IMK-14 – Istraživanje i razvoj u teškoj mašinogradnji 22(2016)3, SR81-89
UDC 621 ISSN 0354-6829

Primena programskog paketa MATLAB u digitalnoj obradi signala brzom Furijeovom transformacijom

Snežana Gavrilović[1*], Nebojša Denić[2], Jelena Erić-Obućina[1], Goran Miodragović[1]
[1]Visoka Tehnička Mašinska Škola Strukovnih Studija, Trstenik, Srbija
[2]Alfa Univerzitet, Beograd, Srbija

Da bi teorijske osnove iz digitalne obrade signala postale razumljive i primenjive, potrebno ih je vizuelizovati. Programski paket Matlab je odličan partner u toj misiji. Zahvaljujući fleksibilnom okruženju, širokom spektru ugrađenih funkcija, mogućnošću razvoja algoritma i programiranja Matlab se nametnuo kao nezamenjiv alat u gotovo svim oblastima inženjerske prakse. Za digitalnu obradu signala jedan od najznačajnijih je algoritam brze Furijeove transformacije, čija funkcija je ugrađena u Matlab. U ovom radu biće vizuelizovani neki od osnovnih primera digitalne obrade signala brzom Furijeovom transformacijom, upotrebom softverskog paketa Matlab.
.

Ključne reči: Matlab, digitalni signal, algoritam, brza Furijeova transformacija

1. UVOD

U skladu sa razvojem informaciono komunikacionih tehnologija, digitalna obrada signala je naglo napredovala u poslednjih nekoliko decenija. Napredak je impresivan, kako u teoriji, tako i u domenu primene. Razvoj digitalne obrade signala povezan je sa prodorima koji su ostvareni u mikroelektronici, kao i u računarskim i softverskim tehnologijama. Zahvaljujući tom ukupnom napretku, razvoj savremenih sistema se zasniva uglavnom na digitalnoj tehnologiji. Digitalna obrada signala predstavlja osnovnu tehnologiju za mnoge grane tehnike kao što su: telekomunikacioni sistemi, mobilne komunikacije, radar, multimedija, obrada govora, audiotehnika, sonarna tehnika, obrada slike, medicinska elektronika, seizmologija, napredne tehnike merenja i upravljanja, robotika, kartografija i mnoge druge.

Posebna lepota digitalne obrade signala je u bliskoj povezanosti teorije i prakse. Softverske tehnologije koje su nam danas na raspolaganju pružaju mogućnost da se već u toku upoznavanja sa osnovnim algoritmima digitalne obrade signala stečeno znanje kreativno primenjuje u rešavanju praktičnih problema Jedan od moćnih softverskih alata za obradu i vizuelizaciju digitalnog signala je programski paket Matlab proizvođača MathWorks .[1]

2. O MATLAB-u

Milioni inženjera i naučnika širom sveta koriste Matlab za analizu i projektovanje sistema , razvijanje algoritama i kreiranje modela i aplikacija. Matlab se može koristiti za veliki niz aplikacija, uključujući obradu signala i komunikacije, obradu slike i videa, sistem kontrole, testiranja i merenja, računanje finansija, itd. Uključuje matematičke funkcije za linearnu algebru, statistiku, Fourierovu analizu, filtriranje, optimizaciju, numeričku integraciju i rešavanje diferencijalnih jednačina, koje podržavaju zajedničke inženjerske i naučne operacije. Temeljne matematičke funkcije koriste procesorsku optimizaciju biblioteka kako bi pružali brzo izvođenje vektorskih i matričnih proračuna. [2]

Matlab nudi alate za analizu i vizuelizaciju podataka što omogućava uvid u dobijene podatke, a za to je potrebno kraće vreme nego kad se koriste proračunske tablice ili tradicionalni programski jezici. Moguće je dokumentovati i podeliti svoje rezultate kao grafikone funkcija ili u obliku izveštaja.

Matlab omogućava pristup podacima sa različitih aplikacija, baza podataka i spoljašnjih uređaja. Podaci se mogu čitati u popularnim alatima poput Microsoft Excel-a, moguće je čitanje tekstualnih i binarnih datoteka, datoteka slike, zvuka i videa.

S Matlab programskim jezikom, moguće je napisati program ili razviti algoritam brže nego kod tradicionalnih programskih jezika zato što ne treba izvoditi administrativne procese niskog nivoa kao što su deklaracija promenljivih, specifikacija vrste podataka i alokacija memorije. Matlab nudi osobine tradicionalnih programskih jezika, kao što su rukovanja greškama, tok kontrole i objektno orijentisano programiranje. Mogu se koristiti osnovni tipovi podataka ili napredne strukture podataka, kao i prilagođeni tipovi podataka. Moguća je integracija s drugim programskim jezicima i aplikacijama.Moguće je deliti individualni algoritam i aplikaciju s drugim Matlab korisnicima ili ih razvijati za one koji ne koriste Matlab. Moguće je dizajnirati i uređivati prilagođeni grafički interfejs.

3. SIGNAL I DIGITALNA OBRADA SIGNALA

Signal je nosilac određene informacije. Sadržaj informacije koju signal nosi zapisan je u promenama funkcije ili nekog od njenih parametara, pa se signal prema tome može definisati kao funkcija vremena koja nosi informaciju o nekoj veličini od interesa. Postoje dve vrste signala: kontinualni i diskretni.

Pod kontinualnim signalom podrazumevamo one kod kojih je funkcija koja predstavlja signal neprekidna funkcija vremena. Ako pri tome amplituda signala može

*Kontakt adresa autora: Visoka tehnička mašinska škola strukovnih studija, Radoja Krstića 19, 37240 Trstenik,
e-mail:gavrilovicsnezana@yahoo.com

uzimati proizvoljnu vrednost iz dozvoljenog opsega, signal se naziva analogni signal.

Diskretni signali su definisani samo za diskretne vrednosti nezavisne promenjljive vremena. Diskretni signali nastaju diskretizacijom analognih signala uz odredjene uslove. Amplituda diskretnog signala moze biti kontinualna ili diskretna. Ukoliko je izvršena kvantizacija amplituda diskretnog signala dobijen je digitalni signal.[3]

Digitalna obrada signala (DOS) predstavlja promenu i/ili analizu informacija koje su sadržane u diskretnoj sekvenci brojeva. Signali dolaze iz realnog sveta, što ukazuje na potrebu da se obrada obavlja u realnom vremenu i da se signali konvertuju u digitalni oblik. Signali su diskretni što znači da su informacije između diskretnih odabiraka izgubljene pa ih je potrebno rekonstruisati. Postoje različiti algoritmi i programski paketi za obradu digitalnog signala.

