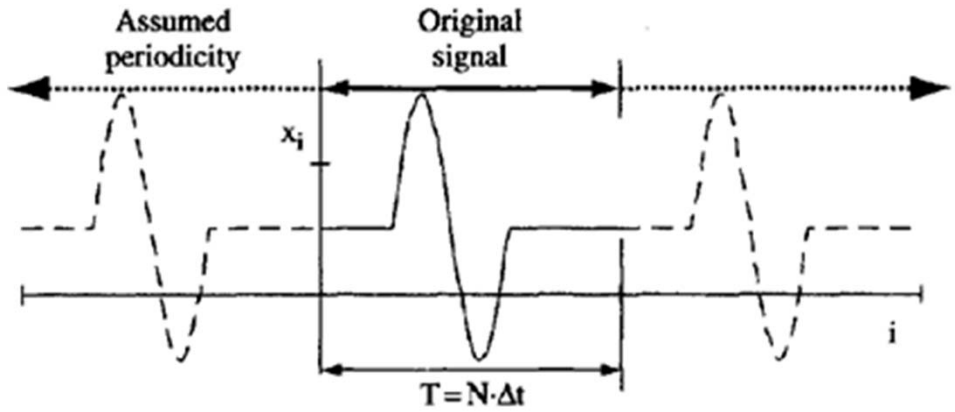
$$e^{jnx}=\cos(nx)+j\sin(nx) \qquad j=\sqrt{-1} \qquad \textbf{Eq.2}$$

- Fourier series can also be expressed as:

$$f(t)=\sum_{u=-\infty}^{+\infty}X_u\cdot e^{j\cdot\left(u\frac{2\pi}{T}t\right)} \qquad j=\sqrt{-1} \qquad \textbf{Eq.3}$$

Discrete Fourier Transform

- Discrete time signals can be analyzed or decomposed into a series of sines and cosines. This representation is called the discrete Fourier transform (DFT)
- DFT presumes periodic signals. Any aperiodic signal $\underline{x}$ with N points $[x_0, x_1, \ldots, x_{n-1}]$ is automatically assumed periodic with fundamental period $T = N \times \Delta t$



(after Santamarina et al., 2005)

Discrete Fourier Transform

- Re-write the Fourier transform in discrete form

$$f(t)=\sum_{u=-\infty}^{+\infty} X_u \cdot e^{j\cdot\left(u\frac{2\pi}{T}t\right)} \xrightarrow[t_i = i\cdot\Delta t]{T = N\cdot\Delta t} x_i=\sum_{u=-\infty}^{+\infty} X_u \cdot e^{j\cdot\left(u\frac{2\pi}{N}i\right)} \qquad j=\sqrt{-1} \qquad \textbf{Eq.4}$$

- Fourier coefficient X_u is a complex number. Then the *amplitude* and the *frequency* of the component of the signal corresponds to X_u can be defined as:

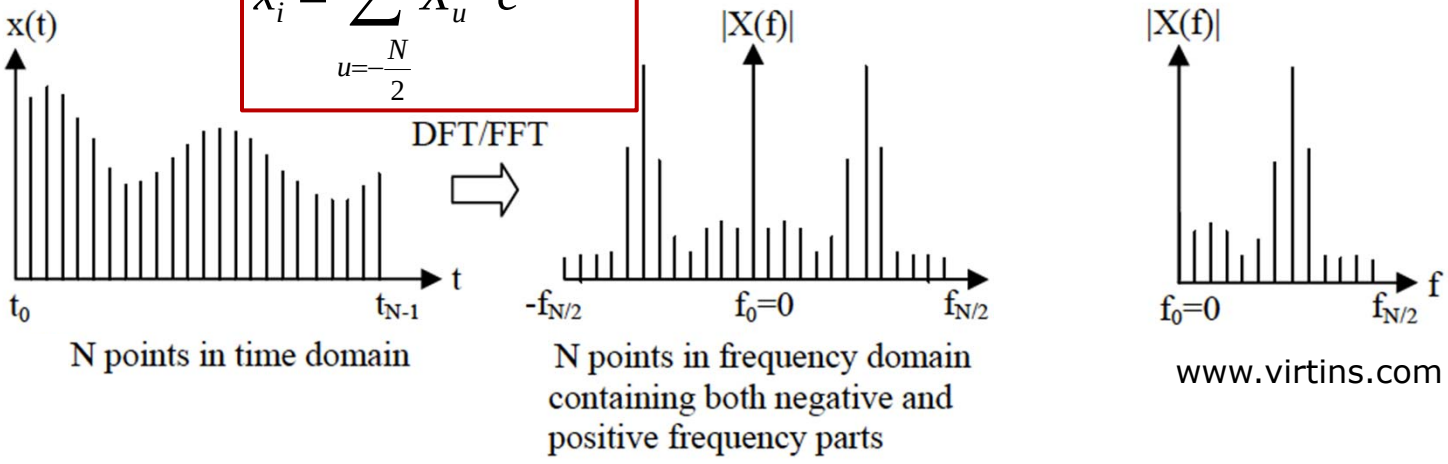
- *Amplitude:* $$\left|X_u\right|=\sqrt{\left[\mathrm{Re}\left(X_u\right)\right]^2+\left[\mathrm{Im}\left(X_u\right)\right]^2} \qquad \textbf{Eq.5}$$

- *Frequency:* $$f=u\cdot\frac{1}{T}=u\cdot\frac{1}{N\cdot\Delta t} \qquad \textbf{Eq.6}$$

