IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⋇ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11133

APPLICATIONS OF MATLAB IN DIGITAL SIGNAL PROCESSING

Prof. Vishal V. Mehtre[1], Shambhavi Sinha[2]

Assistant Professor, Department of Electrical Engineering, Bharati Vidyapeeth (Deemed to be university) College of Engineering, Pune[1].

Student, Department of Electrical Engineering, Bharati Vidyapeeth (Deemed to be university) College of Engineering, Pune[2].

Abstract: Matlab's rich and powerful functions have made it a fundamental teaching tool in the course of linear algebra, signals and systems, control theory, digital signals processing, image processing. Matlab has provided many main signal and system processing functions such as convolution, Fourier transform, Laplace transform, z-transform, etc, which simplify the calculation process greatly.

This paper describes the applications of Matlab in signals and systems and digital signal processing DSP. Matlab provides various methods to analyze the signals and systems, including both continuous and discrete situations. Some methods can simplify the complicated calculation: some finish the same functions in accordance with the mathematical processing: and some can save operation time via efficient algorithms. Matlab can be used at different levels to solve problems quickly and efficiently, to understand the signal processing procedures deeply and to develop new algorithms.

Keywords: Convolution, DSP, Fourier, Matlab, Optimize, Signal, System, Simulation

I. INTRODUCTION

Although Matlab has provided many main signal and system processing functions such as convolution, Fourier transform, Laplace transform, z-transform, etc, which simplify the calculation process greatly, in the mean time some related mathematical processing and physical meaning are covered up. The goal is to introduce simulation. To solve the problem Matlab is used at three different levels:

II. USING MATLAB AT THREE DIFFERENT LEVELS

A. First Level - Using Matlab's Powerful Functions Directly

In first level, Matlab's powerful signals processing and systems analyzing functions are introduced directly and the inner complicated calculation are omitted temporarily. The goal is to get the final results and to verify the theory. Not only does this efficient function simplify the program but also will be used in future application and should be grasped firmly.

B. Second Level—Using Matlab to Realize the Mathematical Process

Matlab is used to realize step by step mathematical formulas and the usual expression of the signals and systems processing methods. This level is a useful supplement of the first level to help comprehend the theory deeply and in detail, though the programs in accordance with the formulas are usually a little complicated and less efficient. Moreover, this shortcoming provides a new chance to learners to optimize the programs, which leads to the usage of Matlab in the third level.

C. Third Level—Developing new Optimizing algorithms

The second level gives a deep understanding of the mathematical process, based on which some new efficient algorithms can be found to simplify and speed up the calculation. An additional unexpected effect is realized. Two important signal processing functions are given as examples to illustrate the three application levels.

III. FOURIER TRANSFORM BY MATLAB

DOI: 10.17148/IJARCCE.2022.11133

Fourier transform is one of the important transforms of the signals and builds the foundation for modern signal processing technology. Simulating Fourier transform by Matlab is important both to the present theoretical study and to the future application.

A. First Level

At first level, we use functions “fourier” and “ifourier” provided by Matlab to realize Fourier and inverse Fourier transform conveniently. Input the following program in a m-file and run it.

```
syms t; %Define a signal
F = fourier (heaviside (t)) %Calculate the Fourier transform of step function.
ezplot (abs (F1),[-3,3]); %Draw magnitude-frequency curve.
```

The following result will be obtained in the command window.

```
F =Pi*dirac (w)-i/w
```

That is to say, F = $\pi\ \delta(\omega) - j/\ \omega$, which is just the Fourier transform of the step function. And the figure shown in Fig.1 will be drawn at the same time. In the figure the impulse function $\pi\ \delta\ (\omega)$ is omitted.

