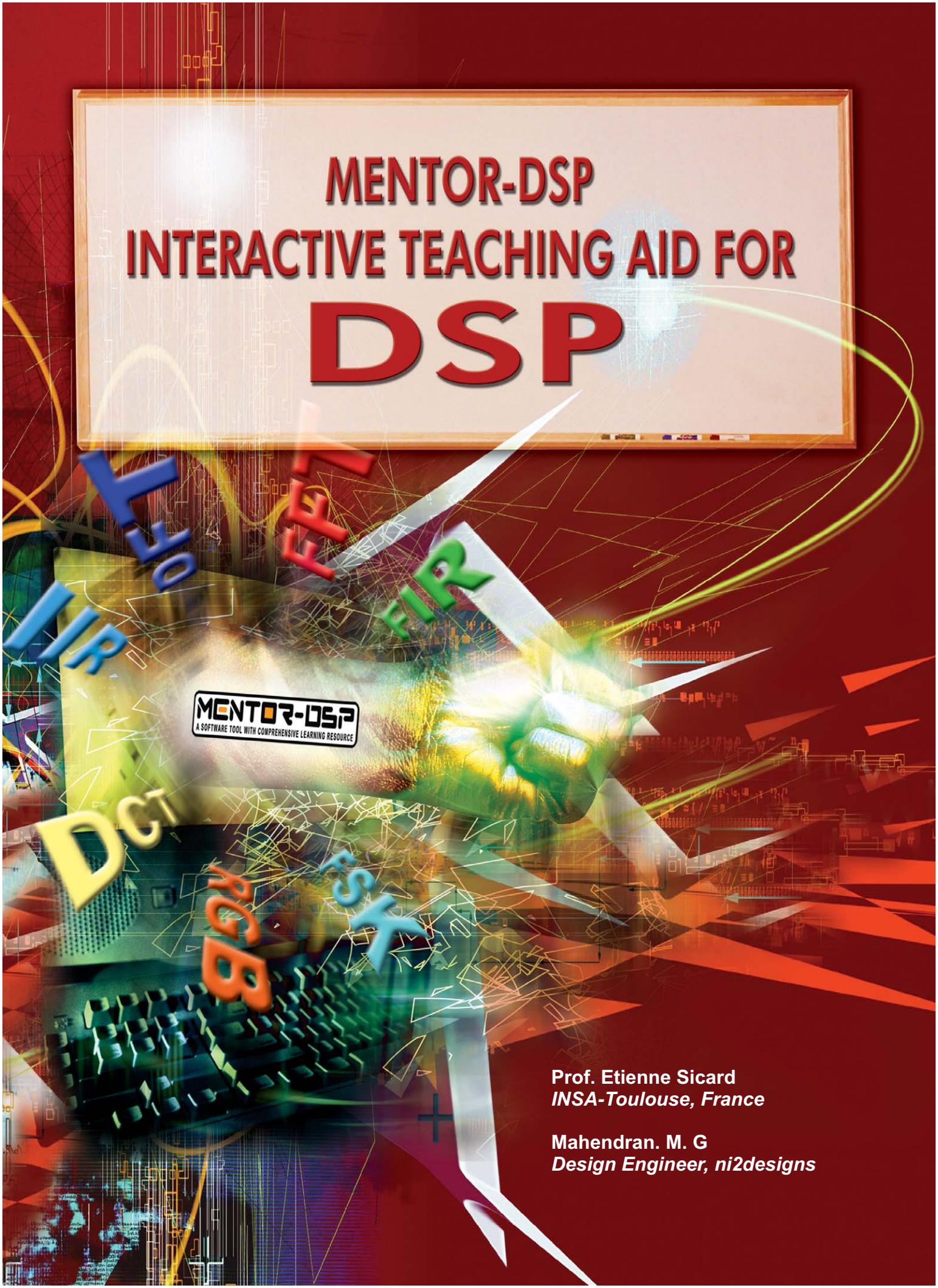
MENTOR-DSP
INTERACTIVE TEACHING AID FOR
DSP



Prof. Etienne Sicard
INSA-Toulouse, France

Mahendran. M. G
Design Engineer, ni2designs

Some years ago, Digital Signal Processing (DSP) was viewed as one of the most abstract and conceptually difficult areas in an electrical engineering curriculum. The rigid lecture and text book structures in some of the Digital Signal Processing (DSP) classes often failed to connect the concepts with applications. Development and use of engineering softwares such as MatLab or SciLab running on personal computers have greatly impacted the DSP curricula thanks to easy plotting of functions, and implementation of numerical computing algorithms. These numerical computational packages are based on a high level programming language which condense many single computations into one line of code.

These softwares are widely used for signal and image processing. Numerous courses are available, based on the high-level language functions, in combination with hardware platforms with semi-automatic implementation of code. However, such tools are poorly suited to interactive courses and require a good knowledge of the language prior to any experiment.

Furthermore, the tool installation and deployment is not an easy task due to the inherent complexity of the package and do not fit with the usual need for simple and illustrative software focused on DSP fundamentals. In parallel, educational tools have been developed with focus on DSP education using a variety of approaches, such as J-DSP, an object-oriented programming environment that enables students to perform interactive computer simulations on the Internet. However, most of these tools either have rigid canvas with predefined demonstrations, or limited functionalities in terms of real-time sound or image processing.

MentorDSP is an interactive educational software package focused on the illustration of digital signal processing. MentorDSP is focused on the illustration of DSP fundamentals such as waveform synthesis and manipulation, Fourier transform, convolution, correlation, filter design and image processing. The software includes real-time features such as sound processing, image capture and interface with a specific DSP hardware platform. It is simple, intuitive and straightforward to use, targeted to students in the field of digital signal processing.

This article describes the general philosophy of the MentorDSP tool and details some key features of DSP algorithm illustration.

Getting Started with MentorDSP

The initial screen of the tool is reported in figure.1. Seven sections are listed: “Getting Started”, “Basic Theory”, “Fourier Transform”, “Convolution”, “Filtering”, “Modulation” and “Image”.

Figure. 1: MentorDSP initial screen.

Real-Time Features

The item “Real-Time-Sound” is a good introduction to the MentorDSP philosophy. At a press of this feature in the lab practical list, the screen is organized as follows (Figure.2). The upper screen is the time-domain representation of the sound. The sound waveform is captured in real-time from the computer microphone and appears as a series of samples (512 by default).

The waveform corresponds to a human voice (male pronouncing the vowel [a]). Figure 2 shows the real-time spectrogram of the voice. The Fourier Transform is used to compute for each set of 512 samples the frequency-domain components of the input waveform. The frequency is the Y axis while the X axis represents the time. The red spots around 100 Hz correspond to high energy contents while the blue color corresponds to very low energy.