Discrete Fourier Transform

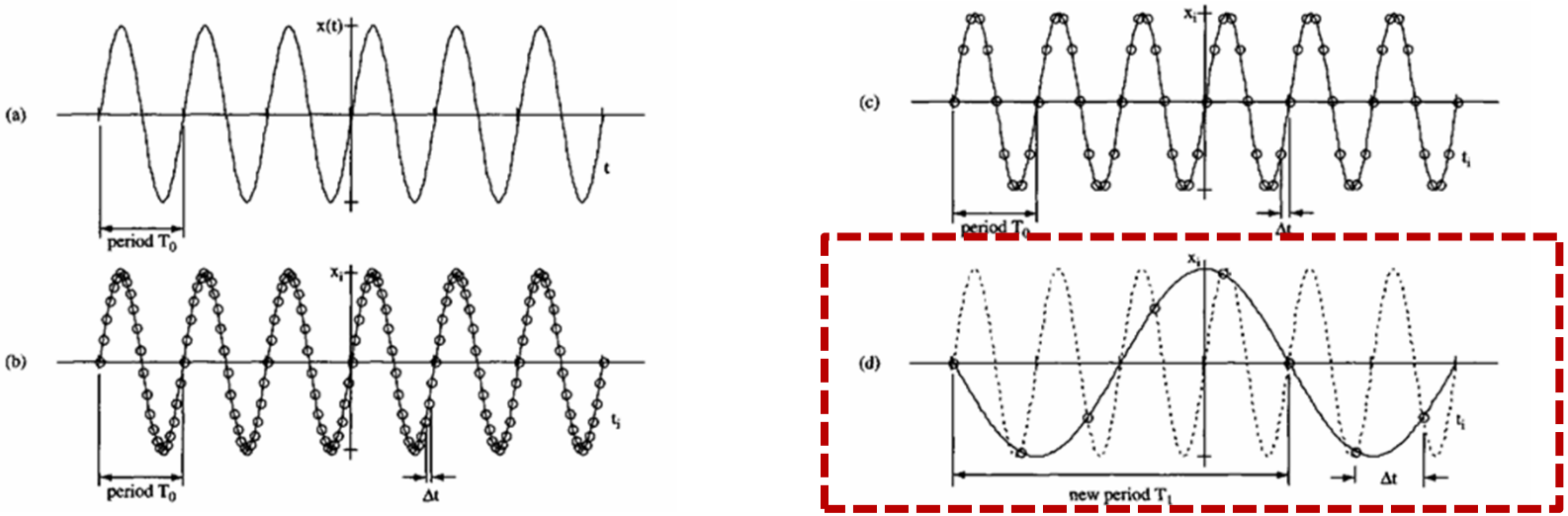
- The highest frequency that can be resolved from a discrete time signals is 1/(2Δt) (Nyquist Criterion), So the Fourier series need not extend beyond u=N/2
- Because negative frequencies are not physically measured, DFT is usually defined only between u =0 and u = N/2

$$x_i = \sum_{u=-\frac{N}{2}}^{\frac{N}{2}-1} X_u \cdot e^{j\cdot\left(u\frac{2\pi}{N}i\right)}$$



Aliasing

- Aliasing occurs when the sampling frequency is not high enough. It causes high frequency components to "fold over" and appear "aliased" into a lower frequency



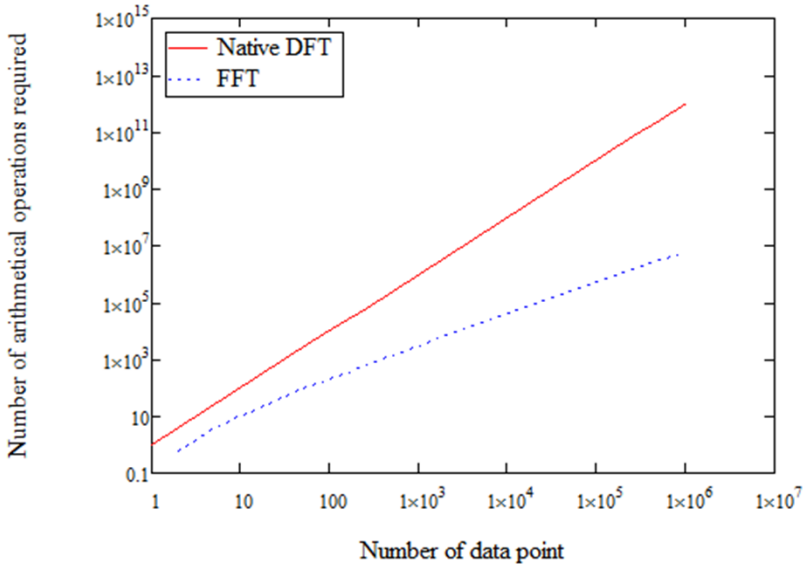
(after Santamarina et al., 2005)

The only solution to the aliasing problem is to ensure that the sampling rate is higher than **twice** the highest frequency present in the signal.

Fast Fourier Transform (FFT)

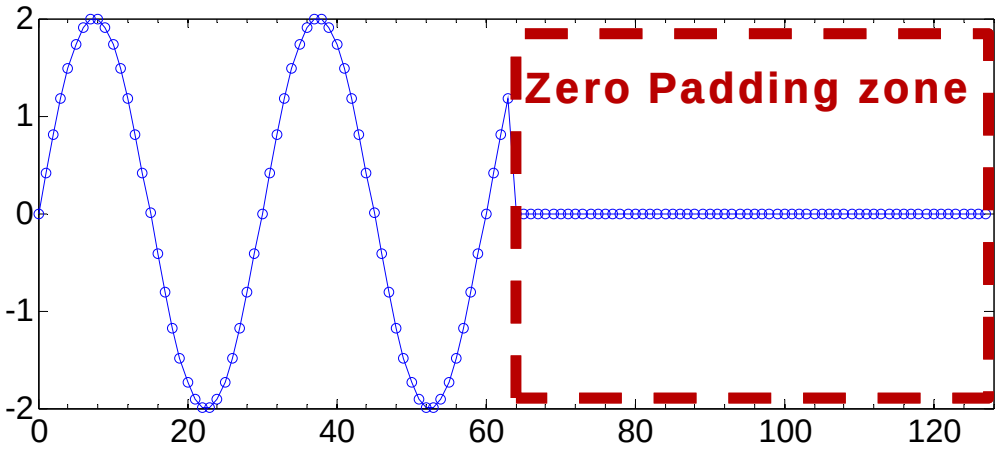
- The FFT is a faster version of the Discrete Fourier Transform (DFT). The FFT utilizes some clever algorithms to do the same thing as the DFT, but in much less time. (Cooley and Tukey, 1965 and many others)

- Computing a DFT of N points in the native way, using the definition, takes $O(N^2)$ arithmetic operations, while an FFT can compute the same result in only O(N log N) operations. This huge improvement made many DFT-based algorithms practical.



Zero Padding

- FFT generally requires the number of data points N to be an integer power of 2 (e.g. 1024, 2048, 4096, etc.) to attain maximum computational efficiency. If not, zeroes can be added at the end of the original record. This is called zero padding.



