DICOM, MRI and Bioinstrumentation using Matlab and Simulink

Dr. Mohammad Rafiq Muqri, DeVry University, Pomona
Prof. Shih Ek Chng, DeVry University, Pomona

Professor College of Engineering and Information Science
**DICOM, MRI and Bioinstrumentation using MATLAB and Simulink**

BMET students interested in biomedical signal processing, digital imaging and communications in medicine (DICOM), picture archiving, communication system (PACS), and bioinstrumentation are deprived of the opportunity to take background courses such as 2D-signal processing, computer communications, radiography, and sensors and instrumentation. Compare to traditional electrical engineering students, the lack of hands-on lab experience becomes more apparent when students are working on capstone senior projects.

One strategy we used to solve this issue was to include a mixed capstone project group comprising of computer, electronic, and biomedical engineering students. This strategy worked for few groups but the success rate was less than thirty five percent due to the students’ lack of preparation and disadvantage of knowledge compared to traditional electrical engineering students.

To rectify this situation we proposed and developed this teaching module which incorporates well selected signal processing, biomedical imaging and instrumentation topics which make extensive use of MATLAB, Simulink, and LabVIEW tools. This teaching module includes a detailed description of associated core lab exercises, student responses and recommendations. This low cost program consists of a series of theory modules coupled with a hands-on laboratory component using readily available test equipment and graphical capabilities of MATLAB, Simulink and LabVIEW software. As such this paper concerns problem solved and lessons learned while developing computer-assisted instruction strategy to improve the current state of learning in the classroom. This will introduce the students to new topics not covered in traditional courses. The goal of this paper is to pass on information useful to anyone contemplating related work, where similar occurrences are likely. This paper will explain how this learning and teaching module was instrumental in progressive learning of these students, improving their performance and successful culmination of capstone senior projects.

**Setting up a Biomedical Instrumentation Laboratory**

Teaching students how to use specialized instruments and equipment that are currently used in the medical field do not serve the purpose for engineering students trying to apply the knowledge of engineering into medicine. Most equipment is designed for taking measurements only which limit students from practicing engineering. They are mainly used for research purposes because they are too complex and very costly. The development of a Biomedical Instrumentation Lab for the undergraduate program requires thoughtful planning especially for most teaching institutions offering BS-BMET program; it has limited budget for acquiring specialized instrument and equipment to be used for conducting experiments mainly for educational purposes. Developing a single platform that is flexible enough to perform most undergraduate physiological experiment in lab become possible with the Biomedical Workbench developed by National Instrument. This platform allows students to develop their own test system with the latest measurement technologies while providing an environment for developing an application in the capstone project.
The instrumentation setup which can be used for most human physiology laboratory experiment is shown in figure 1 below.

**Figure 1 Biomedical Instrumentation Setup**

Typical experiments using this basic instrumentation setup includes Cardiovascular Physiology test, Neurophysiology test, and Pulmonary Ventilation test. The following are a compilation of experiments for each category.

A. Cardiovascular Physiology Experiments
1. Electrocardiogram and Heart Sounds
2. Electrocardiogram and Peripheral Circulation
3. Exercise, the Electrocardiogram and Peripheral Circulation
4. Blood Pressure, Peripheral Circulation and Body Position
5. Blood Pressure, Peripheral Circulation and Imposed Conditions

B. Neuro Physiology Experiments
1. Electroencephalography (EEG)
2. Electromyography (EMG)
3. Evoked potentials (EP)
4. Reflexes and Reaction Times

C. Pulmonary Ventilation
1. Breathing Parameters at Rest and After Exercise
2. Breathing and Gravity
3. Factors that Affect Breathing Patterns
The instrumentation setup can be divided into 4 major parts. I.) The Silver (Ag)-Silver Chloride (AgCl) electrodes, II.) high performance bio-potential amplifier with build-in hardware filter from iWorx, III.) National Instrument Elvis II+, and IV.) Laptop/PC running Biomedical Workbench, LabVIEW, and MATLAB.

I.) Ag-AgCl electrodes — Each 4-mm diameter reusable button scalp electrodes is made of silver covered with silver chloride coating. This electrode has impedance typically less than 100 Ω which is most suitable for detecting EEG signal voltage typically less than 50 uV. To maximize skin contact and reduce artifacts due to movement of the electrode during acquisition of EEG signal, electrode gel is applied in between the electrode and the scalp. The electrodes are connected to the Isolated Module which is connected to the Bio-Potential Amplifier.