This kind of tool has a very impressive impact on students when illustrating the concept of Fourier Transform. The theory of Fourier transform corresponds to the following equations. The captured sound *x(t)* may be represented as a sum of sine and cosine components with increased frequency(the y-axis):

$$x(t) = A_0 + \sum_{n=1}^{\infty} (A_n \cos 2\pi n \frac{t}{T} + B_n \sin 2\pi n \frac{t}{T})$$

Where:

$$A_0 = \frac{1}{T}\int_0^T x(t)dt$$ (mean value of x(t) in the interval T), ignored in Fig. 2

$$A_n = \frac{2}{T}\int_0^T x(t)\cos(2\pi n \frac{t}{T})dt \quad (n>0)$$

$$B_n = \frac{2}{T}\int_0^T x(t)\sin(2\pi n \frac{t}{T})dt \quad (n>0)$$

The color palette corresponds to the evaluation of

$$C_n = \sqrt{A_n + B_n}$$

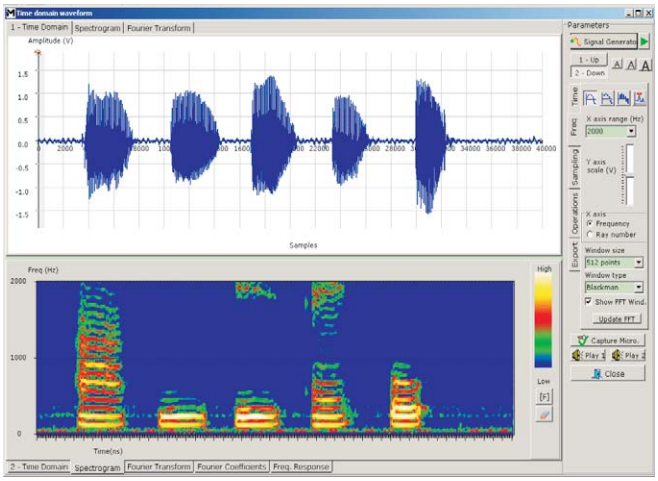


Figure. 2: Real-time Fourier Transform of the sound captured by the Microphone (pronunciation of vowels).

The underlying educational target is to develop critical thinking and help students to connect the formal equations of the Fourier series with its practical application. Students are asked to produce a variety of sounds for which the theory is well-know (pure sinusoidal wave, Dirac, etc...). The example of a Dirac pulse is illustrated in figure 3. The Dirac pulse is an infinitely narrow and tall pulse. The Dirac area is 1. In signal processing, we often use “e” as width and “1/e” as height, with “e” very small (Figure. 3-a). The Fourier Transform is constant in amplitude and spread over the whole spectrum (Figure. 3-b). The pulse produced by a small shock on a table, close to the computer, generates a saturated signal as seen in figure. 3-c, with very complex resonance effects due to the sound propagation in the classroom. The spectrogram shown in Figure. 3-d is close to the expected theory with a spread of energy (white color) all over the frequency domain (y-axis).

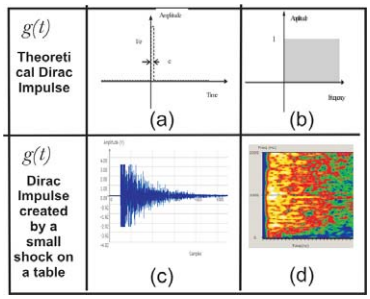


Figure. 3: Theoretical and experimental Dirac pulse.

Data Coding and Sampling

In MentorDSP, a specific lab practical has been focused on the format used in digital signals. Not only floating and fixed point formats are important, but also specific functions such as rounding or saturation. The screen (See Figure. 4) enables the following features:

•Generation of a real, binary, hexadecimal, fixed-point and floating-point number.

•Vary the desired precision from 4 to 16 bits.

•Make the conversion between the Real value and the Binary notation.

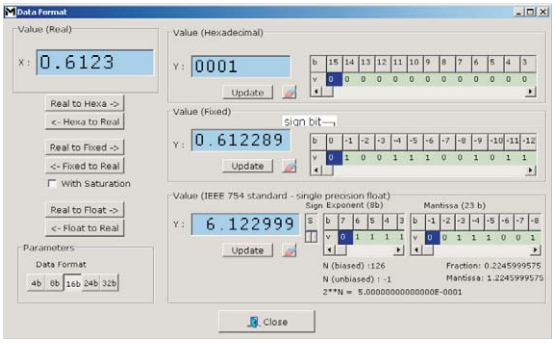


Figure. 4: Illustration of Data Coding.

The two classes of data formats are the integer and real numbers (Figure 5). The integer type is separated into two formats: unsigned format and signed format. The real numbers are also sub-divided into fixed point and floating point descriptions. Each data is coded in 8,16, 24 or 32 bits.

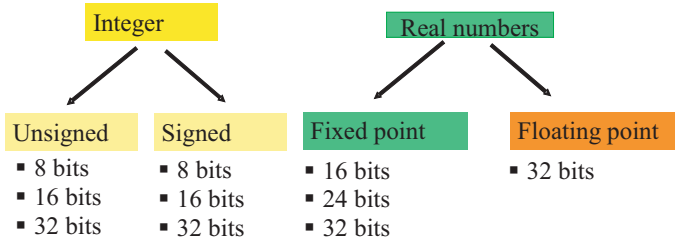


Figure 5: Most common data formats used in ASIC designs.

In digital signal processing, real numbers are often implemented as fixed point numbers. The key idea is to restrict the real numbers within the range [-1.0..+1.0] and to use arithmetic hardware that is compatible with integer hardware. More general real numbers are coded in a 32 bit format. A summary of real formats is reported in table 1.

Type	Size (bit)	Usual name	Range
Fixed point	16	Fixed	-1.0..+1.0 (Minimum 0.00003)
	32	Doubl e fixed	-1.0..+1.0 (Minimum 0.00000000046)
Floating point	32	Real	$-3.4^{e}38..3.4^{e}38$ (Minimum 5.8^{e}-39)

Table 1: Size and range of usual real formats.

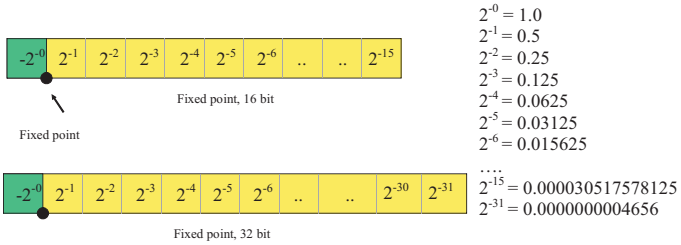


Figure 6: Fixed point numbers in 16 and 32 bit format.