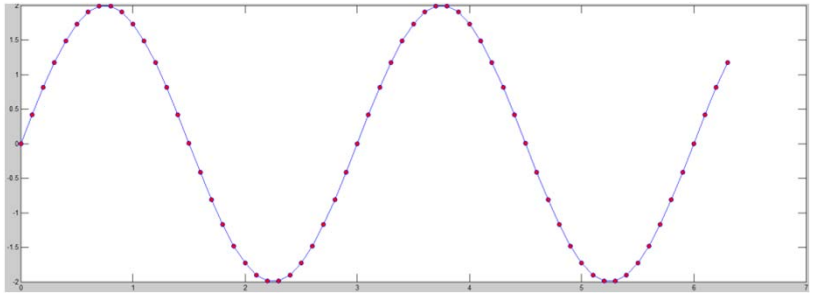
Implementation of FFT in MATLAB

- In MATLAB, the function *fft*(*x*, *N*) is used to perform the fast Fourier transform of a signal in the time domain. It is simple, but may appear confusing for the first-time user.
  - $\underline{X}$ = ***fft***(*x*, *N*) produces an array, containing '*N*' complex Fourier coefficients. (default value of *N* = the number of sampling points in the time domain signal, x)
  - If *N* > the sampling points in the time domain signal, the original signal will be extended by appending a certain number of zeroes to the end of the signal to make it include *N* points.
  - Values of $\underline{X}$ are centered around index *N*/2, corresponding to $f_{N/2}$= 1/(2Δt), not around *zero*. It is a two-sided definition of the spectrum.
  - ***fft***(x, *N*) will not convert the discrete *time* data to *frequency* accordingly. It is up to the user to define the *frequency corresponding to each element in* $\underline{X}$.

Department of Civil, Construction, and Environmental Engineering
North Carolina State University

Implementation of FFT in MATLAB

- Generate a discrete time domain signal : **y=2*sin(2*pi/3.0*t)**



Period of the sinusoidal: 3
Sampling dt: 0.1
Sampling points N: 64
Sampling duration: 6.3

- Perform a FFT on 'y' :

```
Command Window

  Columns 56 through 60

  -0.9433 - 1.6711i  -0.9970 - 1.9388i  -1.0785 - 2.2982i  -1.2121 - 2.8148i  -1.4580 - 3.6405i

  Columns 61 through 64

  -2.0108 - 5.2405i  -3.9869 -10.1722i  25.3372 +54.9651i   3.2946 + 4.2754i

fx >>
```

Return a vector 'X' containing 64 complex numbers

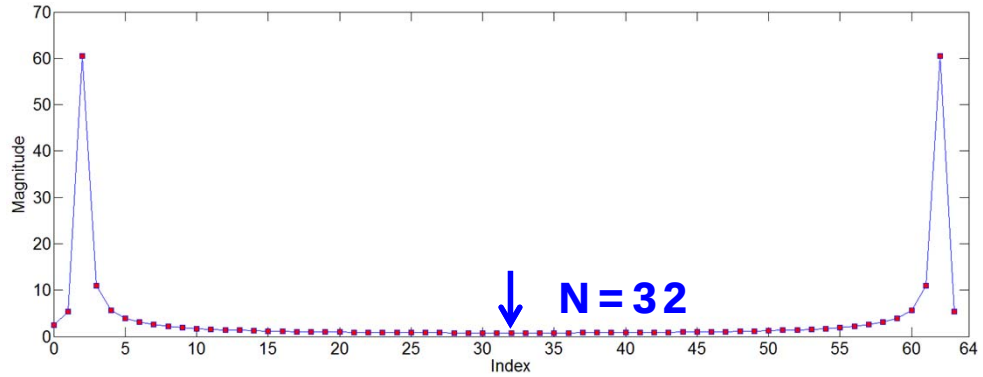
- Calculate the magnitude of 'X' : **X1 = abs(X)**

Course, Curriculum, and Laboratory Improvement: Integration of Sensor Technologies in the Civil Engineering Curriculum, DUE0837612

Department of Civil, Construction, and Environmental Engineering
North Carolina State University

Implementation of FFT in MATLAB

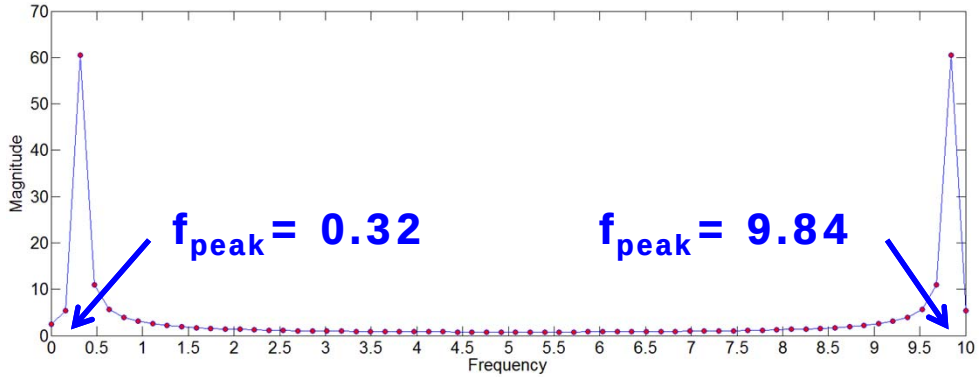
- Plot magnitude '*X1*' over index '*i*'



The actual frequency corresponding to each magnitude has not been defined yet.

This is a two-sided definition of the spectrum, and it is centered around index 64/2=32, which corresponds to f_{max}=1/(2*dt).

- Define frequency vector '*f*', and plot '*X1*' over '*f*'