II.) Bio-Potential Amplifier with build-in hardware filter — The pre-amplifier is a low noise instrumentation amplifier with very high CMRR 85dB up to 200 Hz and input impedance greater than 10GΩ. To provide enough signal level for voltage acquisition by Elvis II plus, the amplifier is equipped with 8 selectable voltage gain of up to x5k. Hardware analog filter is integrated into the amplifier to allow preprocessing of raw signal by removing unwanted signal pickup from body movement or noise of 60 Hz riding on the baseline of the EEG signal. EEG signal frequency typically ranging from 0.5Hz the low end of Delta wave to 30 Hz the high end of Beta wave. The High pass and Low pass filter cutoff frequency is set to 0.3Hz and 50Hz respectively on the Bio-Potential amplifier to meet this preliminary bandwidth requirement.

III.) Elvis II plus — Elvis II+ is a Modular Engineering Educational Laboratory Hardware platform by National Instrument for use in data acquisition. It is connected to PC with a high speed USB cable. The output of the Bio-Potential amplifier is connected to the analog input (AI) which is configured to measure voltage. Since EEG signal contains multiple channels of data corresponding to different positions on the brain, up to 8 analog input channels can be connected to Elvis II plus at any given time.

IV.) Biomedical Workbench — The diagram in figure 2 shows the latest Biomedical Tool Kit from National Instrument. This application software is used for real-time acquisition or generation of simulated bio signals.
Appendix A illustrates the procedure for Instrument Control Settings on this application and the experimental results from Evoked potentials in Neuro physiology experiment using the Biomedical instrumentation setup described above. This experiment detects the electrical signals of the brain in response to light stimulation of the eyes.

**Advantages of using the Biomedical Workbench**

The illustration above shows how simple the Biomedical Workbench software together with the Elvis-II plus hardware platform is to acquire real time biomedical signal. One advantage of using the software is to be able to create multiple virtual channels simultaneously with individual filtering parameters from one acquired real time signal. This feature is especially important for analyzing EEG signal because the acquired biosignal for several given bandwidth of interest can be generated in one acquisition. The ECG Feature Extractor to extract ECG features for heart rate variability analysis is also proven to be useful as well as the Noninvasive Blood Pressure Analyzer. To create a meaningful lab, student should take the acquired data and process through MATLAB or LabVIEW program to try to match with the results from the Feature Extractor and Analyzer from Biomedical Workbench and draw conclusions about the accuracy of their own algorithm. The Feature Extractor and Analyzer is by no means a model answer but can be used as a reference or secondary resource to justify the results from analysis.

Another useful feature is the file format converter. Institutions currently using different hardware equipment that generate different data file format can be easily converted to .tdms file format for use in Biomedical Workbench. Figure 3 lists the file formats that can be converted and saved.
For radiographic imaging experiment, the 3D image re-constructor helps student visualize 3D model from a set of 2D image slices. This software accepts external image files from real applications in different file formats including DICOM, BMP, JPEG, and PNG. Instructor can provide sample DICOM files downloaded from the website to the students. By using this feature, students can view the detailed internal structure in three dimensions, which provides detailed information about the region of interest (figure 4).
Implementation of a real time DSP system

In our university, we have introduced the Tower System Microcontroller for two Eight-Week sessions to the students before the capstone project. Students are required to know how to setup and connect the Tower board (Figure 5) and ADC to DAC data acquisition board (APPENDIX figure B1) for the filter experiments. Students design the digital filter for a given design specification with the MATLAB SPTOOLS graphical filter design editor using a Parks-McClellan iterative algorithm for digital filter coefficients determination. The filter will then be implemented using the CodeWarrior, an integrated development environment (IDE), for the creation of program that runs on the Tower System Microcontroller (APPENDIX figure B2). Signal conditioning using operation amplifier for anti-aliasing and anti-imaging filter is also included in the curriculum to enhance their knowledge in analog filter design. Performance of the Tower embedded system board can be tested by using the NI Elvis Instrument Launcher (APPENDIX C). The Function Generator has frequency sweep capabilities which allow Bode Analyzer to create a frequency response automatically of the real time DSP system. In the capstone project, this real time DSP system can be connected to the Elvis II plus data acquisition platform for real time biosignal processing application.