In the case of fixed point arithmetic, we read bits as fractions in negative power of 2 (Refer Figure 6). Two examples are illustrated below (Eq. 1 and 2).

$$01100100 = 2^{-1} + 2^{-2} + 2^{-5} = 0.5 + 0.25 + 0.03125 = 0.78125$$

(Equ. 1)

$$11100100 = -2^{0} + 2^{-1} + 2^{-2} + 2^{-5} = -1.0 + 0.5 + 0.25 + 0.03125 = -0.21875$$

(Equ. 2)

Floating Point Format

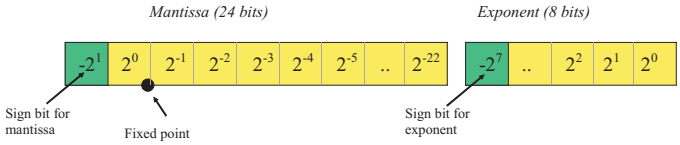


Figure 7: Floating point arithmetic format.

Finally, floating point data is coded using a mantissa multiplied by an exponent (Figure 7). The general formulation of the real number format is as given in Equ. 3. The illustration of this format is given in equation 4. Numbers may range from -3.4×10^{38} to 3.4×10^{38}.

$$\text{data} = \text{mantissa}.2^{\text{exponent}}$$

(Equ. 3)

$$\begin{aligned}(0110100...)(0..101) &= (2^{0} + 2^{-1} + 2^{-3})x(2^{2} + 2^{0}) \\ &= (1.0 + 0.5 + 0.125)x(4 + 1) \\ &= 1.625x2^{5} \\ &= 52.0\end{aligned}$$

(Equ. 4)

The PIC24, dsPIC30F and dsPIC33FJ are 16-bit chips that have fixed-point built-in capabilities but no floating point hardware. The standard IEEE-754 floating-point format is emulated by functions from the math library called by the compiled C code, such as floating “add”, “subtract”, “multiply” and “divide”. MentorDSP may be connected to a hardware platform based on dsPIC component. All the code generated by MentorDSP uses the “fixed-point” format for simplicity's sake.

Sampling

The illustration of sampling in MentorDSP is done by loading or generating a signal and observe the degradation of the signal quality when changing the time resolution (X axis) or the sampling resolution (y- axis). The sub-screen which controls the sampling parameters may be seen at the right of figure 8. Modifying the resolution from 32 bit downto 4 bit significantly changes the aspect of the signal. When selecting the microphone as input waveform, we see the real-time capture of the sound on the top screen and the effect of sampling (X and Y axis) on the lower curve.

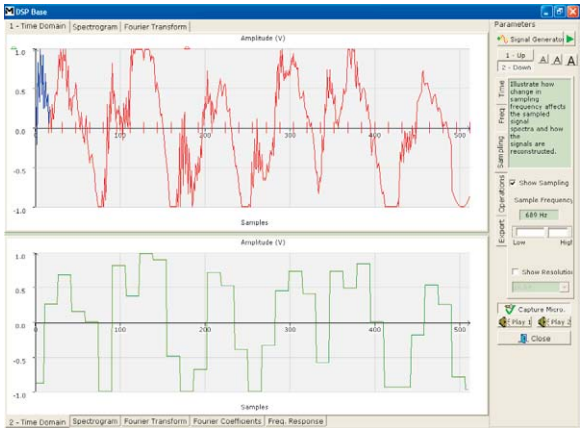


Figure 8: Illustration of sampling on a real-time sound captured by the microphone.

Aliasing

MentorDSP proposes several possibilities for generating signals. One of the most powerful way of creating a signal is to describe its equation using basic functions (sine, cosine, etc..), predefined constants (pi) and main variables (t for time). A chirp signal is generated using the equation shown in figure. 9. If the frequency reaches more than half the sampling frequency, the aliasing effect appears, as shown in the spectrogram of the signal (Figure. 10). The students try to establish the link between the experimental values for the signal description and the theoretical formulations of the limit frequency. The resulting signal may be heard, which gives the sensation of a decreasing pitch instead of an expected steady increase of the signal tone.

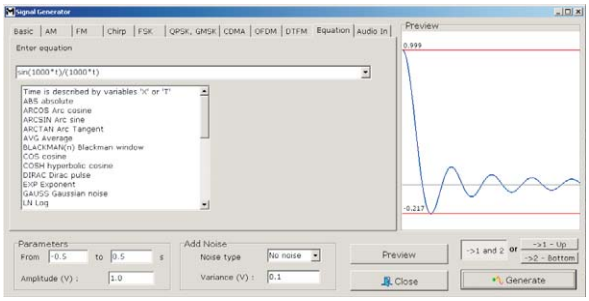


Figure 9: Generating a signal from its formal equation.

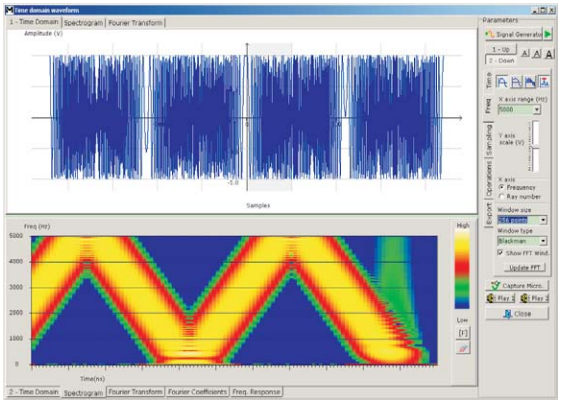


Figure 10: The illustration of aliasing.

Correlation

Correlation has been the focus of particular efforts to give the teacher ways to illustrate the core idea and step-by-step illustration of the theory, recalled in Equ. 5.

$$x(t) * y(t) = \int_{-\infty}^{+\infty} x(\tau)\, y(t-\tau)\, d\tau \qquad \text{Equ. 5}$$

MentorDSP proposes to compute the convolution between two very simple signals, i.e. simple pulses X[n] and Y[n], as shown in figure. 11. The application of Equ. 5 in this case should give a triangle. The idea is to display the instant value of the convolution and to observe the construction of the triangle. Once understood on this simple case, convolution may be applied to any other signal.