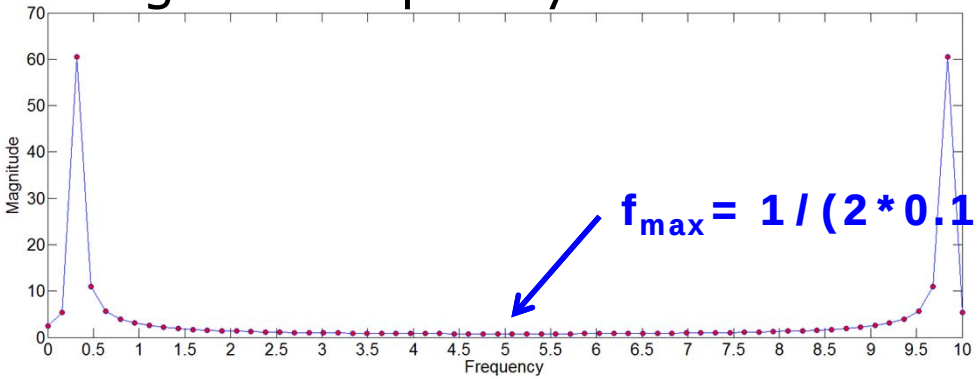


df=1/sampling duration = 1/6.3
f = 0 ~ 63*df

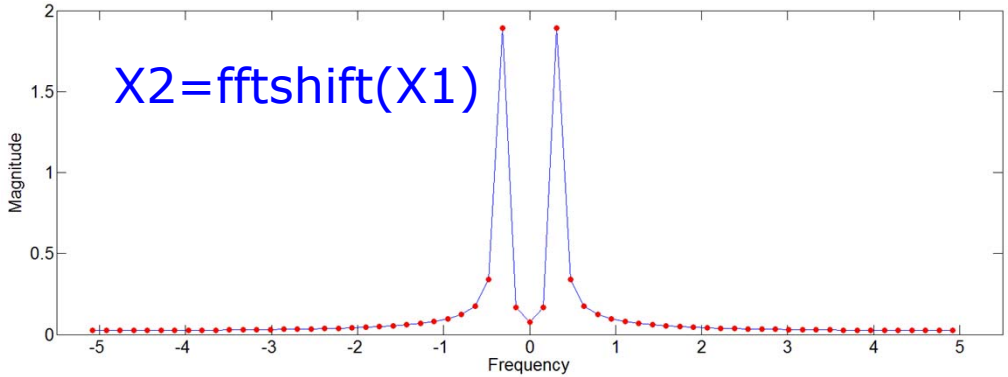
Course, Curriculum, and Laboratory Improvement: Integration of Sensor Technologies in the Civil Engineering Curriculum, DUE0837612

Implementation of FFT in MATLAB

- The highest frequency that can be resolved is 1/(2Δt)



- Use ***fftshift*** to convert it to a two-sided spectrum centered around *f*=0



Don't forget to redefine *f*
***f* = -32*df ~ 31*df**

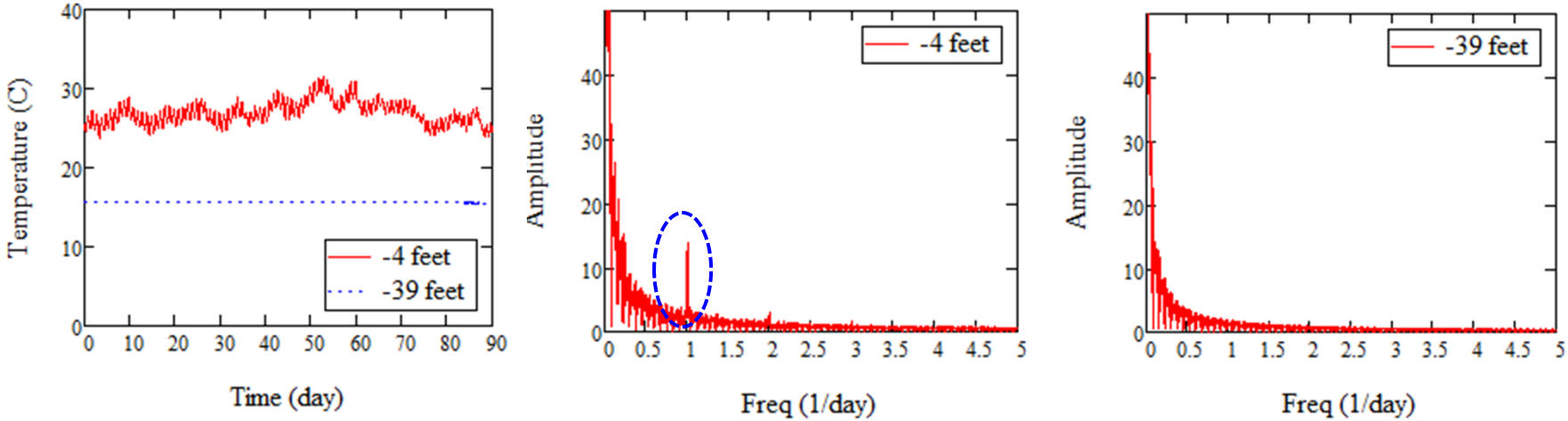
Implementation of FFT in Mathcad

- In Mathcad, the function *fft*(*x*) is used to perform the fast Fourier transform of a signal in time domain. The conversion is more straightforward.
  - $\underline{X}$ = ***fft***(*x*) requires the input signal in time domain contains 2^n data points, and will return a array of contains $2^{n-1}+1$ complex Fourier coefficients, corresponds to the frequency from *0* to $f_{N/2} = 1/(2\Delta t)$. It is a one-sided definition of the spectrum.
  - ***fft***(x) will not convert the discrete *time* data to *frequency* accordingly. It is up to the user to defined the *frequency*, *corresponds to each element in* $\underline{X}$.
  - If the number of sampling points in the input signal is not = 2^n, the user needs to manually extended the signal by appending a certain number of zeroes to the end of the signal to make it includes 2^n points, before using *fft* function.
  - Note: the ***cfft***(x) function can be applied to a signal of any length, but it is neither as fast nor as intuitive. Its use is discouraged

Department of Civil, Construction, and Environmental Engineering
North Carolina State University

Example: Frequency domain analysis of thermistor readings

Temperature readings taken at Lake Raleigh Dam during 06/2011 to 08/2011 are shown below in both time domain and frequency domain.



In the **time domain**, it can be seen that the temperature near the ground surface fluctuated while the temperature at 39 ft. deep was fairly stable. **Frequency domain** analysis helps further reveal that the near ground surface temperature was subjected to a daily cyclic change, which is indicated by a peak at f = 1 day^{-1}.

Course, Curriculum, and Laboratory Improvement: Integration of Sensor Technologies in the Civil Engineering Curriculum, DUE0837612

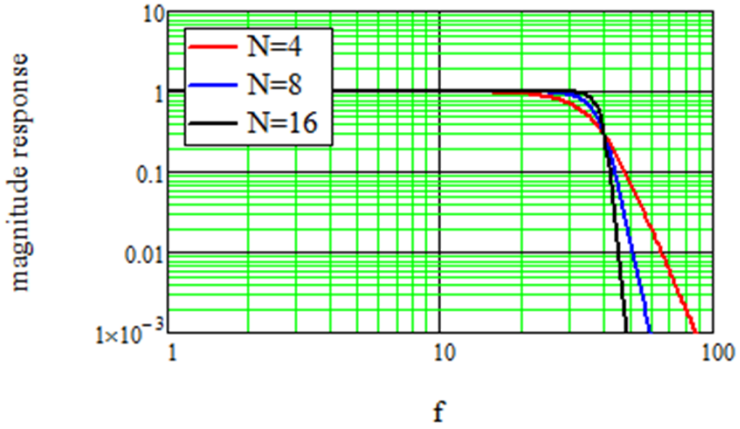
Filters in the Frequency Domain

- A filter in the frequency domain is a 'window' that passes certain frequency components and rejects others. The most common filters are:
  - Low-pass filter: Low frequency components below the cutoff frequency pass. It is used to reduce high-frequency noise.
  - High-pass filter: High frequency components above the cutoff frequency pass. It is used to reduce low-frequency noise.
  - Band-pass filter: A combination of low and high pass filters. It is used to keep a frequency band.