U programskom paketu Matlab, Signal Processing Toolbox obezbeđuje funkcije i aplikacije za generisanje, merenje, transformaciju, projektovanje filtera i vizuelizaciju signala. Toolbox poseduje algoritme za uzorkovanje, izjadnačavanje i sinhronizaciju signala, projektovanje i analizu filtera, spektralnu analizu, propusni opseg i distorziju. Toolbox takođe sadrži parametre i linearne prediktive za modeliranje. Koristeći Signal Processing Toolbox možemo realizovati analizu i poređenje signala u vremenskom, frekventnom i vremenski-frekventnom domenu. Jedan od nezamenjivih algoritama u digitalnoj obradi signala je algoritam brze Furijeove transformacije (FFT), čija je funkcija ugrađena u programskom paketu Matlab.[4]

4. BRZA FURIJEOVA TRANSFORMACIJA

Brza Furijeova transformacija (FFT) je algoritam za "brzo" izračunavanje vrednosti diskretne Furijeove transformacije (DFT). Ubrzanje u odnosu na uobičajen postupak izračunavanja diskretne Furijeove transformacije postiže se izbegavanjem ponovnog izračunavanja izraza koji se međusobno negiraju. Algoritam "podeli pa vladaj" se pripisuje Džejmsu V. Kuliju (James W. Cooley) i Džonu V. Tukiju (John W. Tukey) koji su ga objavili 1965. godine.[5]

Tri glavna koraka principa podeli pa vladaj (engl. the Divide and Conquer Paradigm) su:

- podeliti sekvencu na dve ili više podsekvenci,
- rešiti svaku sekvencu rekurzivno pomoću istog algoritma i postaviti granične uslove da bi završili sa rekurzijom kada dužina podsekvence postane dovoljno mala i
- sastaviti rešenje originalne sekvence kombinujući rešenja podsekvence.

Intuitivno je jasno da će algoritam podeli i reši imati najbolje rezultate ako je dužina sekvence *N*, eksponent broja 2, jer podela sekvenci u sukcesivno manje grupe može se nastaviti dok ne ostane samo jedan par u grupi. Takodje, postoji mnogo slučajeva kada *N* nije eksponent broja 2 i tada je potrebno modifikovati algoritam. Polazi se od pretpostavke da je $N = 2^n$.

FFT algoritam baze 2 je rekurzivni algoritam koji se sastoji od deljenja dobijene sekvence (i svake podsekvence) u dve podsekvence upola manje dužine. Postoje dva uobičajno korišćena FFT algoritma koja se razlikuju u načinu kako su podsekvence definisane. To su brza Furijeova transformacija sa decimacijom po vremenu (DIT FFT) i brza Furijeova transformacija sa decimacijom po frekvenciji (DIF FFT).[6]

4.1.Decimacija po vremenu

Jednodimenzionalni niz *x(n)* razložimo na dva, tako da se u prvom nizu $f_1(n)$ nalaze samo parni uzorci niza *x(n)*, a u drugom $f_2(n)$ samo neparni uzorci.Tada DFT u N tačaka ima oblik:[7]

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} =$$

$$k = 0,1,\dots,N-1$$

$$= \sum_{n\ je\ paran} x(n) W_N^{kn} + \sum_{n\ je\ neparan} x(n) W_N^{kn} =$$

$$= \sum_{m=0}^{\frac{N}{2}-1} x(2m) W_N^{2km} + \sum_{m=0}^{\frac{N}{2}-1} x(2m+1) W_N^{(2m+1)k} =$$

$$= \sum_{m=0}^{\frac{N}{2}-1} f_1(n) W_{\frac{N}{2}}^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_2(n) W_{\frac{N}{2}}^{km} =$$

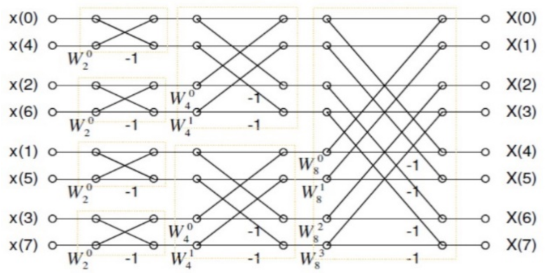
$$= F_1(k) + W_N^k F_2(k), \quad k = 0,1,\dots,N/2-1$$

$$F_1(k) = \sum_{m=0}^{\frac{N}{2}-1} f_1(n) W_{\frac{N}{2}}^{km} \quad \text{i}$$

$$F_2(k) = \sum_{m=0}^{\frac{N}{2}-1} f_2(n) W_{\frac{N}{2}}^{km}$$

su vrednosti DFT u N/2 tačaka i periodični su sa periodom N/2. $W_N^k = e^{\frac{jk2\pi}{N}}$ je rotacioni faktor.

Korišćenjem ove procedure, N-člana DFT je svedena na dve N/2-člane DFT. Sledeći istu proceduru decimacije, svaka N/2-člana DFT se može svesti na dve N/4-člane DFT, i tako sve dok se izračunavanje ne svede na 2-člane DFT. U nastavku je dat detaljan prikaz izračunavanja FFT sa decimacijom u vremenu za slučaj kada je N = 8.



Slika 1. DIT FFT

4.2.Decimacija po frekvenciji

Jednodimenzionalni niz *x(n)* razložimo na dva, tako da se u prvom nizu nalaze prvih n=N/2 elemenata niza *x(n)* , a u drugom druga polovina niza *x(n)* .Tada DFT u N tačaka ima oblik:[7]

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} =$$

$$k = 0,1,\dots,N-1$$

$$= \sum_{prvih\ n} x(n) W_N^{kn} + \sum_{drugih\ n} x(n) W_N^{kn} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{kn} + \sum_{n=N/2}^{N-1} x(n)W_N^{kn} =$$

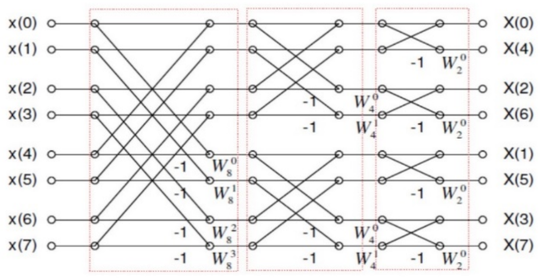
$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)W_N^{(n+\frac{N}{2})k} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{kn} + W_N^{\frac{N}{2}k}\sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)W_N^{nk} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{kn} + (-1)^k\sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)W_N^{nk},$$

$$k = 0,1,\dots,N/2-1$$

Procedura izvršavanja algoritma je slična kao i kod decimacije po vremenu. U nastavku je dat detaljan prikaz izračunavanja FFT sa decimacijom u vremenu za slučaj kada je N = 8.