![Figure 5 Tower System Microcontroller Setup](image_url)

Reading image file using MATLAB

In Capstone project, student who is interested in 2D signal processing application is to use MATLAB Image Acquisition toolbox to acquire images and video from hardware. Students use different commands in MATLAB to experiment and practice them on their favorite sample images.

The following image formats are supported by MATLAB:

- BMP
- HDF
- JPEG
- PCX
- TIFF
- XWB
Most sample images found on the Internet are JPEG-image that is widely used compression standards for images. An image named xxxx.jpg is stored in the JPEG format. APPENDIX D show how an image can be loaded into MATLAB. A digital image is composed of pixels which can be thought of as small dots on the screen. Each pixel is represented by a binary number which describe the color of the pixel.

In general, there are four basic representation of a pixel:

1. **Binary.** Each pixel is either black or white. Since there are only two possible values for each pixel, we need only 1 bit per pixel, as such they are efficient in terms of storage. Images for which a binary representation may be suitable include text (handwriting or printed), architectural plans, or fingerprints.
2. **Grayscale.** Each pixel is a shade of gray, normally from 0(black) to 255 (white). This range implies that each pixel can be represented by 8-bits (1-byte). Other gray scale ranges are also used, but generally they are a power of 2. Such images arise in medicine (x-rays), and images of printed works etc.
3. **True color.** Color digital images require three values to be recorded for each pixel: one red component, one green component, and one blue component. These components are combined with different weightings to produce a range of colors. If each of these components has a range 0-255, this gives a total of $256^3 = 16,777,216$ different colors in the image. Since total number of bits required for each pixel is 24, such images are also called 24-bit color images.
4. **Indexed.** Most color images have only a small subset of the more than 16 million possible colors. For convenience of storage and file handling, the image has an associated color map, or color palette, which is simply a list of all colors used in that image. Each pixel has a value that does not give its color (as for a RGB image, but an index to the color in the map).

Images from one image type can be converted to another image type by using MATLAB functions as shown in the table below.

<table>
<thead>
<tr>
<th>Function</th>
<th>Use</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>ind2gray</td>
<td>Indexed to grayscale</td>
<td>x = ind2gray(y, map);</td>
</tr>
<tr>
<td>gray2ind</td>
<td>Grayscale to indexed</td>
<td>[x, map]=gray2ind(y);</td>
</tr>
<tr>
<td>rgb2gray</td>
<td>RGB to grayscale</td>
<td>x = rgb2gray(y);</td>
</tr>
<tr>
<td>gray2rgb</td>
<td>Grayscale to RGB</td>
<td>x = gray2rgb(y);</td>
</tr>
<tr>
<td>rgb2ind</td>
<td>RGB to indexed</td>
<td>[x, map]=rgb2ind;</td>
</tr>
<tr>
<td>ind2rgb</td>
<td>Indexed to RGB</td>
<td>x=ind2rgb(y, map)</td>
</tr>
</tbody>
</table>

**Table 1  MATLAB image converter functions**

APPENDIX D shows sample of MATLAB commands for reading and converting 2D image. Student can further process the converted 2D image for some specific applications.
Digital Signal Processing using Java

Last but not the least, the four Java programs or code snippets which is shown in APPENDIX E were used by engineering technology students before even they have been exposed to control theory or DSP courses. This was accomplished by a class lecture in OBP which was followed by examples of simple substitutions in the Java program code; however most motivated students will want to do more than that. The Faculty here at our university attempted to indulge students in such activities. Figure E1 is an example of an instructor lead program which the student edited, compiled and displayed the output. The purpose of this program is to show how general Java workhorse discrete Fourier Transform and other control theory methods can be introduced at an earliest stage to engineering technology students with the tools and concepts they will further reinforce in future DSP courses.

The TestDFT application class shown in APPENDIX figure E2, uses class Fourier and invokes its methods. Highly efficient algorithms for computing the DFT were first developed in the 1960s. Collectively known as Fast Fourier Transforms (FFTs), they all rely upon the fact that the standard DFT involves redundant calculation. Strictly speaking, there is no such thing as ‘the FFT’ 3. Rather, there is a collection of algorithms with different features, advantages, and limitations. An algorithm which is suitable for programming in a high level-language on a general purpose computer may not be the best for special purpose DSP hardware. What the different algorithms have in common is their general approach – the decomposition of the DFT into a number of successively shorter, and simpler, DFTs.