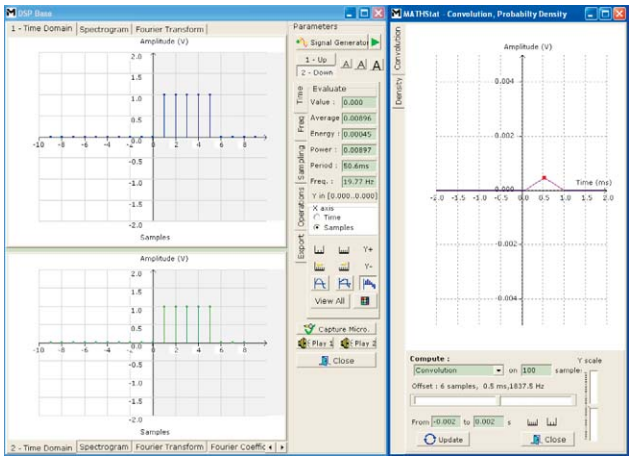


Figure 11: Step-by-step convolution between X and Y.

The auto-correlation consists in applying a similar algorithm as described by Equ. 5 except that X and Y are identical. This technique is applied on sound for fundamental frequency extraction. The student may see the result of real-time auto-correlation and determine the fundamental frequency as the frequency corresponding to the first maximum in time domain, as reported in figure. 12.

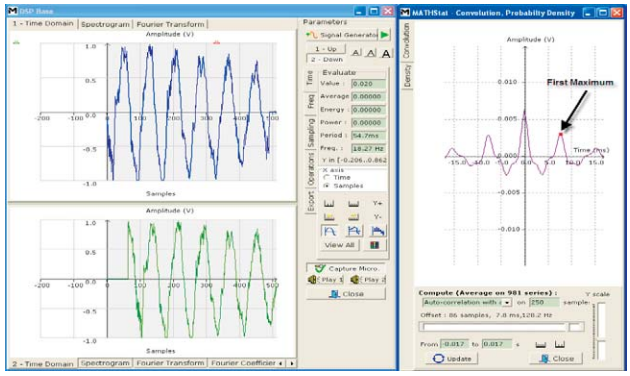


Figure 12: Real-time autocorrelation showing the main period of the signal. This technique is applied to extract the fundamental frequency of the voice in many voice coders.

Another attractive experiment proposed in MentorDSP concerns the autocorrelation of noisy signals. A variety of noises may be added using the equation generator, as shown in figure. 13. An illustration of 2D correlation may also be found in the image section.

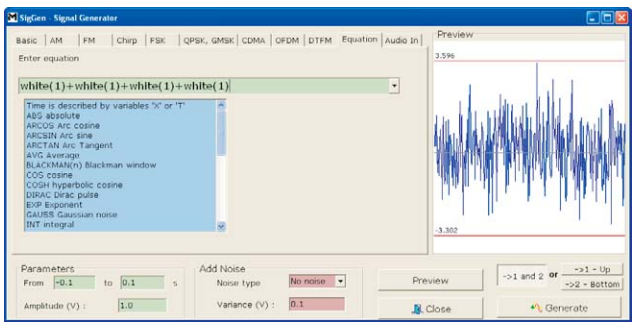


Figure 13: Creating signal with noise (White noise in this example).

Fourier Transform

Introducing Fourier series

The first introduction to Fourier Transform has been proposed in the section “Getting Started”. Although the combination of real-time sound and spectrogram on a single screen is extremely rich in information, captures the attention of students and stimulates in-depths questions related to Fourier theory, it is also important to ensure that the computation of Fourier series is well understood by students. For this purpose, the screen shown in figure. 14 is proposed. Real-time capabilities are not required. Instead, an immediate access to various basic signals eases the explanation in front of students, so that the link between theory and computed An and Bn may be established at any moment of the DSP course. For example, the clock signal should have even An equal to 0 and odd An amplitude reduced by approximately SQRT (2).

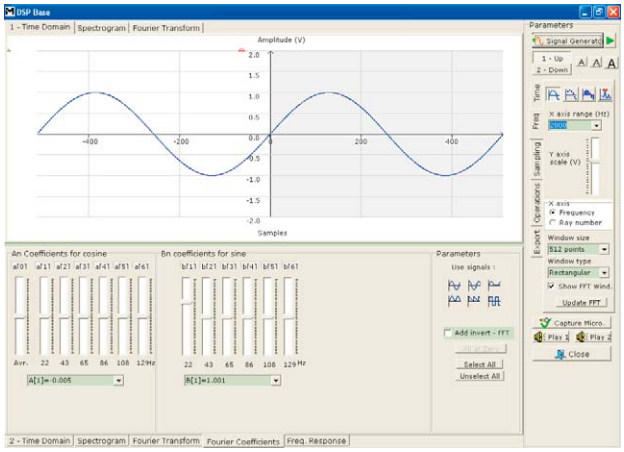


Figure 14: Link between basic signals and Fourier Coefficients An and Bn.

The student selects a desired signal. The Fourier series are computed and the cursors corresponding to A[n] and B[n] are updated. The exact value of these coefficients is also displayed as a list. The Fourier window (Blackman, Hamming, Rectangular) is user's accessible, as well as the window size (128 to 4096 points). A sinusoidal wave with frequency exactly fitted with the Fourier Window is generated at initialization to limit the confusion between all parameters. When changing the window from “Blackman” to “Rectangular”, a numerical noise appears.

Invert Fourier Transform

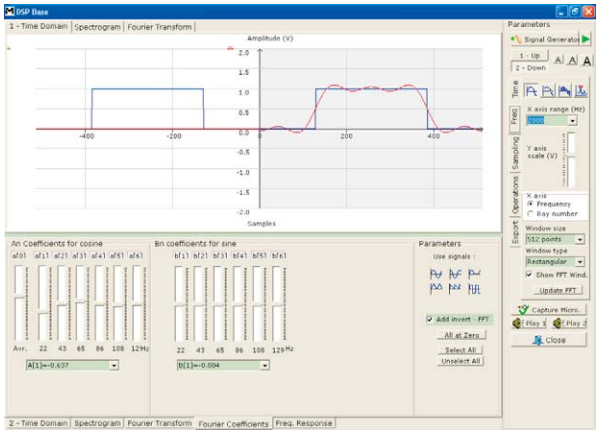


Figure 15: Illustration of Fourier and Invert Fourier Transform.