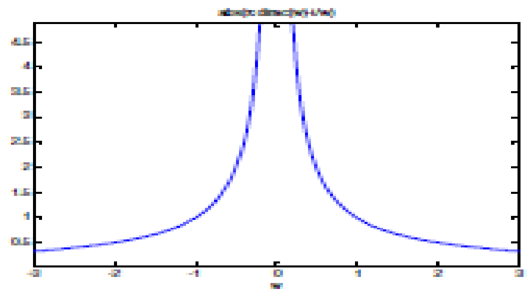


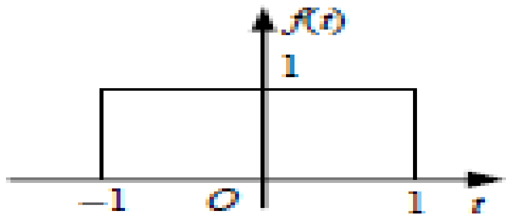
Figure 1. Magnitude-frequency characteristics of step function’s Fourier transform

B. Second Level

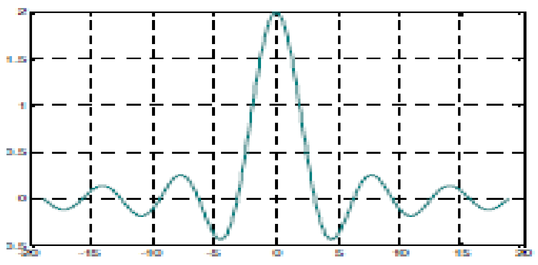
According to Fourier transform’s definition equation (1), program can be written to calculate the integral and F (ω) will also be obtained.

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t}\, dt \tag{1}$$

The following program can calculate the Fourier transform of the square wave shown in Fig.2 (a) and the result, a sample function, is shown in Fig.2 (b).



(a) Square wave



IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 1, January 2022

DOI: 10.17148/IJARCCE.2022.11133

(b) The Fourier transform of square wave

Figure 2. Square wave and its Fourier transform

```
W = linspace (-6*pi, 6*pi, 512); %Assign the frequency points to be calculated.
N = length (w); %Get the total number of the frequency point.
X = zeros (1, N); % Set a vector X of length N to save the calculation results.
for k = 1: N
X (k) =quad (@ (t) qpulse (t, w (k)),-1, 1);
end
% for each frequency point w(k), calculate the definite integral
```

$$F(\mathrm{j}\omega) = \int_{-\infty}^{\infty} f(t)\, \mathrm{e}^{-\mathrm{j}\omega t} dt$$

‘qpulse’ is the square wave function defined in the following qpulse.m file.

```
plot (w, real(X)); grid on; %Draw Fourier transform.
```

The ‘qpulse’ function quoted is defined as the following.

```
function y=qpulse (t, w);
y = (stepfun (t,-1) – stepfun (t, 1)).*exp (-j*w*t);
```

C. Third Level

Some numerical methods can also be developed to realize (1).

Set [t1, t2] as the main value range of t and T = t2 −□t1. If N points are sampled in the range and the sample interval is Δt = T/N

Set [ω1, ω2] as the main value range of ω and Ω = ω2 −□ω1. If K points are sampled in the range and the sample interval is Δω = Ω/K

Then a numerical calculation equation of (1) can be obtained in (2), which can be programmed as the next program

$$F(\omega_1 + k\Delta\omega) = \frac{T}{N}\sum_{n=0}^{N-1} f(t_1 + n\Delta t)\mathrm{e}^{-\mathrm{j}(\omega_1 + k\Delta\omega)(t_1 + n\Delta t)} \qquad (2)$$

```
T=2; %Define the time domain sample interval length.
N=200; %Define the number of the sample points.
t = linspace (-T/2, T/2-T/N, N)'; %Define the sample points.
f =0*t; % Initialize the time function.
F (t>-1&t<1)=1; %Assign value to time function.
OMG=16*pi; % Define the frequency domain sample interval length.

K=100; %Define the number of the sample points.
w = linspace (-OMG/2, OMG/2-OMG/K, K)';
%Define the sample points.
F=0*w; %Initialize the spectrum function.
for k=1:K %Calculate the spectrum for each frequency point in loop.
for n=1:N
F (k) =F (k) +T/N*f (n)*exp (-j*w (k)*t (n));
end %Finish the sum in (2) in loop.
end
k=1: K;
plot (k,real(F));
```

The result is the same with Fig.2 (b).