There are various ways of explaining FFT decomposition. We can show that a DFT can be expressed in terms of shorter, simpler, DFT’s by dividing the signal x[n] into subsequences. The method which is widely used in DSP literature is also referred to as conventional decomposition. Then there is also an alternative approach known as index-mapping. It should be clear in our mind that conventional decomposition and index mapping are just two ways of looking at the same problem and there is no essential difference between them.

Supposed we have a signal with N sample values, where N is an integer power of 2. We first separate x[n] into two subsequences, each with N/2 samples. The first subsequence consists of even number points in x[n], and the second consists of odd number points- Writing n = 2k, when n is even, and n = 2k + 1 when n is odd. We can thus express the original N-point DSP in terms of two N/2 point DFTs. Now we can take the decomposition further, by breaking each N/2- point subsequence down into two shorter, N/4-point subsequences. The process can continue until, in the limit, we are left with a series of 2-point subsequences, each of which requires a very simple 2-point DFT. A complete decomposition of this type gives rise to one of the commonly used radix-2, decimation in time, FFT algorithms.

Now the students are ready to implement an FFT as a Java method. It is called the fastFFT( ) Method as shown in figure E3 and is also defined in the Fourier class.
In the next step, the students apply this method to compute the FFT on a 2Hz cosine wave. They were instructed to take 64 data samples over a 2-second sample period. The program first computes the FFT to obtain the frequency spectrum for a 2Hz cosine wave. Then the program was used by students to perform an inverse Fourier transform that reconstructs the 2Hz cosine wave from its frequency spectrum. Next, we implement the FFT as a Java method. It is called the fastFFT( ). The next thing to do is to test the fastFFT( ) by applying it to the composite cosine signal that were processed earlier. The amplitude time history for a signal containing three different frequency components is generated and sent to fastFFT() method. The TestFFT class source code is shown in figure E4.

These are just a few of the representative Biomedical Instrumentation, DSP using Java programming, Image processing using MATLAB laboratory modules to which students get exposed.

Feedback and Assessment

Continuous examining the evolving needs of our students and employers for career-oriented higher education programs as basis for development of additional programs is our university mission and purpose. Agencies accrediting our programs are also increasingly focused on student outcomes and achievement. Student outcomes are the skills and abilities students are expected to demonstrate at graduation. One of the student outcomes in our program is the ability to conduct standard tests and measurements; to conduct, analyze, and interpret experiments; and to apply experimental results to improve biomedical systems and processes.

We have offered this DICOM, MRI, Bioinstrumentation course as a special topics course which is one way new courses are piloted locally at our university campus. This course is offered to target student outcome as described above. This course is instrumental in the progressive learning of the students by relating and applying fundamentals of circuit analysis, analog and digital electronics, micro-computing, power electronics, electro-mechanics, and various energy and material concepts to bioengineering systems. Although special topics are not evaluated in the same manner as standard session-long courses, feedback directly from students indicates that initial offering was well-received.

A compilation of feedback was from four junior students enrolled in the course for credit, and three sophomore students attended the lectures and discussions but did not receive the credit.

The feedback from the student indicates that:

1. Students gained an appreciation of the extensive background required by our engineering and computer information sciences technology program. MRI, DICOM and Bioinstrumentation puts many concepts in calculus, linear algebra and java programming to practical use in a context that has been described as “cool” by the students.

2. Students gained an appreciation of the natural sciences courses required by our engineering program, particularly physics. Often students reported that the physical concepts related to a particular topic (for example electrostatics, resonance, Amperes Law, and Faraday’s Law of electromagnetic induction) made much more sense after implementing the concept in the context of the Magnetic resonance Imaging and bioinstrumentation applications.
3. Students gained an appreciation for the difficulties involved in developing and debugging complex software system. The DFT and FFT coding was for many students, the first java programming experience with a non-trivial code base that has to be designed and written from scratch and leverages the power of java data structures.

4. Students spent a significant amount of time on the signal processing with MATLAB, Simulink, applications of FFT and Java programming assignments (presumably relative to their other course work) but the results were satisfying. We did not receive any complaints about the level of effort required by, nor the time spent on the programming assignments.