Slika 2. DIF FFT

5. OPIS FUNKCIJE FFT U MATLAB-u

Y=fft(X) izračunava diskretnu Furijeovu transformaciju signala X, koristeći algoritam brze Furijeove transformacije.[8]

• Ako je X vektor, onda fft(X) vraća Furijeovu transformaciju kao vektor.

• Ako je X matrica, onda fft(X) tretira kolone kao vektore i vraća Furijeovu transformaciju svake kolone posebno.

• Ako je X višedimenzionalni niz, tada fft(X) tretira vrednosti duž cele prve dimenzije niza, koji su različiti od 1, kao vektor, i vraća Furijeovu transformaciju svakog vektora.

Primer: *Audio signal*

Koristeći Furijeovu transformaciju izdvaja se korisni signal iz buke.

Navode se parametri signala sa frekvencijom uzorkovanja od 1kHz i trajanja od 1 sekunde.

```
Fs = 1000;         % Sampling frequency
T = 1/Fs;          % Sampling period
L = 1000;          % Length of signal
t = (0:L-1)*T;     % Time vector
```

Formiraju se signali koji sadrže sinusoide frekvencije 50 Hz i amplitude 0.7, i sinusoide od 120 Hz i amplitude 1.

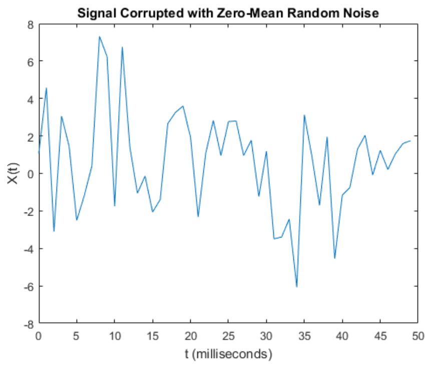
```
S = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
```

U signalu je prisutan beli šum.

```
X = S + 2*randn(size(t));
```

Iscrtava se signal u vremenskom domenu. Uočava se da je teško identifikovati frekvencijske komponente signala X(t).

```
plot(1000*t(1:50),X(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('t (milliseconds)')
ylabel('X(t)')
```



Slika 3: Noisy Signal-decimacija po vremenu

Pristupa se izračunavanju Furijeove transformacije signala sa decimacijom po frekvenciji.

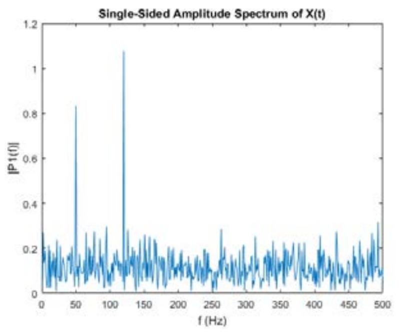
```
Y = fft(X);
```

Vrednosti dvostranog spektra P2 se prebacuju u pozitivne, i na taj način se formira jednostrani spektar P1 i vrši se dalje izračunavanje na celoj dužini signala.

```
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
```

Nakon ovoga vrši se iscrtavanje signala P1 u frekvancijskom domenu. Amplitude nisu tačno 0.7 i 1, kao što se očekivalo, zbog prisustva belog šuma. U proseku, duži signali će proizvesti bolju aproksimaciju frekvencije.

```
f = Fs*(0:(L/2))/L;
plot(f,P1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```



Slika 4: Noisy Signal-decimacija po frekvenciji X(t)

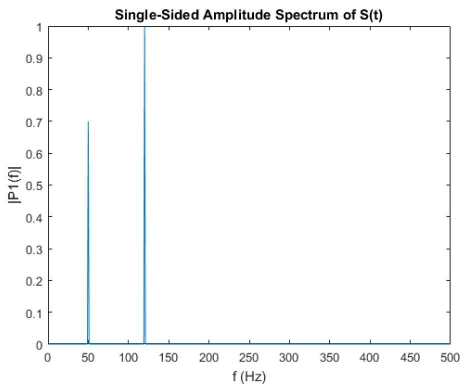
Zatim se primenjuje Furijeova transformacija na originalni signal bez prisustva šuma, sa precizno definisanim amplitudama od 0.7 i 1.0.

```
Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
```

Iscrtava se signal u frekventnom domenu.

```
plot(f,P1)
title('Single-Sided Amplitude Spectrum of S(t)')
```

```
xlabel('f (Hz)')
ylabel('|P1(f)|')
```



Slika 5: Noisy Signal-decimacija po frekvenciji S(t)

Y = fft(X,n) vraća n vrednosti DFT. Ako se drugačije ne navede, Y ima isti broj vrednosti kao X.[8]

• Ako je X vektor čija je dužina manja od n, tada se X dopuni nulama do dužine n.

• Ako je X vektor čija je dužina veća od n, onda se X skrati na dužinu n.

• Ako je X matrica tada se svaka kolona tretira kao poseban vektor

• Ako je X višedimenzionalni niz, tada se vrednosti duž cele prve dimenzije niza, koji su različiti od 1, tretiraju kao vektori.

Primer: *Gausov impuls*

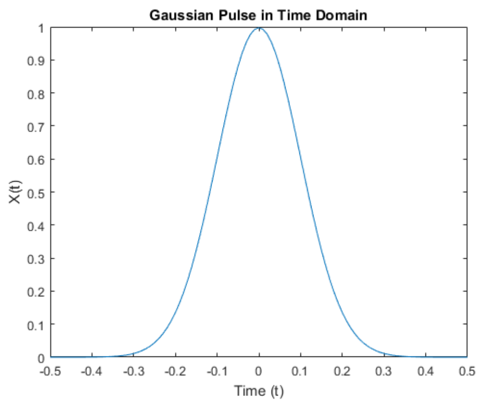
Izvršeno je pretvaranje Gausovog impulsa u vremenski i frekvencijski domen.

Definisani su parametri Gausovog impulsa, X.