Filters in the Frequency Domain

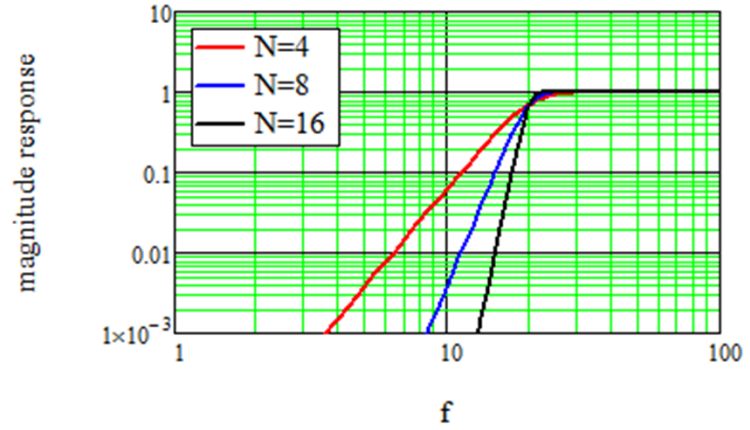
- Butterworth Low-pass filter

$$G_{low}(f) = \frac{1}{\sqrt{1+\left(\frac{f}{f_c}\right)^{2\cdot N}}}$$



- Butterworth High-pass filter

$$G_{hi}(f) = \frac{1}{\sqrt{1+\left(\frac{f_c}{f}\right)^{2\cdot N}}}$$



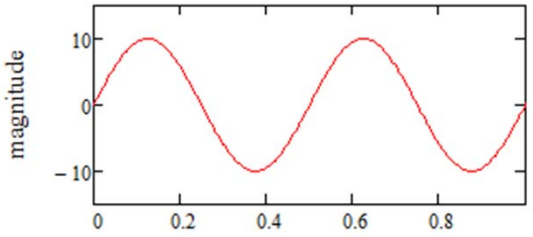
G = magnitude response (reduction factor)
f = frequency
f_c = cut-off frequency
N = "order" of the filter

Filters in the Frequency Domain

- Implementation Procedure
  - convert signal from time domain to frequency domain by FFT: $\underline{X}=\mathbf{FFT}(\underline{x})$.
  - Plot Magnitude $|X_u|$ vs. f_u to determine the frequency of interest
  - Choose the type of filter, select filter parameters , define filter array *G* base on frequency *f*
  - Apply filter to the amplitude of the signal in the time domain; i.e., multiply point by point: $|Y_u|=G_u .|X_u|$.
  - Compute the filtered signal: $Y=|Y_u|[\cos(\varphi_u)+i\sin(\varphi_u)]$, where φ is phase angle.
  - Convert the signal back in time domain by inverse FFT: $y=\mathbf{IFFT}(Y)$