Student’s performance in the initial course offering and in the course of capstone projects was exceptionally high. This result was due to a biased sampling; the four juniors taking the special topic course initiated the effort, and the sophomores that attended regularly were invited by the instructor. We hope to see better understanding of basic principles and excellent performance in the future versions of the course, but a wider range of students will undoubtedly test the validity of mixed performance in course of capstone senior projects.

Conclusion

With proper guidance, monitoring, and diligent care, the biomedical engineering technology students can be exposed earlier to medical instrumentation, Java data structures and the basics of MATLAB. Recent assessment results have shown that with the layout of training modules before taking the capstone project, the fear of BMET students taking the capstone projects is not only eliminated, but has also built up their confidence and improved the quality and creativity of their projects. With proper conditioning and judicious course selection, students will become more motivated and will help reinforce the best practices in implementing capstone senior projects.
Bibliography

9. Zhijun Gu, NI Biomedical Start up Kit 3.0, July 2010, https://decibel.ni.com/content/docs/DOC-12646
APPENDIX A: Instrument Control Settings and Results for Evoked potentials (EP) experiment

There are two Instrument Control Settings on this application. When the selector switch is on the Logger Settings mode, step 1 of 3 for Setting involve selecting the physical channel (AI), Input Range, and Terminal Configuration. The diagram in figure A1 shows one Physical Channel Setting. Up to 8 analog input channels (ai0-ai7) can be acquired simultaneously on Elvis II plus. Input Range is set to -5/+5V. This allows acquisition of amplified raw EEG signals up to +/- 5V after the Bio-Potential amplifier. The input range can be changed to improve signal resolution. Differential Input is selected to amplify the signals between the +/- electrodes.

Figure A1 Physical Channel Setting

Step 2 involves setting the filtering parameters for the 4 main frequency ranges of the EEG signal. Figure A2 to A5 shows the filter parameters used for Delta, Theta, Alpha, and Beta Wave. A tabulation of the four filters parameters is shown in figure A6. Butterworth filters are used for all 4 virtual channels with Passband ripple of 0.1dB and Stopband attenuation of 60 dB.
### Table A6  Tabulation of the four filters parameters

<table>
<thead>
<tr>
<th>Virtual Channel Name</th>
<th>Passband edge frequency (Hz)</th>
<th>Stopband edge frequency (Hz)</th>
<th>Bandpass Filter Designed Order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower</td>
<td>Upper</td>
<td>Lower</td>
</tr>
<tr>
<td>Delta Wave</td>
<td>0.5</td>
<td>3</td>
<td>0.2</td>
</tr>
<tr>
<td>Theta Wave</td>
<td>4</td>
<td>7</td>
<td>3.5</td>
</tr>
<tr>
<td>Alpha Wave</td>
<td>8</td>
<td>13</td>
<td>7.5</td>
</tr>
<tr>
<td>Beta Wave</td>
<td>14</td>
<td>30</td>
<td>13.5</td>
</tr>
</tbody>
</table>

Step 3 in figure A7 shows all 4 virtual channels and the physical channels are enabled for 30 seconds duration of logging time. Logging will stop automatically after the 30 second duration has past.

![Logger setting](image)

### Figure A7  Logger setting

Significant increase of Alpha Wave amplitude was detected as shown in figure A8 when eyes were closed and relaxed as compared to figure A9 when eyes were open and concentrated starring at the computer monitor.
Figure A8  Eyes wide opened and stared at the computer monitor

Figure A9  Eyes closed and relaxed
APPENDIX B: TOWER_ADCDAC_AXIOM board

This board features one ADC and one DAC. The user can access the channels of these chips from the green connection strips located on the right and left of the board. Note that the ADC inputs are on the left side (TB1) and the DAC outputs are on the right side, as shown in the picture below. Also note that there are two DAC output strips. The output of the DAC is bipolar on the upper right connector (TB4) with 4 outputs, but unipolar on the lower right connector with 8 outputs (TB2).