```
Fs = 100;          % Sampling frequency
t = -0.5:1/Fs:0.5;  % Time vector
L = length(t);      % Signal length
X = 1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));
```

Iscrtava se signal u vremenskom domenu.

```
plot(t,X)
title('Gaussian Pulse in Time Domain')
xlabel('Time (t)')
ylabel('X(t)')
```



Slika 6: Gaussian Pulse- decimacija po vremenu

Da bi se upotrebio algoritam FFT za prikazivanje signala u frekventnom domenu, prvo se mora definisati nova dužina signala n=2^nextpow2(L) . Ovo će biti osnova signala X koga treba dopuniti nulama za poboljšanje performansi FFT.

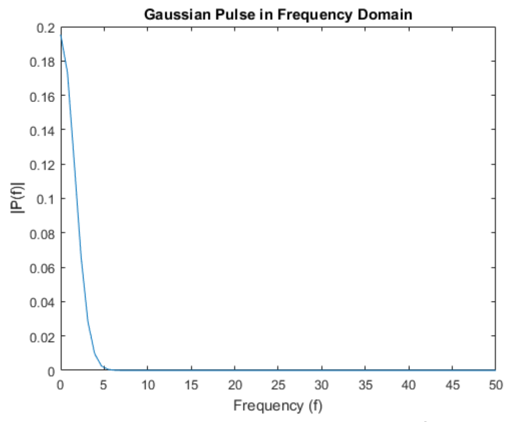
```
n = 2^nextpow2(L);
```

Vrši se pretvaranje Gausovog impulsa u frekventni domen.

```
Y = fft(X,n);
```

Definiše se frekventni domen i iscrtavaju jedinstvene frekvencije.

```
f = Fs*(0:(n/2))/n;
P = abs(Y/n);
plot(f,P(1:n/2+1))
title('Gaussian Pulse in Frequency Domain')
xlabel('Frequency (f)')
ylabel('|P(f)|')
```



Slika 7: Gaussian Pulse- decimacija po frekvenciji

Y = fft(X,n,dim) vraća Furijeovu transformaciju dužine dim. Na primer, ako je X matrica, tada fft(X,n,2) vraća n vrednosti Furijeove transformacije svakog reda.[8]

Primer: *Kosinusoidni talasi*

Prikazani su kosinusni talasi u vremenskom i frekvencijskom domenu.

Navedeni su parametri signala sa frekvencijom uzorkovanja od 1kHz i trajanja od 1 sekunde.

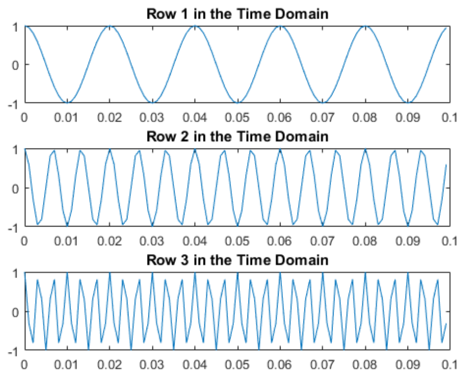
```
Fs = 1000;               % Sampling frequency
T = 1/Fs;                % Sampling period
L = 1000;                % Length of signal
t = (0:L-1)*T;           % Time vector
```

Kreirana je matrica u kojoj svaki red predstavlja kosinusni talas sa različitim frekvencijama. Postoje tri reda sa po 1000 vrednosti. U prvom redu su talasi frekvencije 50 Hz, u drugom 150 Hz, a u trećem talasi frekvencije 300 Hz.

```
x1 = cos(2*pi*50*t);        % First row wave
x2 = cos(2*pi*150*t);       % Second row wave
x3 = cos(2*pi*300*t);       % Third row wave
X = [x1; x2; x3];
```

Prikazano je prvih 100 vrednosti iz svakog reda i izvršeni poređenje njihove frekvencije u vremenskom domenu.

```
for i = 1:3
   subplot(3,1,i)
   plot(t(1:100),X(i,1:100))
   title(['Row ',num2str(i),' in the Time Domain'])
end
```



Slika 8: Cosine Waves- decimacija po vremenu

Za postizanje boljih performansi algoritma FFT potrebno je ulazni signal dopuniti nulama. Za definisanje nove dužine koristimo nextpow2 funkciju.

```
n = 2^nextpow2(L);
```

Argument dim u fft će u ovom slučaju biti 2.

```
dim = 2;
```

Pristupa se izračunavanju Furijeove transformacije signala.

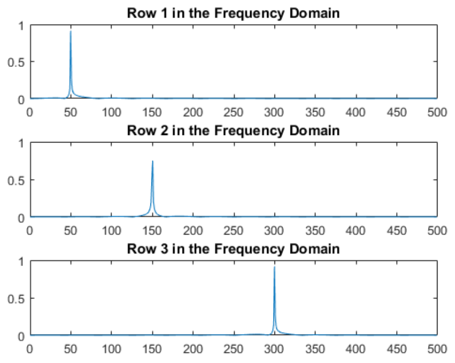
```
Y = fft(X,n,2);
```

Određuje se dvostrani i jednostrani spektar svakog signala.

```
P2 = abs(Y/n);
P1 = P2(:,1:n/2+1);
P1(:,2:end-1) = 2*P1(:,2:end-1);
```

U frekventnom domenu iscrtava se jednostrani amplitudni spektar za svaki red signala..

