Fourier Workbench

Ahmet Turkmen and Jon Breen
University of Wisconsin-Stout

Abstract

In this study, a Java-based program, Fourier Workbench, was developed. This software processes an audio input signal on the fly, displaying both the FFT (Fast Fourier Transform) and the DFT (Discrete Fourier Transform) of the signal at the same time. The software automatically updates these graphs at a user specified interval. This can be paused or updated manually as determined by the user. The time the calculations took for each algorithm is also displayed after every update. The software was designed especially for processing of musical instruments like a guitar. The software includes an option to display lines at the main frequency and harmonics of a well-tuned guitar note. The other options that the user can control include but aren't limited to, sampling frequency, the number of samples per transform, the displayed area on the graph and whether the DFT calculates the whole spectrum or just what is displayed. The main purpose of this program is to demonstrate the effect of these parameters on the resolution and speed of the transforms, and also to demonstrate the difference between the two algorithms. The software can be used for tuning musical instruments, for students' exercises in ECE introductory digital signal processing courses, and also for research in this field. In future studies, some additions can be made to the software to enable students to develop, apply and observe the effects of various filters to recordings.

Introduction

The Discrete time Fourier Transform (DFT) can be used to transform a sampled audio signal into a frequency domain representation. The frequency domain representation can be shown as graphs of variation of magnitude and phase angle with frequency. This representation is called a Bode plot. The Fast Fourier Transform (FFT) is an algorithm to implement the DFT process more efficiently. In this study, a Java-based interactive program has been developed to visualize the DFT and FFT of recorded audio signals. The software is an interesting application of the FFT and DFT algorithms to musical signals like those from an electric guitar. Some researchers have shown in their papers that signal processing techniques can be applied to musical recordings for analysis and synthesis.[1-2] This program is useful for comparing the FFT and DFT and also for exploring the effect of different sampling frequencies and sample sizes. In the future, perhaps the program will also assist in the design of real-time signal processing algorithms. This is still within the scope of an introductory digital signal processing (DSP) course.[3] First, the overall software system is presented in this article. The interface and user options are explained. Some of the results are reported, and usability of the program in an introductory DSP course is discussed.

Fourier Workbench Software

The block diagram of the overall system is shown in Figure 1. The audio/musical signal is processed on the fly. An electrical guitar is plugged into the sound card of the computer.

Sampling rate can be selected by the user between 4 KHz and 96 KHz. A certain number (N) of signal samples are kept in the buffer, which keeps only the most recent N samples. This number is selected by the user between 64 and 8192. The user can obtain magnitude plots of the DFT and the FFT of the signals on the display with the software. The sampling rate and number of samples determine resolution and maximum frequency of the transform.

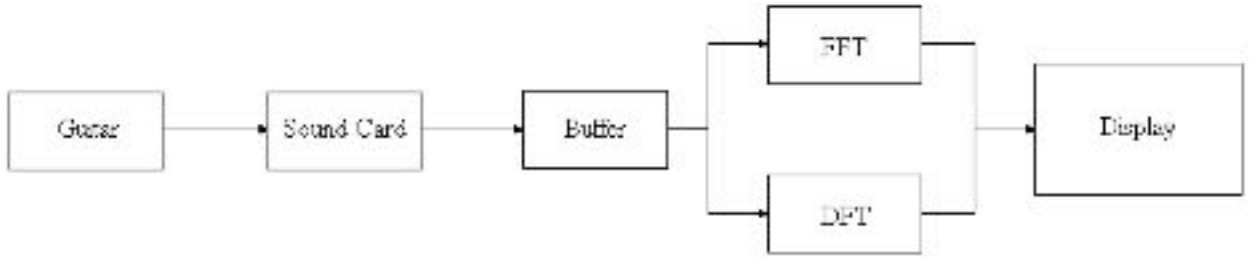


Figure -1. Block diagram of the system.

The Discrete Fourier Transform (DFT) is used to transform a time domain signal x[n] sequence to a frequency domain X(k) sequence.