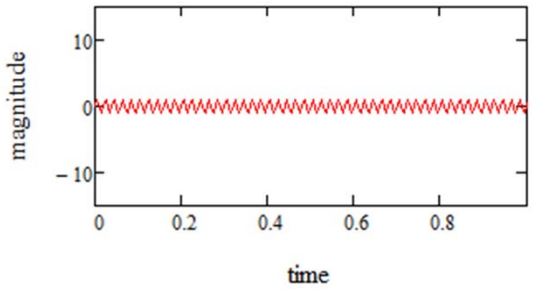
Filters in the Frequency Domain

$$S_1 = 10\sin\left(\frac{2\pi}{0.5}t\right)$$

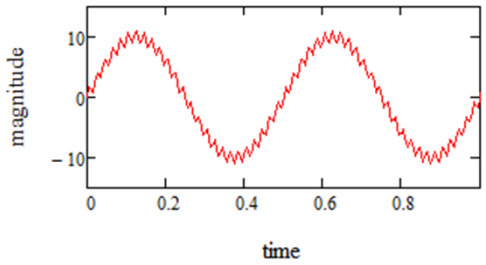


$$S_2 = \sin\left(\frac{2\pi}{0.02}t\right)$$

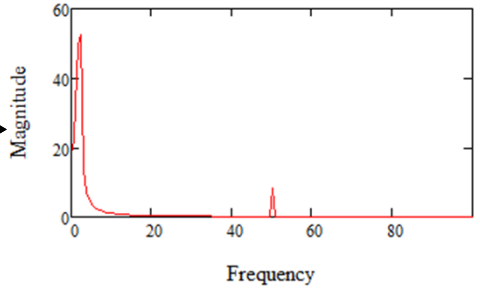
+



=

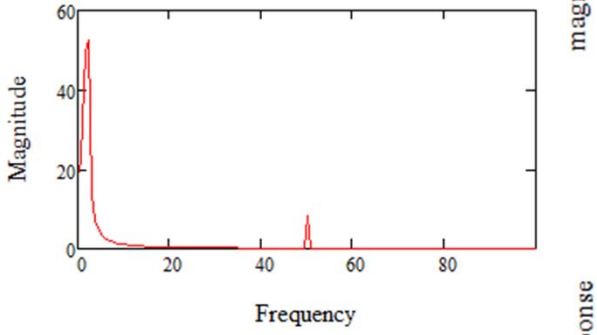


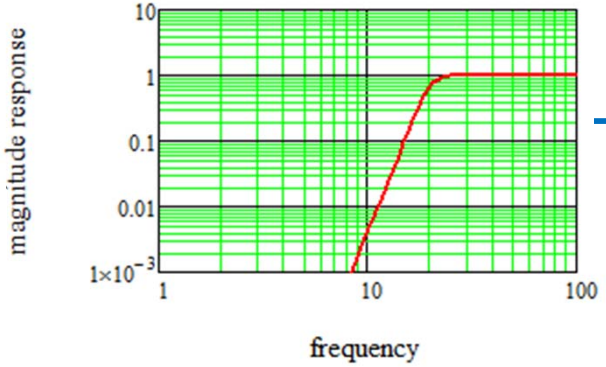
FFT →

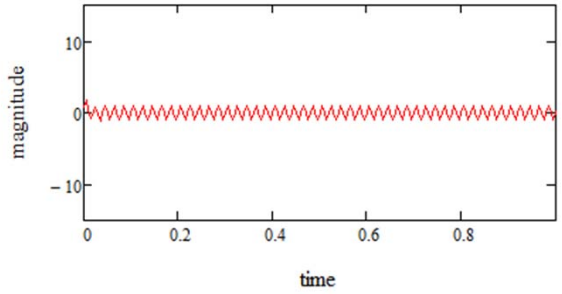


Filters in the Frequency Domain

Apply high-pass filter to Remove low frequency noise

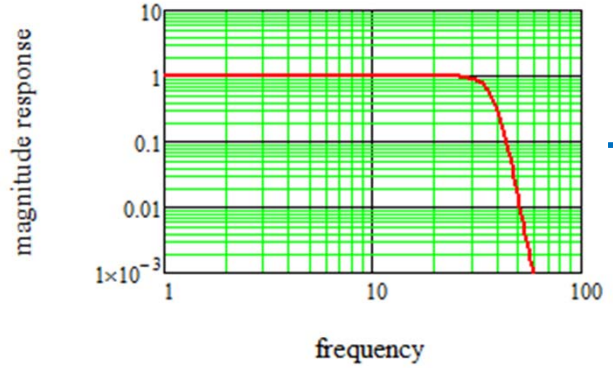


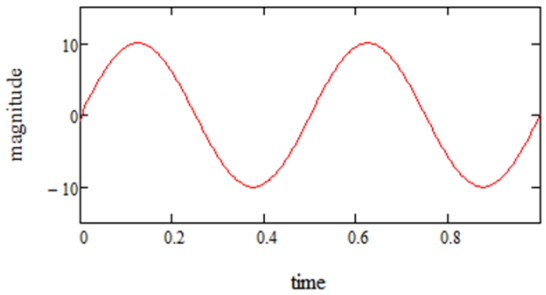




Cut-off frequency = 20

Apply low-pass filter to Remove high frequency noise





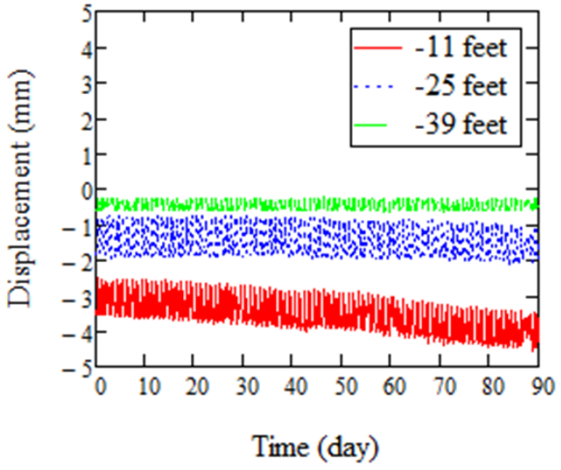
Cut-off frequency = 40

Course, Curriculum, and Laboratory Improvement: Integration of Sensor Technologies in the Civil Engineering Curriculum, DUE0837612

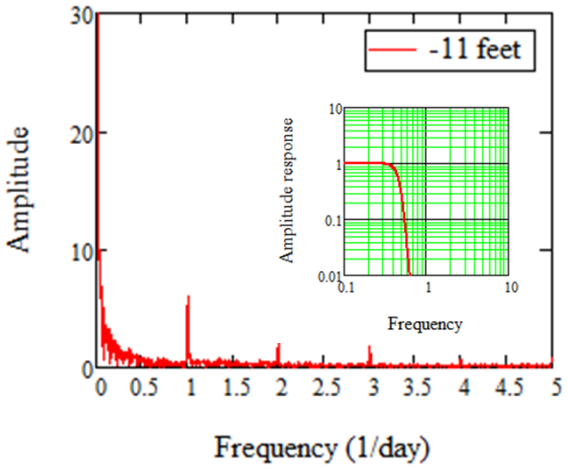
Department of Civil, Construction, and Environmental Engineering
North Carolina State University

Example: Removing noise from inclinometer readings with filter

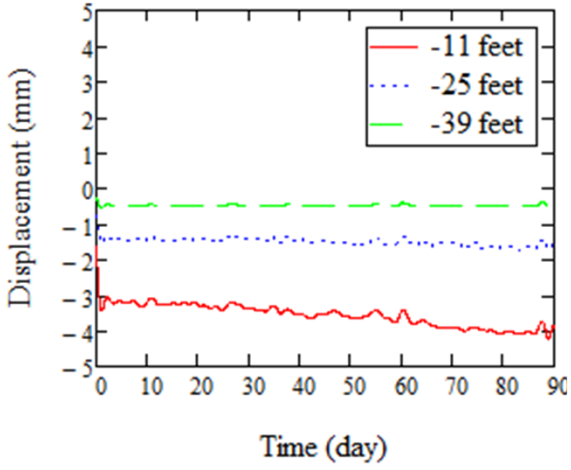
Unknown noise are presented in inclinometer readings taken at Lake Raleigh Dam at NCSU during 06/2011 to 08/2011.



Original signals in time domain



A signal in frequency domain and the low-pass filter applied



Signals in time domain after the application of filter

Frequency domain analysis helps to choose a low-pass filter with cut-off frequency = 1/day to reduce the high frequency noise from the original signals. Noise-free signals are obtained as shown above.

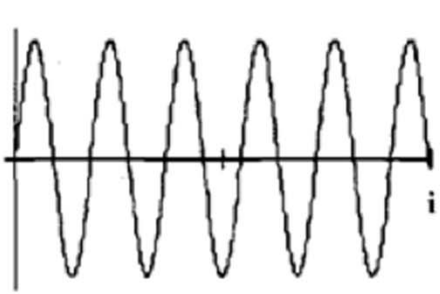
Course, Curriculum, and Laboratory Improvement: Integration of Sensor Technologies in the Civil Engineering Curriculum, DUE0837612