The Invert Fourier Transform is probably understood best by enabling the student to act on the cursors A[n] and B[n], and reconstruct the wave by applying Equ. 6. The reconstruction of a square wave is illustrated in figure. 15.

$$x(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos 2\pi n \frac{t}{T} + b_n \sin 2\pi n \frac{t}{T}) \qquad \text{Equ. 6}$$

Filtering

Introducing Recurrence Equations

$$y[n] = \sum_{i=1}^{N} a_i y[n-i] + \sum_{i=0}^{M} b_i x[n-i] \qquad \text{Equ. 7}$$

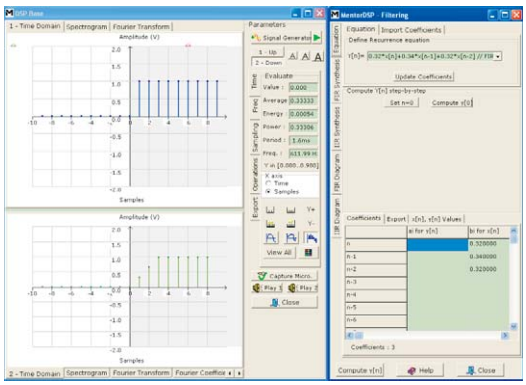


Figure 16: Filter window in Mentor DSP.

Refer Figure 16 for Filter Window in Mentor DSP tool. One general form for a recurrence equation is given by Equ. 7. Its intrinsic power for filtering is far for being immediate for students. The interface used in MentorDSP gathers the usual time-domain aspect of the signal “before” (upper window) and “after” (lower window) applying the recurrence equation. The student can enter its own recurrence equation, see the block diagram of Recursive and Non-recursive filters, and performs the step by step computation of X[n]. Other X[n] signals may also be investigated through the signal generator menu. Both the input X[n] and output Y[n] are displayed as discrete values. Refer figure. 17.

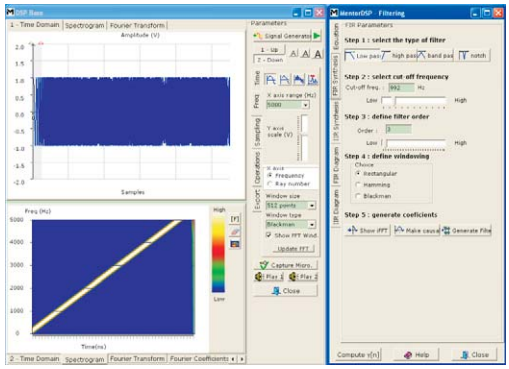


Figure 17: Applying a recurrence equation on a simple signal.

In the screen dedicated to Finite-Impulse-Response (FIR) filter design, the user selects the type of filter (low pass, high pass, band pass, notch), the cut-off frequency and filter order. The coefficients are then automatically computed. The result on a chirp signal may be observed (variation of the amplitude of the Y[n] signal depending on frequency) in figure. 18. The low frequency contents are not attenuated, while the frequencies above 1000 Hz are reduced.

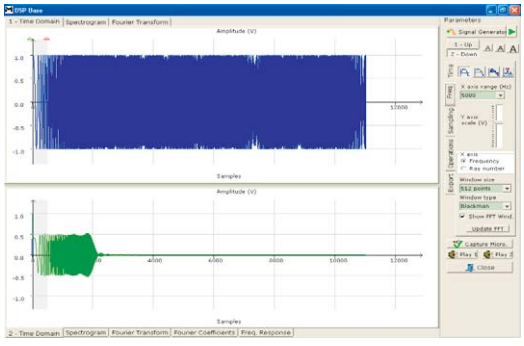
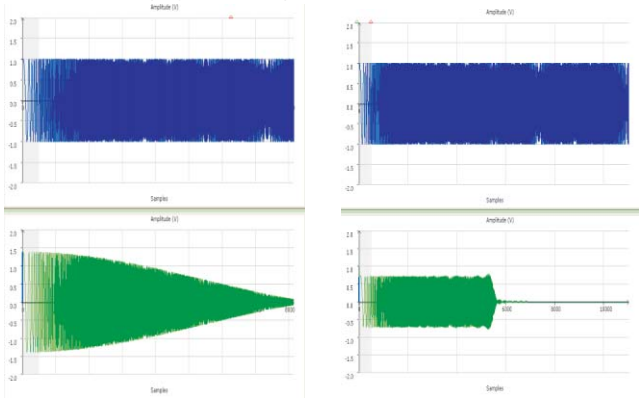


Figure 18: Applying a recurrence equation on a simple signal.

Pre-computed filters may be loaded, for rapid investigation of the effect of the number of coefficients. For the same specification (Here a low-pass filter), the chirp signal X[n] is poorly attenuated by a 3-order filter (Figure. 19-A) and efficiently attenuated using a 101-order filter (Figure. 19-B).



(A) (B)

Figure 19: Applying a recurrence equation on a simple signal.

Similar features are offered to Infinite-Impulse-Reponse (IIR) filters. The user selects the type of filter (low pass, high pass, band pass, notch). The cut-off frequency and filter parameter numbers may be changed. The synthesis method may be chosen between Bessel, Chebyshev and Butterworth (Figure. 20). The coefficients are then automatically computed.

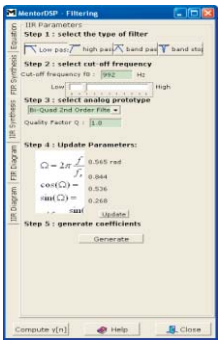


Figure 20: User's Interface for IIR Filter Synthesis.

Once familiarized by the key concepts of digital filtering, the student may want to enter its own recurrence equation. He can investigate several X[n] (pulse, step, user-defined), hear the input and output, observe the filter efficiency in the spectrogram or instant Fourier Transform. Note the general graphical view of the filter reported by figure. 21 which is automatically updated from the user's equation. The filter can be unstable depending on the value of coefficients.

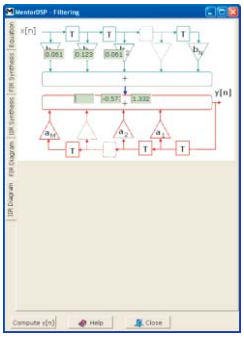


Figure 21: User's defined equation and visualization in FIR or IIR diagram.

The example of tone identification is illustrated in figure. 22. A Dual Tone Multi Frequency can be generated by MentorDSP using a dedicated screen. A band pass filter is created by using FIR/IIR filter synthesis tools. The student sees how the filter isolates the energy at the desired time

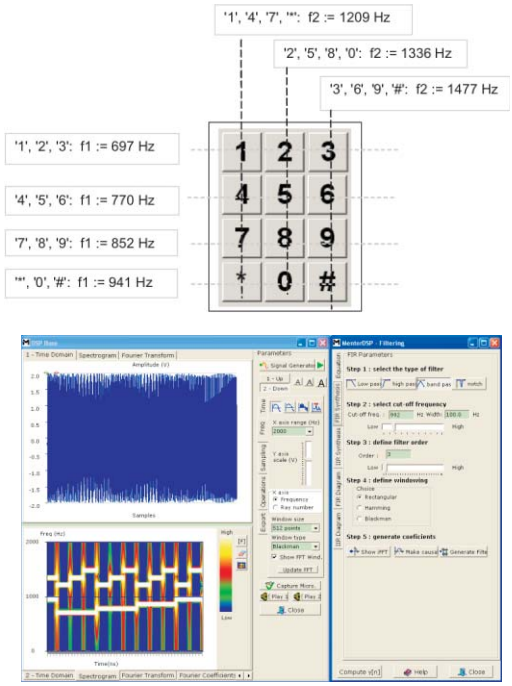


Figure 22: DTMF signal and spectrogram. A narrow band-pass filter may then be applied to isolate a given frequency component.