![Figure B1 ADC/DAC AXIOM Board](image)

![Figure B2 Implementation of FIR filter coefficient in CodeWarrior](image)
**Figure C1** NI ELVIS Instruments

- **Digital Multimeter**: DC voltage, AC voltage, Current (DC and AC), Resistance, Capacitance, Inductance, Diode test, Audible continuity
- **Oscilloscope**: Two channels, scaling and position adjustment knobs, modifiable timebase, autotrigger, digital or analog hardware triggering, cursors for accurate screen measurements
- **Function Generator**: Sine, output sine, square, or triangle waveforms, amplitude selection, and frequency settings, DC offset setting, frequency sweep capabilities, amplitude and frequency modulation
- **Variable Power Supply**: Positive (0 and +12 V) or negative (−12 and 0 V)
- **Bode Analyzer**: Set the frequency range of the instrument, choose between linear and logarithmic display scales
- **Dynamic Signal Analyzer**: Continuous or single scan measurements, apply various window and filtering options
- **Arbitrary Waveform Generator**: Uses Waveform Editor (included with NI ELVISmx), load waveforms, generate two waveforms simultaneously, run continuously or once
- **Digital Reader**: Reads digital data from eight consecutive lines at a time (0...7, 8...15, 16...23), continuous single reading
- **Digital Writer**: Manually create a digital pattern or select predefined patterns (ramp, toggle, walking 1s), control eight consecutive lines, continuous or single write, TTL compatible
- **Impedance Analyzer**: Capable of measuring the resistance and reactance for passive two-wire elements at a given frequency
- **2-wire Current-Voltage Analyzer**: Conduct diode parametric testing, view current-voltage curves; full flexibility in setting parameters such as voltage and current ranges
- **3-wire Current-Voltage Analyzer**: Conduct transistor parametric testing, view current-voltage curves; base current settings for measurements of NPN and PNP transistors
APPENDIX D  Sample MATLAB commands for reading 2D image

Images are read into MATLAB environment using function imread, whose syntax is

.>> imread (‘file-name’)  

1. Here file-name is a string containing the complete name of the image file (including any applicable extension). For example, the command line

.>> I = imread (‘x-ray-white2.jpg’)  

reads the JPEG image x-raywhite2 into image array I. Then type

.>> imshow(I)  
.>> J = im2double(I);  
.>> whos

2. Depict and explain the output for this step .

3. Now try using this command line, which is really three commands on one line.

.>> figure, imshow (I), pixval on

The three commands we use here are :

figure, which creates a figure on the screen. A figure is essentially a window in which a graphics object can be placed. Objects may include images or various types of graphs.

imshow (I), which displays the matrix I as an image.

pixval on, which turns on the pixel values in our figure. This is a display of the gray values of the pixels in the image. They may appear at the bottom of the figure in the form

c, r = p

where c is the column value of the given pixel; r its row value, and p its gray value.
5. Given below are some examples of MATLAB commands and their typical outputs.

>> I = imread('D:\MyDocuments\My Pictures\Image\cell1.jpg');

>> whos

Name  Size          Bytes  Class  Attributes
I     512x512       262144  uint8

>> imshow(I)

>> J = im2double(I);

>> whos

Name  Size          Bytes  Class  Attributes
I     512x512       262144  uint8
J     512x512       2097152 double

For this step 6 use your favorite digital color image.

Again given below are some more examples of MATLAB commands and their typical outputs.

>> I = imread('D:\MyDocuments\My Pictures\Image\dv-portal1.jpg');

>> whos

Name  Size          Bytes  Class  Attributes
I     227x730x3      497130  uint8
J     512x512       2097152 double

>> K = rgb2gray(I);

>> whos

Name  Size          Bytes  Class  Attributes
I     227x730x3      497130  uint8
J     512x512       2097152 double
K     227x730       165710  uint8

>> imshow(K)

>> I = imfinfo('D:\MyDocuments\My Pictures\Image\cell1.jpg')
I =
Filename: [1x43 char]
FileModDate: '13-Apr-2013  21:12:43'
FileSize: 98531
Format: 'jpg'
FormatVersion: 

Width: 512
Height: 512
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: 
NumberOfSamples: 1
CodingMethod: 'Huffman'
CodingProcess: 'Sequential'
Comment: 