```
for i=1:3
   subplot(3,1,i)
   plot(0:(Fs/n):(Fs/2-Fs/n),P1(i,1:n/2))
   title(['Row ',num2str(i), ' in the Frequency Domain'])
end
```

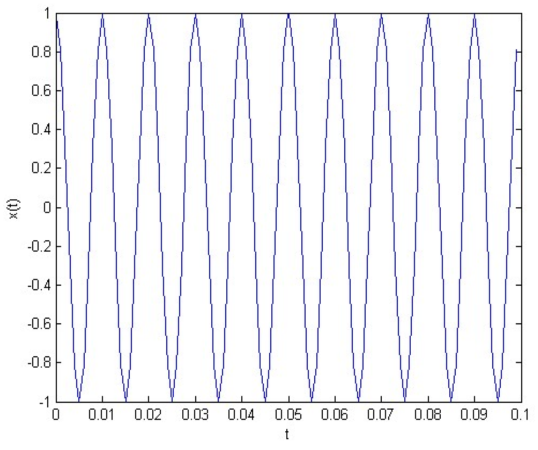


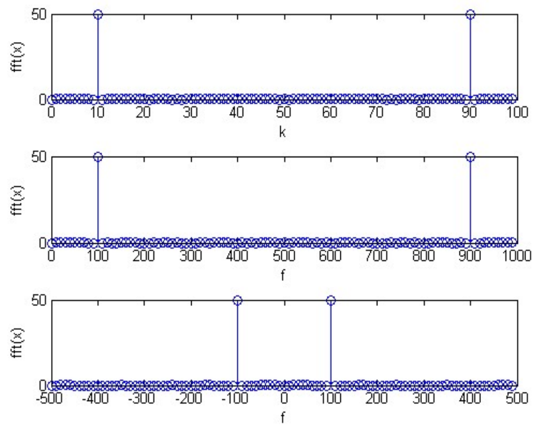
Slika 9: Cosine Waves- decimacija po frekvenciji

6. PRIMERI VIZUELIZACIJE DIGITALNOG SIGNALA

Primer 1: Sinusoida- odabiranje

```
close all; clear
fs=1000;
f=100;
N=100;
t=(0:N-1)'/fs; x=cos(2*pi*f*t);
figure,plot(t,x),xlabel('t'),ylabel('x(t)');
X=fft(x);
figure,subplot(3,1,1),stem((0:length(X)-1),abs(X)),
xlabel('k'),ylabel('fft(x)');
subplot(3,1,2),stem((0:length(X)-1)/length(X)*fs,abs(X)),
xlabel('f'),ylabel('fft(x)');
subplot(3,1,3),stem((0:length(X)-1)/length(X)*fs-
fs/2,fftshift(abs(X))),
xlabel('f'),ylabel('fft(x)');
```

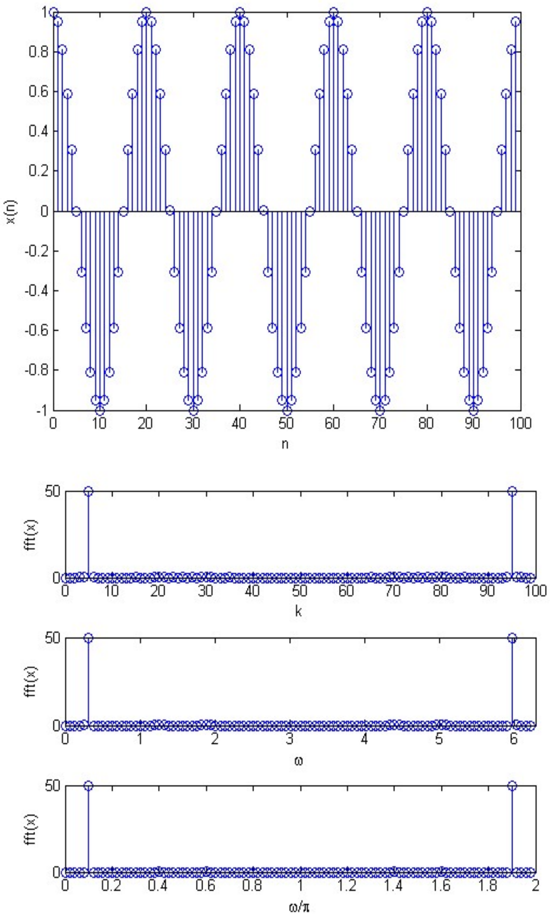




Slika 10: Vizuelizacija primera 1

Primer 2: Sinusoida- digitalni signal

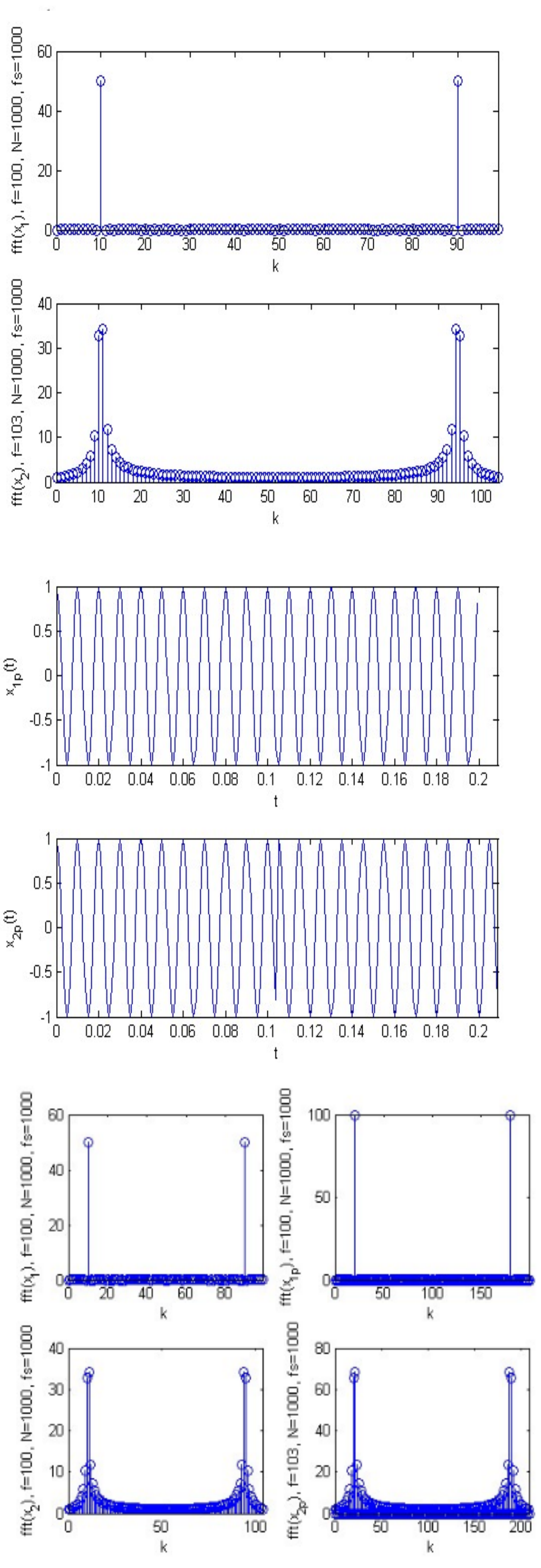
```
close all; clear
N=100;
n=(0:N-1)';
x=cos(2*pi*0.05*n);
figure,stem(n,x),xlabel('n'),ylabel('x(n)');
X=fft(x);
figure,subplot(3,1,1),stem((0:length(X)-1),abs(X)),
xlabel('k'),ylabel('fft(x)');
subplot(3,1,2),stem((0:length(X)-
1)/length(X)*2*pi,abs(X)),xlim([0 2*pi]),
xlabel('\omega'),ylabel('fft(x)');
subplot(3,1,3),stem((0:length(X)-
1)/length(X)*2,abs(X)),xlim([0 2]),
xlabel('\omega/\pi'),ylabel('fft(x)');
```



Slika 11: Vizuelizacija primera 2

Primer 3: Curenje spektra [9]

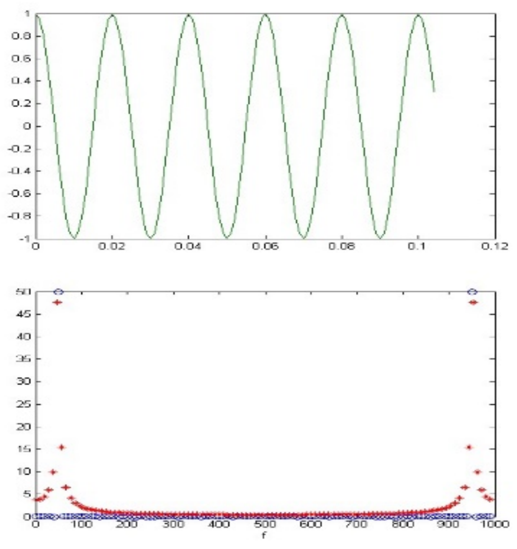
```
close all; clear fs=1000;
f1=100; f2=103; N=100;
t=(0:N-1)'/fs; x1=cos(2*pi*f1*t); x2=cos(2*pi*f2*t);