The frequency response of a sequence and its DFT are related as expressed in equation 1.

$$X(k) = X(e^{j\Theta}), \text{ where } \Theta=2\pi k/N,\ k=0, 1, 2, \ldots, N-1 \tag{1}$$

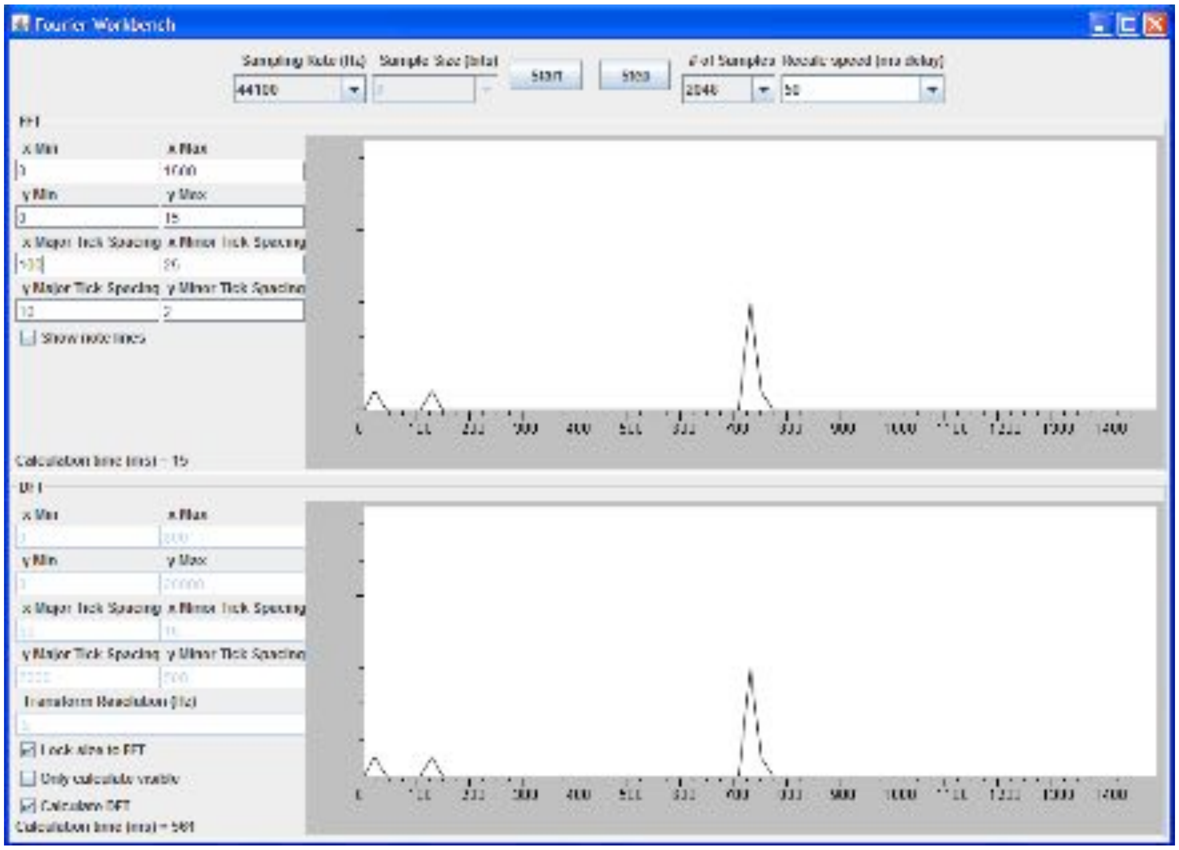
The elements of X(k) as obtained from this equation are at discrete frequencies spaced $2\pi/N$ apart. This is the natural spacing of the transform, which doesn't need time information to perform properly, but to find actual frequencies, we need time information. This is contained in the true frequency spacing of the transform, Δf, as defined in equation (2). Multiplying k by Δf yields the frequency of that point on the transform in Hz.

$$\Delta f = 1/T_0 = f_s/N \tag{2}$$

In this equation, T_0 is the length of the recorded signal in seconds, f_s is the sampling frequency of the original signal in Hz, and N is again the number of samples in the signal being transformed.