>> H = figure, imshow('D:\MyDocuments\My Pictures\ImageP\cell1.jpg'),pixval on
>> H = figure, imshow('D:\MyDocuments\My Pictures\ImageP\cell1.jpg'),impixelinfo
>> H = figure, imshow('D:\MyDocuments\My Pictures\ImageP\cell1.jpg'),imdistline
public class Fourier {
    public static double[] discreteFT(double[] fdata, int N, boolean fwd) {
        double[] X = new double[2*N];
        double omega;
        int k, ki, kr, n;

        if (fwd) {
            omega = 2.0*Math.PI/N;
        } else {
            omega = -2.0*Math.PI/N;
        }
        for(k=0; k<N; k++) {
            kr = 2*k;
            ki = 2*k + 1;
            X[kr] = 0.0;
            X[ki] = 0.0;
            for(n=0; n<N; ++n) {
                X[kr] += fdata[2*n]*Math.cos(omega*n*k) +
                        fdata[2*n+1]*Math.sin(omega*n*k);
                X[ki] += -fdata[2*n]*Math.sin(omega*n*k) +
                         fdata[2*n+1]*Math.cos(omega*n*k);
            }
        }
        if (fwd) {
            for(k=0; k<N; ++k) {
                X[2*k] /= N;
                X[2*k + 1] /= N;
            }
        }
        return X;
    }
}

Figure E1  DFT program
```java
public class TestDFT {
    public static void main(String args[]) {
        int N = 64;
        double T = 2.0;
        double tn, fk;
        double fdata[] = new double[2*N];
        for(int i=0; i<N; ++i) {
            fdata[2*i] = Math.cos(4.0*Math.PI*i*T/N);
            fdata[2*i+1] = 0.0;
        }
        double X[] = Fourier.discreteFT(fdata, N, true);
        for (int k=0; k<N; ++k) {
            fk = k/T;
            System.out.println("f["+k+"] = "+fk+"Xr["+k+"] = "+X[2*k]+ " Xi["+k+"] = "+X[2*k + 1]);
        }
        for (int i=0; i<N; ++i) {
            fdata[2*i] = 0.0;
            fdata[2*i+1] = 0.0;
            if (i == 4 || i == N-4) {
                fdata[2*i] = 0.5;
            }
        }
        double x[] = Fourier.discreteFT(fdata, N, false);
        System.out.println();
        for (int n=0; n<N; ++n) {
            tn = n*T/N;
            System.out.println("t["+n+"] = "+tn+"xr["+n+"] = "+x[2*n]+ " xi["+n+"] = "+x[2*n + 1]);
        }
    }
}

Figure E2  TestDFT Program
```
public class Fourier {

public static void fastFFT(double[] fdata, int N, boolean fwd) {

double omega, tempr, tempi, fscale;
double xtemp, cosine, sine, xr, xi;
int i, j, k, n, m, M;

j = 0;
for(i = 0; i < N - 1; i++) {
    if (i < j) {
        tempr = fdata[2*i];
        tempi = fdata[2*i + 1];
        fdata[2*i] = fdata[2*j];
        fdata[2*i + 1] = fdata[2*j + 1];
        fdata[2*j] = tempr;
        fdata[2*j + 1] = tempi;
    }
    k = N/2;
    while (k <= j) {
        j -= k;
        k >>= 1;
    }
    j += k;
}

if (fwd)
    fscale = 1.0;
else
    fscale = -1.0;

M = 2;
while (M < 2*N) {
    omega = fscale*2.0*Math.PI/M;
    sin = Math.sin(omega);
    cos = Math.cos(omega) - 1.0;
    xr = 1.0;
    xi = 0.0;
    for (m = 0; m < M - 1; m += 2) {
        for (i = m; i < 2*N; i += M*2) {
            j = i + m;
            tempr = xr*fdata[j] - xi*fdata[j+1];
            tempi = xr*fdata[j+1] + xi*fdata[j];
            fdata[j] = fdata[i] - tempr;
            fdata[j+1] = fdata[i+1] - tempi;
            fdata[i] += tempr;
            fdata[i+1] += tempi;
        }
    }
    xtemp = xr;
}
public class TestFFT {
    public static void main(String args[]) {
        int N = 64; double T = 1.0; double tn, fk;
        double fdata[] = new double[2*N];

        for(int i=0; i<N; ++i) {
            fdata[2*i+1] = 0.0;
        }
        Fourier.fastFT(fdata, N, true);
        System.out.println();
        for(int k=0; k<N; ++k) {
            fk = k/T;
            System.out.println("f[" + k + "] = " + fk + " Xr[" + k + "] = " + fdata[2*k] + " Xi[" + k + "] = " + fdata[2*k+1]) ;
        }
    }
}

Figure E3 FFT program

Figure E4 Test FFT program