figure,subplot(2,1,1),plot(t,x1),xlim([0 t(end)]),
xlabel('t'),ylabel('x_1(t)');
subplot(2,1,2),plot(t,x2),xlim([0
t(end)]),xlabel('t'),ylabel('x_1(t)'); X1=fft(x1);
X2=fft(x2);
figure,subplot(2,1,1),stem((0:length(X1)-1),abs(X1)),
xlabel('k'),ylabel('fft(x_1), f=100, N=1000, fs=1000');
subplot(2,1,2),stem((0:length(X2)-1),abs(X2)),
xlabel('k'),ylabel('fft(x_2), f=103, N=1000, fs=1000');
tp=(0:2*N-1)'/fs;
xp1=[x1;x1];
xp2=[x2;x2];
figure,subplot(2,1,1),plot(tp,xp1),xlim([0 tp(end)]),
xlabel('t'),ylabel('x_{1p}(t)');
subplot(2,1,2),plot(tp,xp2),xlim([0 tp(end)]),
xlabel('t'),ylabel('x_{2p}(t)');
Xp1=fft(xp1);
Xp2=fft(xp2);
figure,subplot(2,2,1),stem((0:length(X1)-1), abs(X1)),
xlabel('k'), ylabel('fft(x_1), f=100, N=1000, fs=1000'),
xlim([0 length(X1)-1]);
subplot(2,2,2),stem((0:length(Xp1)-1), abs(Xp1)),
xlabel('k'),
ylabel('fft(x_{1p}), f=100, N=1000, fs=1000'),
xlim([0 length(Xp1)-1]);
subplot(2,2,3),stem((0:length(X2)-1),abs(X2)),xlabel('k'),
ylabel('fft(x_2), f=100, N=1000, fs=1000'),
xlim([0 length(X2)-1]);
subplot(2,2,4),stem((0:length(Xp2)-1), abs(Xp2)),
xlabel('k'), ylabel('fft(x_{2p}), f=103, N=1000, fs=1000'),
xlim([0 length(Xp2)-1]);
```



Slika 12: Vizuelizacija primera 3

primer 4: Rezolucija DFT-a

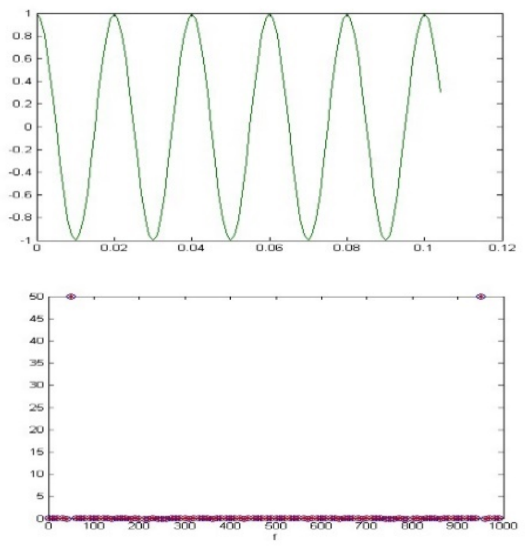
```
close all; clear fs=1000;
N1=100; N2=105;
t1=(0:N1-1)'/fs; t2=(0:N2-1)'/fs; f=50; x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1);
X2=fft(x2,N2);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
   fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```