The direct computation of the DFT requires a large number of complex multiplications. The FFT algorithms are employed to compute the DFT much more efficiently. These algorithms use a power of 2 points and exploit the periodic nature of the complex exponential $e^{j2\pi nk/N}$ occurring in the DFT equation. For a 256-point DFT computation, direct DFT computation requires 65536 complex multiplications. However, in a FFT based computation of the same number of points, 1024 multiplications are needed. This is a 64-times reduction. For a 512-point DFT computation, direct DFT computation and FFT based computation require 262144 and 2048 multiplications respectively. This corresponds to a 128-times reduction.[4-5]

The Java-based software developed in this study allows a user to obtain frequency vs. magnitude components for audio signals. The interface of the program is shown in Figure 2. The user can control a number of options not yet mentioned. One group of options involves what is graphed. This includes axis maxima and minima, axis tick spacing, and whether to make the DFT and FFT plot the same space or not. The user can also choose to display lines on the graphs at every point a 22-fret electric guitar in standard tuning will have the fundamental frequency of a note as seen in Figure 3. Other options are specific to the DFT. Because of the nature of the DFT algorithm, it is possible to calculate only a portion of what the FFT calculates. This selectivity is granted to the user in the form of a check box. This gives the user the choice of calculating everything the FFT is calculating or just what the graph is displaying. The user can also choose not to calculate the DFT at all, as sometimes only the FFT is desired and the DFT could bog the system down. Finally, the user may choose to have the software automatically recalculate the transforms and update the graphs. If this option is chosen, recalculation speed may be selected between 50 and 1000 ms.



Sampling Rate (Hz): 44100
Sample Size (bits):
# of Samples: 2048
Recalc speed (ms delay): 50

FFT

x Min	x Max
0	1000
y Min	y Max
0	15
x Major Tick Spacing	x Minor Tick Spacing
100	25
y Major Tick Spacing	y Minor Tick Spacing
10	2