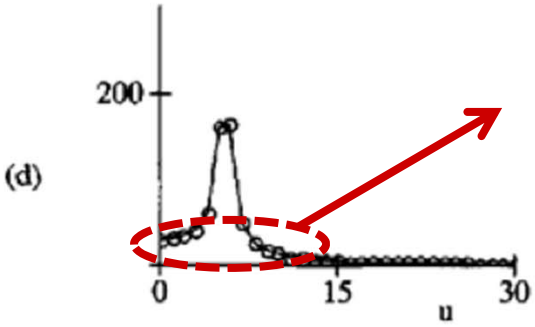
Leakage

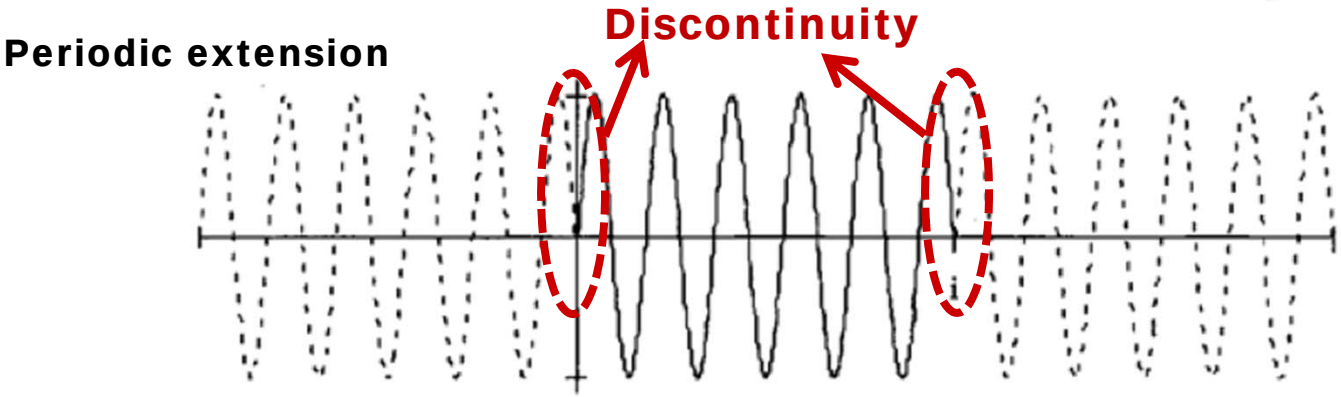
- Spectral leakage is the result of the inherent assumption in the DFT/FFT algorithm that the time record in a DFT/FFT segment is exactly repeated throughout all time and that signals contained in a DFT/FFT segment are thus periodic at intervals that correspond to the length of the DFT/FFT segment .

- If the time record in a DFT/FFT segment has a non-integer number of cycles, this assumption is violated and spectral leakage occurs

Leakage







(after Santamarina et al., 2005)

For an FFT segment with a non-integer number of cycles, there are discontinuities between successive segments after periodic extension, which introduces frequencies that are not present in the original signal.

Window Function

- The best approach to minimize leakage in practical applications is to multiply the time record by a suitable 'window' function before performing FFT.
- A window function generally has a unit gain at the center of the FFT segment, decreasing gradually to zero or a small value at both ends of the FFT segment, and becomes zero outside the FFT segment. Thus it greatly suppresses the discontinuity of the time record between FFT segments.

Window Function

- Triangle Window

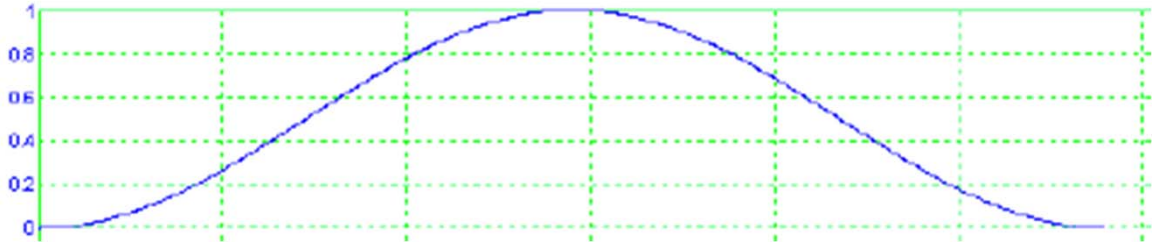
$$w_i = i/(N/2) \qquad i = 0,1,\cdots,N/2$$

$$w_i = 2/N/(N-i) \qquad i = N/2,\cdots,N-1$$

www.virtins.com

- Hanning Window

$$w_i = \sin^2(i \cdot \pi / N) = 0.5 - 0.5\cos(2 \cdot i \cdot \pi / N) \qquad i = 0,1,\cdots,N-1$$



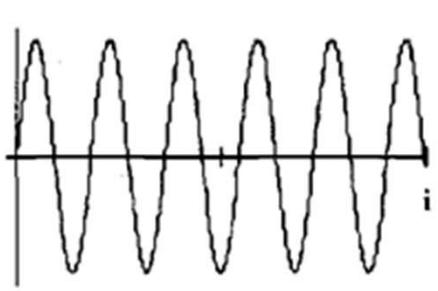
www.virtins.com

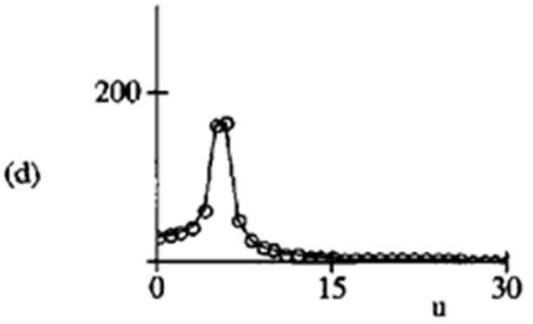
Window Function

- The windowed signal $x^{<win>}$ is obtained by multiplying the signal x with the window function w point by point:

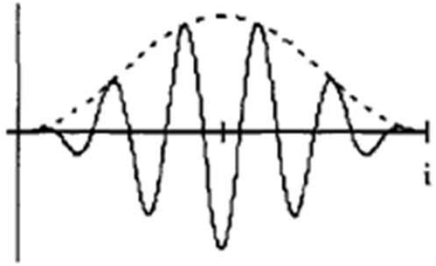
$$x_i^{<win>} = x_i \cdot w_i$$

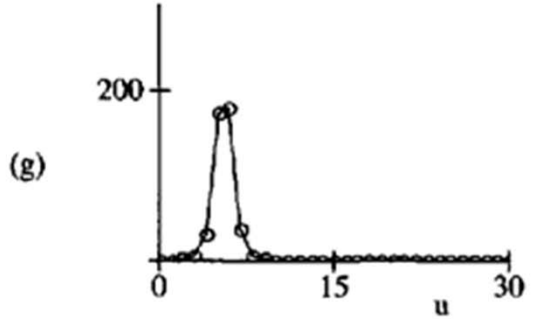
- Before





- After





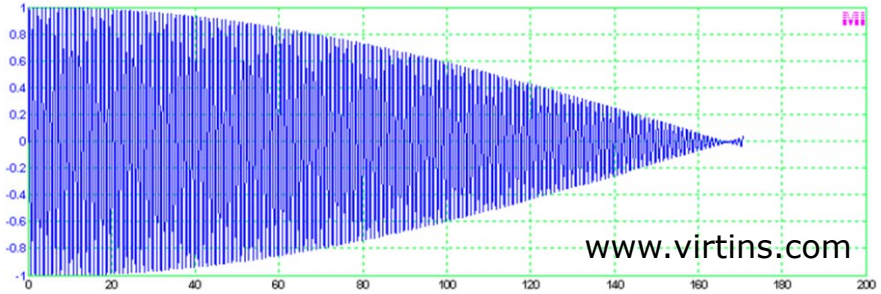
(after Santamarina et al., 2005)