Slika 13: Vizuelizacija primera 4

primer 5: fft sa dva argumenta

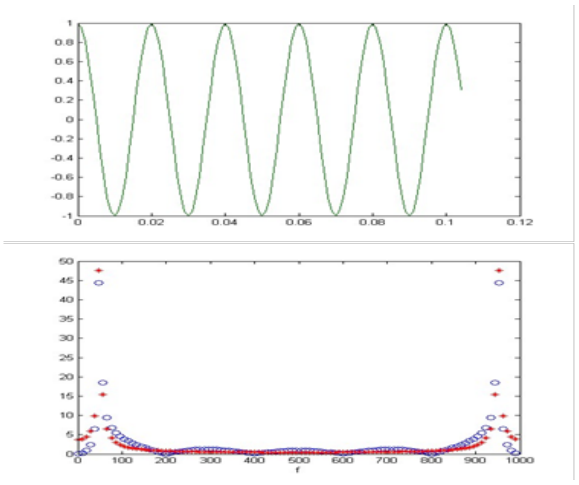
```
close all; clear fs=1000;
N1=100; N2=105;
t1=(0:N1-1)'/fs; t2=(0:N2-1)'/fs; f=50; x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1);
X2=fft(x2,N1);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
   fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```



Slika 14: Vizuelizacija primera 5

primer 6: fft sa dva argumenta

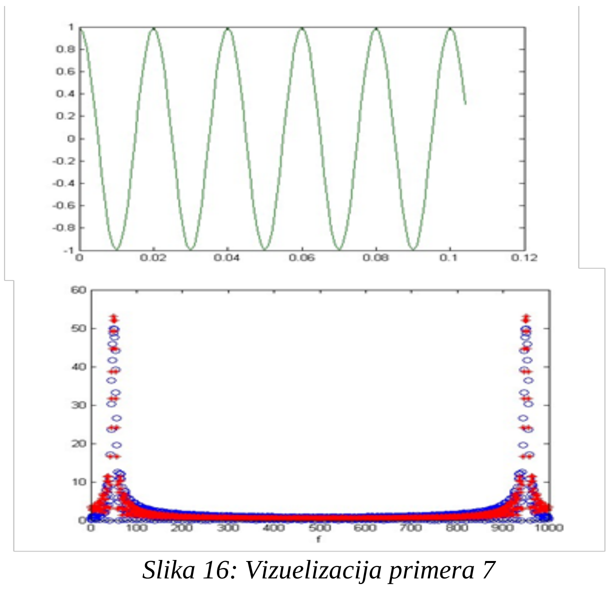
```
close all; clear fs=1000;
N1=100;
N2=105;
t1=(0:N1-1)'/fs;
t2=(0:N2-1)'/fs; f=50;
x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1,N2);
X2=fft(x2);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
   fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```



Slika 15: Vizuelizacija primera 6

primer 7: fft sa dva argumenta

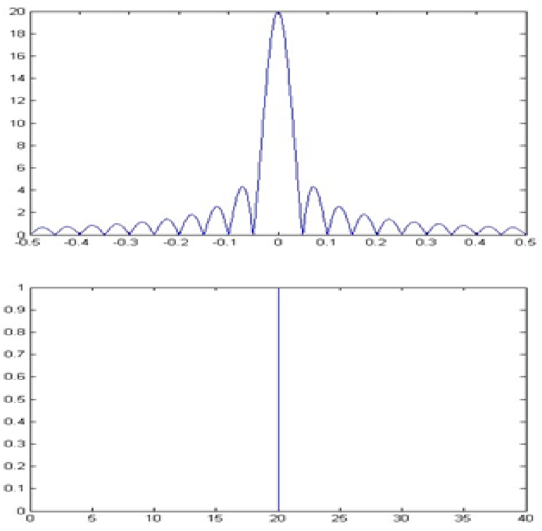
```
close all; clear fs=1000;
N1=100; N2=105;
t1=(0:N1-1)'/fs; t2=(0:N2-1)'/fs; f=50; x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1,1000);
X2=fft(x2,1000);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```



Slika 16: Vizuelizacija primera 7

primer 8: Usamljen pravougaoni impuls - TLKM 1

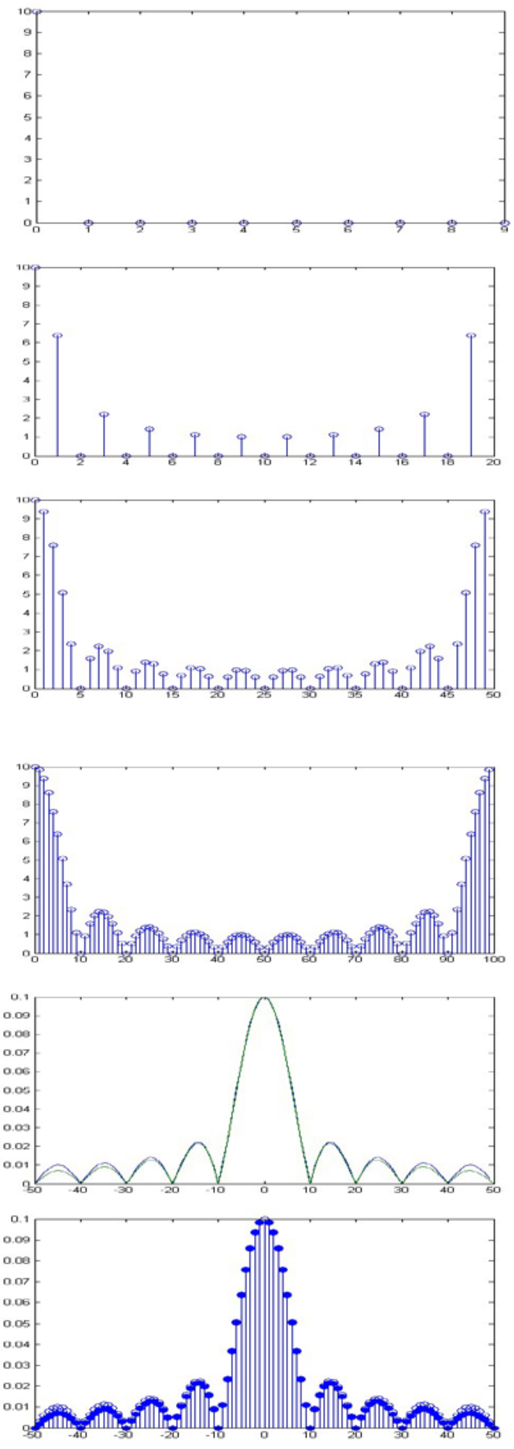
```
close all; clear;
tau=20;
fa=(-0.5:0.001:0.5);
Xa=tau*sin(2*pi*fa*tau/2)./(2*pi*fa*tau/2);
figure,plot(fa,abs(Xa));
ta=(0:0.0001:40);
xa=(ta<=tau);
figure,plot(ta,xa);
```



Slika 17: Vizuelizacija primera 8

primer 9: Dopunjavanje nulama [9]