Calculation time (ms) = 15

DFT

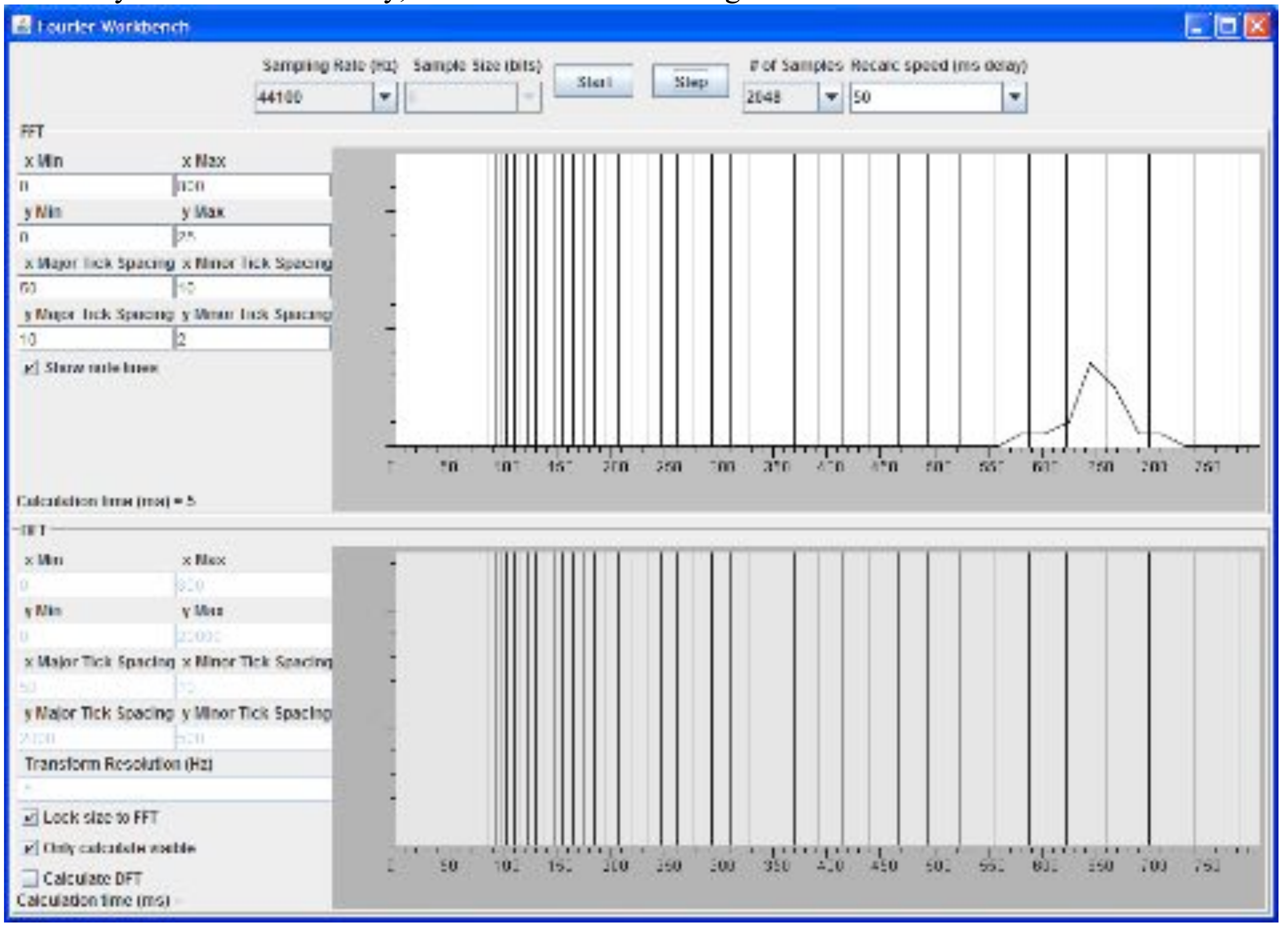
x Min	x Max
0	800
y Min	y Max
0	20000
x Major Tick Spacing	x Minor Tick Spacing
[illegible]	[illegible]
y Major Tick Spacing	y Minor Tick Spacing
2000	500
Transform Resolution (Hz)	
[illegible]	

[✓] Lock size to FFT
[ ] Only calculate visible
[✓] Calculate DFT
Calculation time (ms) = 561

Figure -2: The Fourier Workbench interface

As far as actual numbers go, there's no difference between the FFT and DFT. Given the same series of time domain samples, they will produce exactly the same numbers for the frequency domain representation (see Figure 2). The only difference as far as input/output comparison goes is that the FFT calculates the whole transform (all calculable frequencies) whether it is needed or not, and the DFT can be used to calculate specific points, or a range, within the transform.

The most significant difference between the two algorithms, when it comes to application, is the speed. The FFT is significantly faster than the DFT when sample size (N) is large. The DFT has a runtime, in Landau notation, of $O(N^2)$ and the FFT has a runtime of $O(N\log_2 N)$. Since this program competes for processor time with everything else on the computer, this relationship obviously won't match exactly, but there is still a striking resemblance.



Sampling Rate (Hz) 44100
Sample Size (bits)
# of Samples 2048
Recalc speed (ms delay) 50

FFT
x Min 0 x Max 800
y Min 0 y Max 25
x Major Tick Spacing 50 x Minor Tick Spacing 10
y Major Tick Spacing 10 y Minor Tick Spacing 2
[✓] Show note lines
Calculation time (ms) = 5

DFT
x Min 0 x Max 800
y Min 0 y Max 25000
x Major Tick Spacing 50 x Minor Tick Spacing 10
y Major Tick Spacing 2000 y Minor Tick Spacing 500
Transform Resolution (Hz)
[✓] Lock size to FFT
[✓] Only calculate visible
[ ] Calculate DFT
Calculation time (ms) =