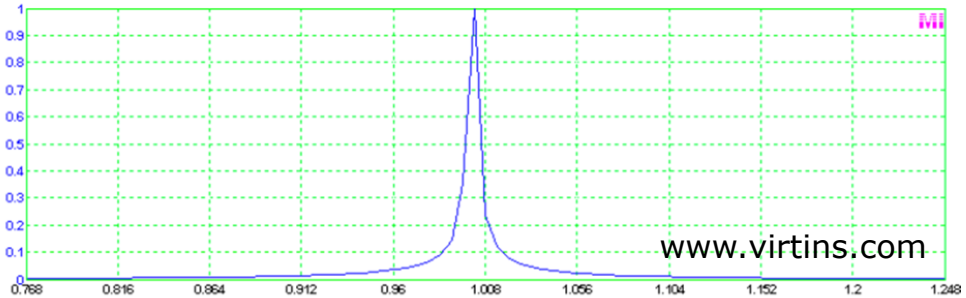
Frequency Resolution

- Frequency resolution is the smallest difference in frequency that can be distinguished. It is defined by:

  $\Delta f = 1/\text{sampling duration} = 1/(dt * \text{number of sampling points})$

Below is a signal in time domain with two components:



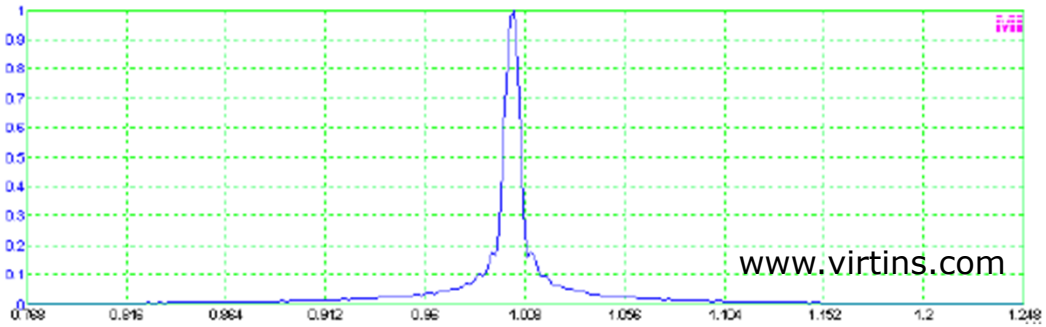


f_1= 1000 Hz, **f_2**= 1003 Hz;
dt = 1/48000 sec;
Sampling point: 8192;
FFT size: 8192

The **Frequency resolution** is:
48000/8192=5.86 Hz > 3 Hz
It can be seen that in the frequency domain, the two components can not be distinguished

Zero Padding and Frequency Resolution

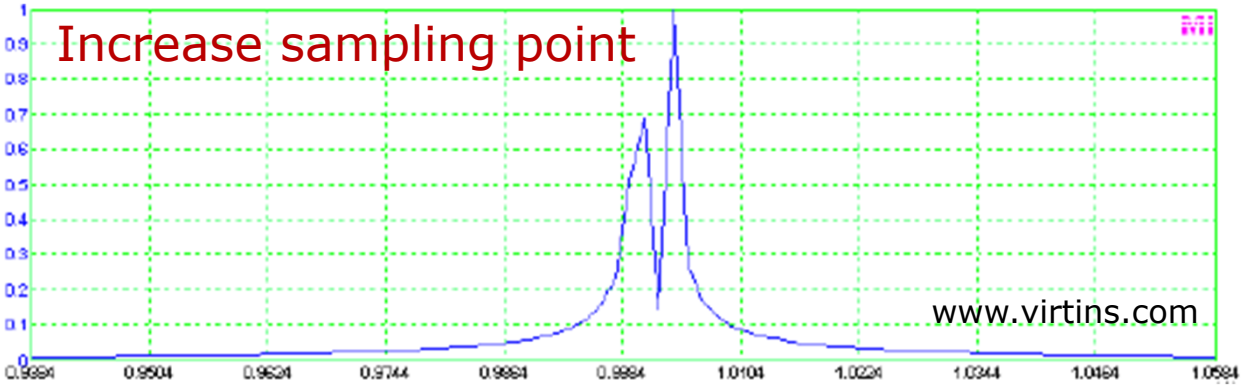
- Keep the sampling points = 8192 unchanged. Increase FFT Size to 32768. Apply Zero Padding.
  - **Apparent fft frequency resolution**: 1/(dt* signal points after zero padding) = 48000/32768 = 1.46 Hz < 3 Hz
  - **Real fft frequency resolution**: 1/ (dt* sampling points) = 48000/8192 = 5.859 Hz > 3 Hz (unchanged)



- It is shown that zero padding does not improve the real frequency resolution although it does provide more spectral lines via interpolation.

Increasing Frequency Resolution

- The real frequency resolution can be increased by increasing the sampling duration. There are two ways to increase the sampling duration
  - Increase the number of sampling points
  - Decrease the sampling frequency (increase dt)



The number of sampling points are increased to 32768. so the frequency resolution become:
1/(dt* sampling points) = 48000/32768 = 1.46 Hz < 3 Hz .
This is enough to resolve the two frequency components.

Summary and Conclusions

- This module is about frequency domain data analysis and in-depth understanding of physical phenomena behind monitoring data.
- Examples are provided based on data collected from the sensor network in NCSU Centennial Campus.
- Will you be able to answer these questions now?

Main References

1. Santamarina, J. C. (2005). *Discrete signals and inverse problems : an introduction for engineers and scientists.* Wiley, Hoboken, NJ.
2. *Virtins Technology*, (2009), FFT Basics and Case Study using Multi-Instrument.
3. Handouts and *Mathcad* scripts from Dr. Brina Montoya, NCSU.