But the two-fold loops generate a large amount of calculation and a long running time, which can be overcome by some optimization. The summing in (2) can also be realized by the vector calculation in (3).

$$F(\omega_1 + k\Delta\omega) = \frac{T}{N}\left[\mathrm{e}^{-\mathrm{j}(\omega_1 + k\Delta\omega)t_1} \ \mathrm{e}^{-\mathrm{j}(\omega_1 + k\Delta\omega)(t_1+\Delta t)} \cdots \mathrm{e}^{-\mathrm{j}(\omega_1 + k\Delta\omega)(t_2-\Delta t)}\right]$$

(3)

$$\cdot\left[f(t_1)\ f(t_1+\Delta t) \cdots f(t_2-\Delta t)\right]^{\mathrm{T}}$$

Substitute the two-fold loops in the above program with the following program, and the same result will be obtained.

```
for k=1:K
F(k)=T/N*exp(-j*w(k)*t).'*f;
End
```

And the computation can be simplified further. When a matrix

$$\mathbf{U} = \exp\left[-j\begin{pmatrix}\omega_1\\ \omega_1+\Delta\omega\\ \vdots\\ \omega_2-\Delta\omega\end{pmatrix}\otimes\left(t_1\ t_1+\Delta t\cdots t_2-\Delta t\right)\right]$$

is defined, F = T/N Uf, where

F = [F (ω_1) F ($\omega_1 + \Delta\omega$) ………. F (ω_2 - $\Delta\omega$)]^T
And f = [f (t1) f ($t_1 + \Delta t$)……. f (t_2 - Δt)]^T The loop can be substituted by the following vector and matrix calculation completely.

```
U = exp (-j*kron (w, t.’)); % Calculate the tensor product of two vectors.
F=T/N*U*f; % Fourier transform is finished.
```

The Fourier transform has been simulated at three levels, whose results can be verified mutually. Each level helps to view the same problem at a different angle. The inverse Fourier transform can be dealt with similarly.

IV. CONVOLUTION BY MATLAB

Convolution integral and sum are important calculation, because by convoluting an input signal e(t) with a system’s impulse response h(t) the system’s response r(t) will be obtained, just as in (4). And the same conclusion applies to the discrete systems.

$$r(t) = e(t) * h(t) \tag{4}$$

A. First Level

Matlab provides function “conv” to convolve two vectors, which can be used to obtain a convolution sum directly and a convolution integral after a little modification. A convolution integral is defined by

$$f(t) = f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t-\tau)d\tau \tag{5}$$

It can be discretized into

$$f(nT) \approx T\cdot\sum_m f_1(mT) f_2(nT - mT) \tag{6}$$

where T is the sample interval. Compared with the function ‘conv’ in Matlab:

$$w(n) = \sum_m u(m)v(n-m) \tag{7}$$

we should multiply T with the results of the function ‘conv’ to get the correct answer. So a new convolution function ‘cconv1’ applied to continuous convolution integral is programmed.

```
function [w, tw] = cconv1 (u, tu, v, tv)
% u and v are two vectors to be convoluted, and tu and tv are their sampling times respectively.
% w is the convolution result and tw is w’s sampling time.
T = tu (2)-tu (1); %Get sample interval.
```

DOI: 10.17148/IJARCCE.2022.11133

```
W = T*conv (u, v); %Calculate (6) via function ‘conv’.
tw = tu(1)+tv(1)+T*[0:length(u)+length(v)-2]';
%Distribute sample time for w.
```

Then the function ‘cconv1’ can be called in any convolution integral calculation, taking the following program as an example. It finishes the convolution of two square wave with different width and a trapezoidal wave is obtained.

```
t= [-2:0.01:2]; %Generate sampling time.
e= (t>-1 & t<1); %Define a square wave with width of 2.
h= (t>-0.5 & t<0.5);
% Define a square wave with width of 1.
[r1, t1] = cconv1 (e, t, h, t);
% Call ‘cconv1’ to finish convolution integral
subplot(3,1,1);plot(t,e);axis([-2 2 -0.2 1.2]);title('e(t)');
subplot(3,1,2);plot(t,h);axis([-2 2 -0.2 1.2]);title('h(t)');
subplot(3,1,3);plot(t1,r1);axis([-4 4 -0.2 1.2]);title('r(t)');
```

The result is shown in Fig.3.