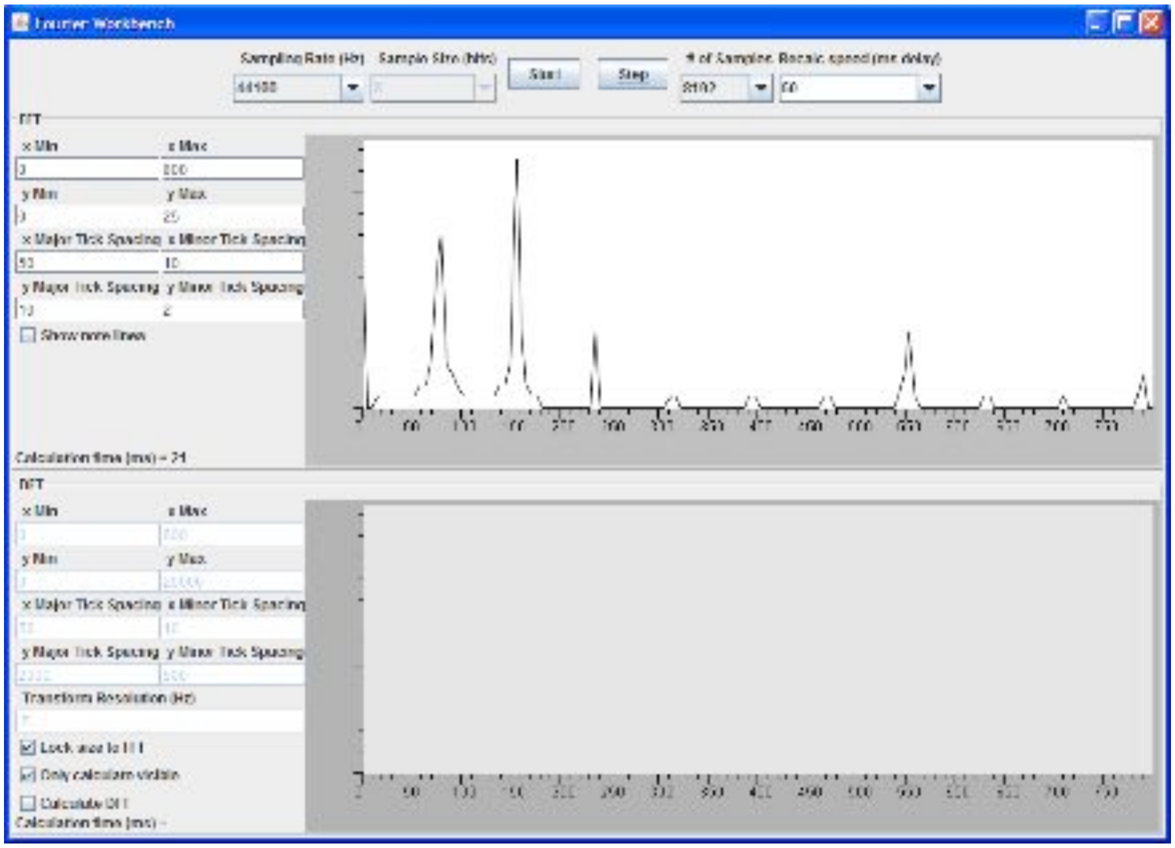
Figure -3: Note lines output

One performance issue, when comparing the DFT and FFT, as mentioned, is the fact that the DFT will allow the user to choose a specific range (whatever is selected as the DFT graph range) of frequencies to calculate. This changes the relationship denoted above because the algorithms are no longer calculating the same thing and the DFT's runtime is now O(N*nk) where nk represents the number of discrete frequencies calculated. From this, it can be concluded that the DFT will only be faster than the FFT when it is calculating less than $\log_2 N$ discrete frequencies. In the case of a 2048 point transform, this means 10 or less discrete frequencies. This time comparison can readily be demonstrated with the Fourier Workbench software.