Modulation

An important section covered by MentorDSP concerns modulations. Most modulations are illustrated by coupling time and frequency domain representations and give access to the key parameters such as the modulation amplitude and signal frequencies.

Amplitude Modulation

The instant spectrum shows the components of the modulated signal according to the theory. In the case of amplitude modulation, the result is given by Equ. 8; and the corresponding time and frequency responses are shown in figure. 23.

And knowing that

$$S(t) = (1 + m.A)\sin(\omega_c t)$$

Where

w_c is the carrier pulse

m is the modulation coefficient

A is the modulation signal

When $A = \sin(\omega_m t)$

And knowing that

$$\sin A \sin B = \frac{\cos(A-B) - \cos(A+B)}{2}$$

$$S(t) = (1 + m.\sin(\omega_m t))\sin(\omega_c t)$$

$$S(t) = \sin(\omega_c t) + \frac{1}{2}m.\cos((\omega_m - \omega_c)t) - \frac{1}{2}m.\cos((\omega_m + \omega_c)t) \quad \text{Equ. 8}$$

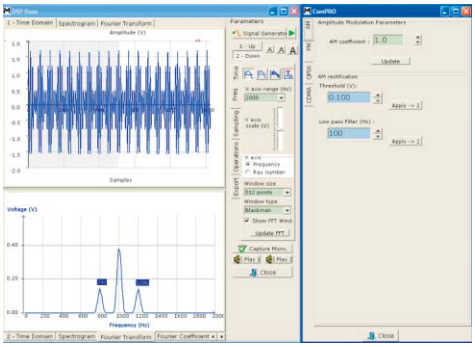


Figure 23: Amplitude modulation; time and frequency representations.

In the “COMpro Tool” window, the user may access to the modulation coefficient and see how sub-harmonic amplitude are altered by the AM coefficient. In AM modulation, the two harmonics at *f0+fc* and *f0-fc* (1200 Hz and 800 Hz) clearly appear on both sides of the carrier frequency 1000 Hz.

Frequency Modulation

Considering $A = \sin(\omega_m t)$

$$S(t) = \sin(\omega_c t + m.\sin(\omega_m t))$$

$$S(t) = \sum_{n=-\infty}^{+\infty} J_n(m)\sin((\omega_c + n\omega_m)t) \qquad \text{(Equ. 9)}$$

Where

Jn(m) are the Bessel coefficients usually provided by dedicated tables

The FM wave, represented by Equ. 9, contains an infinite number of sidebands at *(fc+fm), (fc+ 2fm), (fc+ 3fm),* etc, symmetrical at *fc.* As ***m*** increases, each sideband harmonic follows its own path of increasing and decreasing amplitude called a ***Bessel* function.** The student may either change the FM modulation parameters or enter its own equation for the Frequency Modulation, the corresponding time and frequency responses are shown in figure. 24.

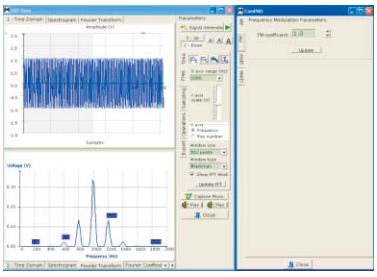


Figure 24: Frequency modulation; time and frequency representations.

The “Spectrogram” of the frequency modulation is proposed in figure. 25. At time t=0.1s, the signal ends and returns to a default value 0.0. This creates a discontinuity and generates a “Dirac-like” series of harmonics in the whole spectrum. Notice the harmonics at *f0+n.fc* and *f0-n.fc* (1200, 1400, 1600 Hz, etc. and 800, 600, 400 Hz, etc.

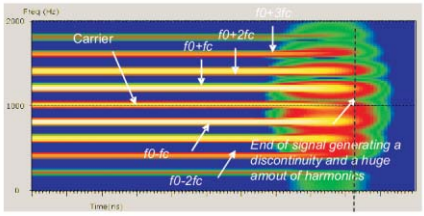


Figure 25: Spectrogram of the Frequency Modulation.

Quadrature-Phase-Shift Keying (QPSK)

The practical training for QPSK modulation, refer figure 26, is as follows: the student enters the data to be phase-modulated, sees the time-domain result, compute spectrogram. Spectrum can show discontinuities when phase changing. The view of constellation eases the link between data and phase of the signal. The step-by-step demodulation is the most innovative part of the QPSK illustration as the student can see how bits appear as DC levels. The user can demodulate signal modulated by other students, with possibility to add noise.

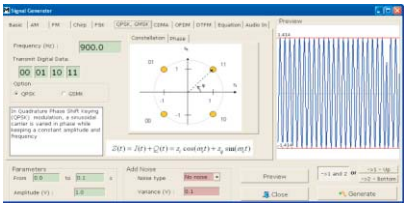


Figure 26: User's Interface for QPSK modulation.

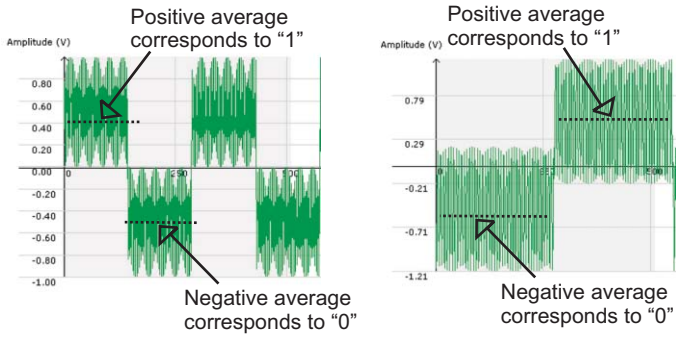


Figure 27: Demodulation of the QPSK signal by sine and cosine multiplication

In the COMpro tool menu, the QPSK signal is multiplied by the sinusoidal wave. The result of the multiplication of S(t) by a sine wave is the LSB of the data, that is “.0”, “.1”, “.0”, “.1”. The result of the multiplication of S(t) by a cosine wave is the MSB of the data, that is “0.”, “0.”, “1.”, “1.” (Figure. 27).