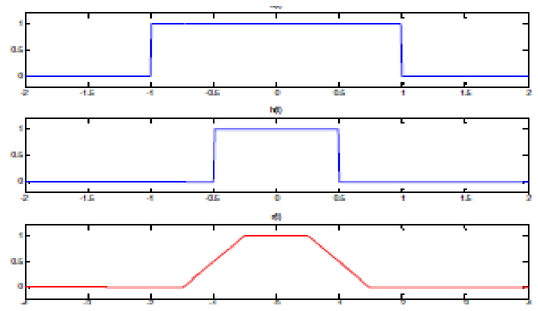


Figure 3. Square waves and their convolution

B. Second Level

The convolution sum described by (7) can also be finished by loop program without via the function ‘conv1’, which asks the programmer understand the summing process clearly, and many possible situations must be considered first. For example, the starting point and the end point of the summing in (7) must be justified attentively to ensure the program be run smoothly. Referring to the graphical interpretation of convolution sum, after careful study we found that the starting point must be the larger one between the starting points of u(m) and v(n-m) and the end point must be the smaller one between the end points of them, as shown in the following program, function ‘cconv2’, a substitute of‘cconv1’

```
function [w,tw] = cconv2(u,tu,v,tv)
T=tu (2)-tu (1);
L1 = length (u); %Get the length of vector u.
L2 = length (v); %Get the length of vector v.
for n=1:L1+L2-1 % define the length of w.
w (n) =0; %Initialize w.
for m=max (1,n+1-L2):min(n,L1);
% Determine the starting and end points of the summing.
W (n) = w(n)+u(m)*v(n-m+1); %Calculate the sum.
end
w (n)=T*w(n); %Modify the convolution sum to convolution integral.
end
tw = tu(1)+tv(1)+T*[0:L1+L2-2]';
```

If we call ‘cconv2’ instead of ‘cconv1’ in main program, the same result as Fig.3 will be obtained. During the process, the formation of the convolution sum has been analyzed thoroughly, which is a good supplement of the first level.

C. Third Level

Of course the two-fold loops in the above program can be replaced by some vector calculations and the program will be optimized. The detail will not be described here.

DOI: 10.17148/IJARCCE.2022.11133

V. CONCLUSION

In this paper application of Matlab in signals and systems and digital signal processing DSP is presented. Application of Matlab at three levels is introduced and two typical examples are given. The first level utilizes the Matlab already-provided functions directly and the calculations are simplified. The second level finishes the same functions in accordance with the mathematical processing and helps to understand the signal processing procedures deeply and in detail. The third level develops some new algorithms to optimize the programs.

Analyzing the same problem at three levels helps to understand the theory deeply and thoroughly and some related mathematical processing and physical meaning are revealed. Matlab's rich and powerful functions are used and the simulation is enhanced.

REFERENCES

[1] Digital Signal Processing using Matlab by Vinay K. Ingle & John G. Proakis, 2007 Thomson Learning
[2] Fundamentals of Digital Signal Processing using Matlab by Robert J. Schilling & Sandra L. Harris,2007, Thomson Learning
[3] Digital Signal processing A practical approach by Emmanuel C. Ifeachor & Barrie W. Jervis, 2004, Pearson Education
[4] Oppenheim, Alan V. & Willsky, Alan S., "Signals and Systems", Prentice Hall of India, 2nd Edition
[5] Oppenheim A.V., Schafer, Ronald W. & Buck, John R.,"Discrete Time Signal processing", Pearson Education ,2nd Edition
[6] Proakis, J.G. & Manolakis, D.G.," Digital Signal Processing: Principles Algorithms and Applications", Prentice Hall of India.