Different sample sizes and sampling rates:

Sample size (N) and sampling rate (f_s) are important considerations when applying a Fourier transform algorithm. Together, they determine how long it will take to record the data needed for the transform and frequency domain resolution. Sampling frequency will also determine the maximum frequency which can be calculated. This is provided by the Nyquist sampling theorem which states that a sampled signal can't represent a frequency higher than half of the sampling frequency.

The resolution of the transform is determined by $\Delta f = f_s/N$, as mentioned earlier. As can be seen, a higher sampling frequency lowers the resolution of the frequency domain representation of the signal, but a higher sample size increases the resolution. When applied to musical notes, this linear distribution of discrete frequencies is important because musical notes do not distribute themselves at equal intervals. Musical notes are typically denoted A, B, C, D, E, F, or G. This represents one octave of notes and there are several octaves within human hearing range. The octave a note is in is generally denoted by a number after the note letter. A3 is 220 Hz, A4 is 440 Hz, and A5 is 880 Hz. Note that there is the same number of notes between A3 and A4 as there is between A4 and A5 but the number of Hz has doubled. This doubling of frequencies between octaves correlates to the human ear's ability to detect changes in frequency. Because of this relationship, a given frequency-domain resolution may be sufficient to tune an instrument with high notes, but not one with low notes. The Fourier transform may have higher resolution than the human ear at those high frequencies but lower resolution than the human ear for low frequencies.



Sampling Rate (Hz) 44100
Sample Size (bits) [illegible]
# of Samples 8192
Repaint speed (ms delay) 50

x Min 0 x Max 800
y Min 0 y Max 25
x Major Tick Spacing 50 x Minor Tick Spacing 10
y Major Tick Spacing 10 y Minor Tick Spacing 2
Calculation time (ms) = 24

x Min 0 x Max 800
y Min 0 y Max 25000
x Major Tick Spacing 50 x Minor Tick Spacing 10
y Major Tick Spacing 2000 y Minor Tick Spacing 500
Transform Resolution (Hz) [illegible]
☑ Lock axes to FFT
☑ Only calculate visible
☐ Calculate DFT
Calculation time (ms) -

Figure -4: Visualization of a signal with components at several frequency values

Multi-tone signals:

Signals, in the real world, don't happen as pure sine waves. Most musical instruments, for example, produce a unique combination of tones that distinguish one instrument from another. The tone quality of instruments that makes them unique is called timber. This word simply describes a profile of overtones. An overtone is a frequency that is any integer multiple of the fundamental frequency. So any given note has a potentially infinite series of evenly-spaced overtones. The note E2 (82.41 Hz) on an electric guitar is shown in Figure 4. Notice all the "bumps" in the graph beyond the fundamental frequency; these are the overtones. The ratio of intensities between the fundamental frequency and its overtones is what we call timber. This is just one example of a signal that is more complicated than a sine wave, but the world is full of them. Viewing real world signals in the frequency domain is often useful in understanding the signal, the system that produced it, and how one might process the signal for a certain purpose.

Results and Conclusions

The Fourier transform is an incredibly useful construct, but for many students, it can be difficult to understand. For most, a change in one number, say sampling rate, isn't easily followed through the math to discover a property of the Fourier transform. Visual demonstration and the chance to experiment easily and quickly can make the material much more accessible to students.

Processing speed of different Fourier transform algorithms can also be better appreciated if those algorithms can be compared side-by-side. Table 1 and Figure 5 show how long it took the author's computer to calculate the FFT and DFT with different sample sizes in the Fourier Workbench program.