The Gaussian-Modulated-Shift-Keying ensures a smooth transition between two symbols to avoid time-domain discontinuities as follows (Figure. 28). Click “Modulation GMSK Modulation”. The time domain waveform and corresponding spectrogram appear. Phase changes almost generate no discontinuity except at the signal termination.

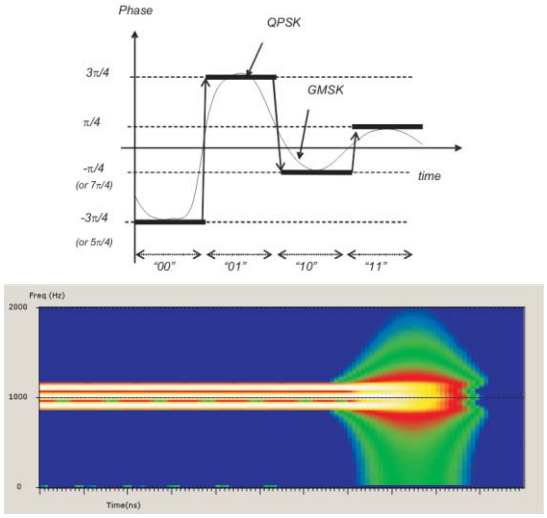


Figure 28 : GSMK vs. QPSK modulation.

Code-Division-Multiple Access

The CDMA modulation is one step further in modulation complexity due to the introduction of codes. Student enters the data to be emitted, choose the code, see time-domain result (Figure. 29) and compute spectrogram. When the CDMA signal is multiplied by the good code, data is restored (Figure. 30-A). When the CDMA signal is multiplied by the wrong code, data is similar to noise (Figure. 30-B).

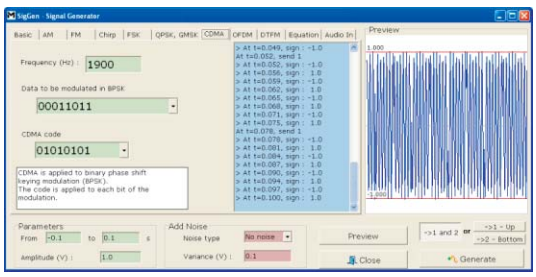
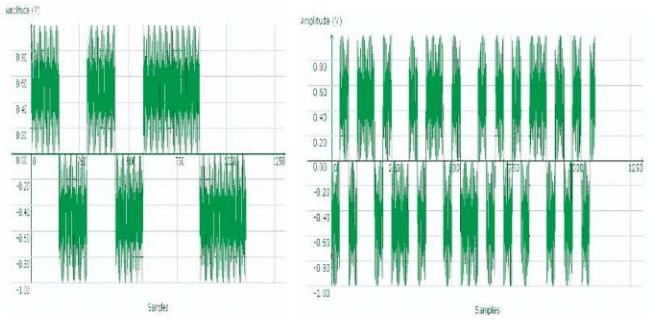


Figure 29: CDMA signal waveform with data and code information.



(A) (B)

Figure 30: CDMA demodulation with correct and incorrect codes.

OFDM Modulation

OFDM modulation is fairly complex as it uses Fourier and Invert Fourier operators. OFDM transmits data through a large number of narrowband carriers, closely spaced in the frequency domain. Invert-FFT avoid the use of a large number of modulators and filters at the transmitter. FFT does the same for the receiver. The OFDM modulation in MentorDSP is accessible through the menu reported in figure. 31. The OFDM symbol s(t) is the sum of N sinusoidal carriers with increased frequency, modulated by symbols *m* containing the data as written in Equ. 10. A 4-bit data modulated in OFDM has a spectral contents reported in figure. 32.

$$s[n] = \sum_{k=1}^{N} m[k] e^{j\frac{2\pi}{N}kn} \qquad \text{Equ. 10}$$

Where

N is the number of carriers (a power of 2 for FFT and iFFT implementation)

m[k] is the k^{th} symbol which contains the data

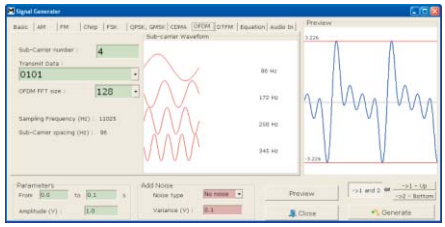


Figure 31: OFDM signal waveform with data and FFT information.

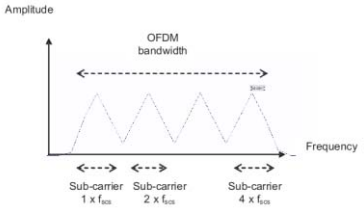


Figure 32: Spectral contents corresponding to a 4-bit data modulated in OFDM.

Image Processing

Image Properties

In the field of Image Processing, the primary objective of the MentorDSP software is to help the student to understand the link between pixel and RGB values, discover the HSV (Hue, Saturation, Value) coding, illustrate the correspondence in the palette. For example, the student loads a bitmap (Figure. 33), move the cursor in the image situated on the main image. The R,G,B values and corresponding H,S,V values are updated. The integer and hex values are also displayed. In the palette, the H and S points are plotted. In the right upper window, a zoom at the (5 x 5) nearest pixels is proposed.

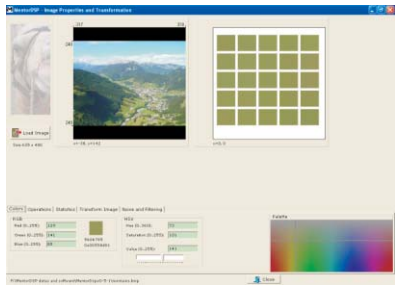
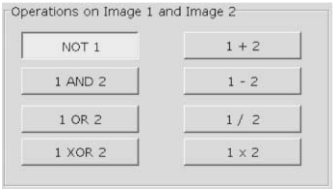


Figure 33: Introducing image coding (R,G,B - H,S,V).

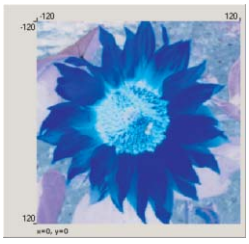
Histogram provides the frequency of one given property on a portion or the complete image. The region where most of the values are present is the tonal range. The user's interface reported in figure. 34 is designed to help the students to understand the decomposition of a picture into Red, Green, Blue planes, thanks to histograms. Histograms will help to understand the link between the picture and Hue, Saturation, Value maps, and also understands more advanced concepts such as brightness range, mid-tones, highlights, low-contrast and high-contrast images. A library of images is proposed to support these demonstrations.

Figure 34: Color decomposition using statistics.

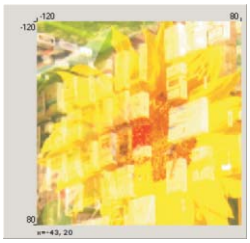
Basic Image Operations



(A)Menu



(B)Image inversion



(C)Adding images

Figure 35: Basic operations on images.

A set of basic image transforms is proposed to visualize the image inversion and Boolean operation applied to pixels. Addition of two images, substration or other arithmetic operators is also available. Some result examples are given in figure. 35.

Discrete Cosine Transform

The image compression is addressed through the Discrete Cosine Transform (DCT), and Invert DCT, for which the formulation is recalled in Equ. 11 and Equ. 12. The conversion from raw pixel to frequency coefficients and image reconstruction is supported by a user's interface which illustrates the coefficients quantization. The main DCT parameters are user accessible. The DCT is applied to very simple images to understand the link between image pixels and frequency. The student may also visualize the impact of quantization resolution and image compression factor on the resulting image quality.

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\ \cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad \text{(Equ. 11)}$$

$$f(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad \text{(Equ. 12)}$$

f(x, y) = pixel value at (x,y) coordinates
F (u,v) = DCT value at (u,v) frequencies
N = number of points (8 points in MentorDSP, as for JPEG images)

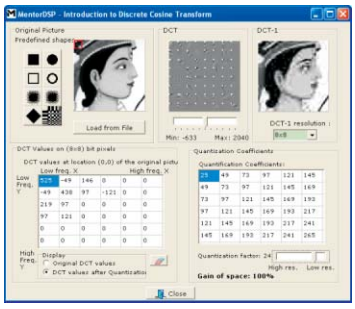


Figure 36: Tutorial on DCT and invert DCT.

The tutorial on DCT is as follows: the student selects a predefined picture, select an area in the picture, visualize its frequency contents. He may also modify the DCT array to investigate the impact on the reconstructed image by applying DCT^{-1}. Modifying the "Quantization factor" has a direct impact on coefficient resolution, and consequently on the picture quality, as seen in figure. 36.

Picture Transform

The general equation for picture transform are as follows

$$X=\frac{ax+by+c}{gx+hy+1} \quad \text{Equ. 13}$$

$$Y=\frac{dx+ey+f}{gx+hy+1} \quad \text{Equ. 14}$$

The student investigates first basic 2D transforms such as the 45°-rotation shown in figure. 37. He/She can investigate the effect of other transforms using presets (Stretch in X or Y). More complex transforms are also proposed, where the initial and final coordinates are defined by the user. The matrix coefficients are updated according to the user's coordinates, and the final image is computed.

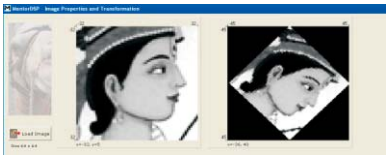


Figure 37: Image Transform.

Image Correlation

In MentorDSP you can found 2D correlation between images

$$r=\frac{(\sum ai\,bi)^2}{\sum ai^2\sum bi^2} \quad \text{Equ. 15}$$

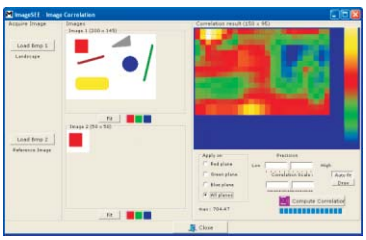


Figure 38:The peak of correlation corresponds to the best (X,Y) offset to match Image 1 and Image2.

Image Noise and Filtering

In MentorDSP, white noise or Gaussian noise (Equ. 16) may be added to the image as defined in Equ. 17.

- **Gaussian noise** :

$$f=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(t-m)^2}{2\sigma^2}} \quad \text{(Equ. 16)}$$

- **Additive noise** :

$$I2[x,y]=I1[x,y]+\nu(x,y) \quad \text{(Equ. 17)}$$

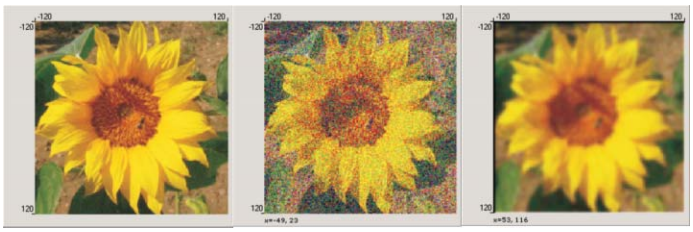
The image filtering implemented in MentorDSP corresponds to the application for all pixels of the general equation 18.

$$I2[x,y]=\frac{I1[x,y]}{\partial x\partial y}$$

$$I2[x,y]=w(2,2).I1[x,y]+w(1,1).I1[x-1,y-1]+w(3,3).I1[x+1,y+1]+w(1,3).I1[x+1,y-1]+w(3,1).I1[x-1,y+1] \quad \text{(Equ. 18)}$$

	X-1	X	X+1
Y-1	W(1,1)		W(1,3)
Y		W(2,2)	
Y+1	W(3,1)		W(3,3)

The user may add noise to an image and change its amplitude until a significant degradation is observed. In the case of figure. 39-a, the noise level is 1/10 of the maximum amplitude. When the noise is comparable to the color amplitude, the degradation is significant (Figure. 39-b). By applying a filter as described by figure. 40, the image quality is significantly improved (Figure. 39-c).



(a) low noise *(b) high noise* *(c) Noise filtering*

Figure 39: Additive Noise to an image.

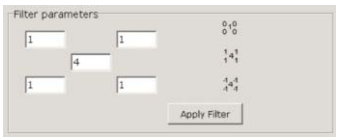


Figure 40: Filter example.

Conclusion

This article has described the educational software MentorDSP designed for efficient education in the field of signal and image processing. The software features real-time sound and image processing capabilities to enhance its impact on students. Particular effort has been dedicated on illustrating signal and image coding. The most important algorithms (Correlation, Fourier Transform, Filter Design) have been illustrated in a user-friendly and attractive way, both for the illustration of theoretical courses by teachers and to conduct practical training. Important effort has also been dedicated to modulations, including GMSK, CDMA and OFDM.

The MentorDSP software is proposed with around 40 lab practical covering all fundamental aspects of digital signal processing, with step-by-step procedures, tutorials and most important observations. The software is also interfaced with a hardware platform, simple and efficient, to illustrate the implementation of the algorithms on an embedded digital signal processor. We hope that the practical approach proposed in this article will attract students to the fascinating world of signal processing, enabling them to contribute later to innovative future designs.