N	FFT time (ms)	DFT time (ms)
64	Less than 1 ms	1
128	Less than 1 ms	2
256	1	10
512	2	37
1024	2	150
2048	5	572
4096	10	2298
8192	23	9324

Table-1: FFT and DFT processing durations

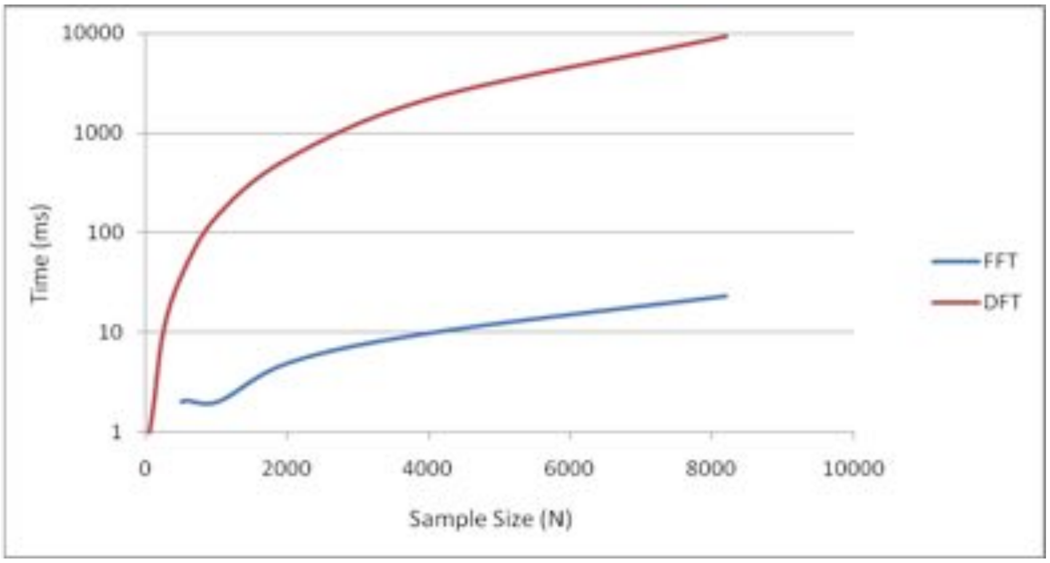


Figure -5: FFT and DFT processing durations

Fourier Workbench offers a chance to view the results of the FFT or DFT while hearing the sound that it relates to. The affects of changing sampling rate and sample size can immediately be seen, and speed comparisons between the two algorithms can easily be made. This can be a valuable tool for students learning about the Fourier transform and its applications, and also for learning about the frequency composition of different sounds, which is important for audio processing and music theory.

References

1. Karjalainen, M.; Penttinen, H.; Valimaki, V.; "Sound from the Electric Guitar Using DSP Techniques" , in *Proc. IEEE Conf. on Acoust. Speech, and Sig. Proc*., pp.11773-11776, June 2000
2. Bradley, K.; Mu-Huo Cheng; Stonick, V.L.; "Automated analysis and computationally efficient synthesis of acoustic guitar strings and body", *IEEE ASSP Workshop on* Appl. Sig. Proc. to Audio and Acoustics, 1995., 15-18 Oct. 1995 pp.238 - 241
3. Mauro J. Caputi,; Developing real-time digital audio effects for electric guitar in an introductory digital signal processing class, *IEEE Trans. On Education, Vol. 41(4), pp. 10, 1998*
4. Avtar Singh, S. Srinivasan. *Digital Signal Processing.* Belmont, CA 94002: Brooks/Cole - Thomson Learning, 2004.
5. Alan V. Oppenheim, Allan S. Willsky, S,Hamid Nawab, *Digital Signal Processing, 2nd edition* : Prentice Hall, 1996

JON BREEN
Jon Breen has a BS degree from the University of Wisconsin - Stout and is expected to earn an MS degree from University of Wisconsin - Madison in 2011. His studies have ranged from industrial automation and robotics, to manufacturing systems, power electronics and motor control. His recent work focused on hybrid vehicle systems and control. His e-mail is jlbreen@wisc.edu

AHMET TURKMEN, Ph.D.
Ahmet Turkmen, Ph.D. is an associate professor at Engineering and Technology department of University of Wisconsin – Stout. His fields of interest are medical instrumentation, modeling of cardiovascular system, processing of physiological signals and engineering education. His e-mail is turkmena@uwstout.edu