```
close all; clear; N=10;
x=ones(N,1); X=fft(x);
figure,stem((0:length(X)-1),abs(X));
x1=x;
x1(2*N)=0;
X1=fft(x1);
figure,stem((0:length(X1)-1),abs(X1));
x2=x;
x2(5*N)=0;
X2=fft(x2);
figure,stem((0:length(X2)-1),abs(X2));
x3=x;
x3(10*N)=0;
X3=fft(x3);
figure,stem((0:length(X3)-1),abs(X3));
tau=0.1;
fa=[-50:0.01:50];
Xauc=tau*sin(2*pi*fa*tau/2)./(2*pi*fa*tau/2);
Xapc=tau*sin(2*pi*(-50:50)*tau/2)/(2*pi*(-50:50)*tau/2);
figure,plot((0:length(X3)-1)/length(X3)*2*50-50,
fftshift(abs(X3))/100,... fa,abs(Xauc));
figure,stem((0:length(X3)-1)/length(X3)*2*50-50,
fftshift(abs(X3))/100);
hold on
stem((-50:50),abs(Xapc),'filled');
```



Slika 18: Vizuelizacija primera 9

7. ZAKLJUČAK

U ovom radu je dat pregled nekih najosnovnijih primera digitalne obrade signala pomoću brze Furijeove transformcije. Svi navedeni primeri su realizovani u programskom paketu Matlab. Svaki primer je vizuelizovan i za svaki primer je dat detaljan programski kod. Primena programskog paketa Matlab u digitalnoj obradi signala, na ovakav način, umnogome pomaže objedinjavanju teorije i prakse i olakšava razumevanje izložene problematike.

LITERATURA

[1] Lj. Milić, Z. Dobrosavljević, J. Ćertić, "Uvod u digitalnu obradu signala", Akademska misao, Beograd, (2015)

[2] D. Radunović, A. Samardžić, F. Marić, "Numeričke metode – Zbirka zadataka kroz C, Fortran i Matlab", Akademska Misao, Beograd,(2005)

[3] T. Petrović, A. Rakić, "Signali i sistemi", Univerzitetski udžbenik, DEXIN, Beograd, (2005)

[4] Hwang, K., F. A. Briggs, "Computer Architecture and Parallel Processing", New York: McGraw-Hill, (1984)

[5] Solowiejczk, Y., J. Petzinger, "Large 1-D Fast Fourier Transforms on a Shared-Memory System", ICPP91 Proceedings, (1991)

[6] www.wikipedia.org

[7] E. Chu, "Discrete and Continuous Fourier Transforms", USA, Chapman & Hall/CRC, (2008)

.

[8] Frigo, M., and S. G. Johnson. "FFTW: An Adaptive Software Architecture for the FFT." *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3, 1998, pp. 1381-1384

[9] Mujo Hebibović, Predavanja iz predmeta Digitalni sistemi upravljanja, Elektrotehnički fakultet, Sarajevo, www.etf.unsa.ba, akademska 2008/2009. godina

Application of MATLAB Software in Digital Signal Processing by Fast Fourier Transform

Snezana Gavrilovic[1*], Nebojsa Denic[2], Jelena Eric-Obucina[1], Goran Miodragovic[1]
[1]School of Mechanical Engineering of Applied study Trstenik, Trstenik, Serbia
[2]Alfa Univerzitet, Beograd, Srbija

Theoretical bases of digital signal processing must be visualized in order to be comprehensible and applicable. Software package Matlab is an excellent partner in this mission. Thanks to its flexible environment, a wide range of built-in functions, and the possibility of algorithm developing and programming Matlab has become an indispensable tool in almost all areas of engineering. One of the most important algorithms for digital signal processing is fast Fourier transform whose function is built into Matlab. In this paper some of the basic examples of digital signal processing by means of fast Fourier transform and Matlab software are visualized.

Keywords: Matlab, digital signal, algorithm, fast Fourier transform

1. INTRODUCTION

In line with the development of information and communication technologies, digital signal processing has progressed rapidly in recent decades. Progress has been impressive, both in theory and in application domain. The development of digital signal processing associated with breakthroughs achieved in microelectronics, as well as in computer and software technology. With this overall progress, the development of modern systems is based mainly on digital technology. Digital signal processing is the basic technology for many branches of tehnical science such as telecommunications systems, mobile communications, radar, multimedia, speech processing, audio, sonar equipment, image processing, medical electronics, seismology, advanced techniques of measurement and control, robotics, mapping and many others.

Special beauty of digital signal processing is in the close relation of theory and practice. Software technologies that are available to us today to offer the possibility that already during the introduction to the basic algorithms of digital signal processing knowledge creatively applied to solve practical problems. One of the powerful software tools for processing and visualization of digital signal is a software package Matlab, MathWorks manufacturers. [1]

2. ABOUT MATLAB

Millions of engineers and scientists around the world use MATLAB for analysis and system design, development of algorithms and creation of models and applications. Matlab can be used for a wide range of applications, including signal processing and communications, image and video processing , testing and measurement, calculation of finance, etc. Includes mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, solving differential equations, which support the common engineering and science operations. Fundamental mathematical functions used the processor optimization libraries to provide fast performance vector and matrix calculations [2].

Matlab provides tools for analysis and visualization of data, which provides insight into the obtained information, and it takes less time than when using spreadsheets or traditional programming languages. It is possible to document and share their results as graphs of functions or in the form of a report.

Matlab provides access to data from different applications, databases, and external devices. Data can be read in the popular tools such as Microsoft Excel, it is possible to read text and binary files, image files, sound, and video.

In the Matlab programming language, it is possible to write a program or algorithm developed faster than with traditional programming languages because they do not need to perform low-level administrative processes such as declaration of variables, specification data types and memory allocation. Matlab offers the features of a traditional programming languages, such as error handling, flow control and object-oriented programming. You can use basic data types and advanced data structures and custom data types. Possible integration with other programming languages and applications.Possible is to share individual algorithm and Matlab application with other users or to develop them for those who do not use Matlab. It is possible to design and edit custom graphical interface.

3. SIGNAL AND DIGITAL PROCESSING SIGNAL

Signal is the holder of certain information. The content of the information that the signal carries is written a change in the function or one of its parameters, and the signal can therefore be defined as a function of time, which carries the information about a value of interest. There are two types of signals: continuous and discrete.

Under continuous signal we mean those in which the function that a signal indicating a continuous function of time. If the amplitude of the signal may take an arbitrary value of the permissible range, the signal is called analog signal.

Discrete signals are defined only for discrete values independent of the variable time. Discrete signals generated discretization of analog signals with certain conditions.

The amplitude discrete signal can be continuous or discrete. If quantization is carried amplitude discrete signal obtained by the digital signal is [3].

Digital Signal Processing (DOS) represents change and / or analysis of information contained in discrete

*Corresponding author: School of Mechanical Engineering of Applied study, Radoja Krstića 19, 37240 Trstenik, Serbia, e-mail: gavrilovicsnezana@yahoo.com

sequences of numbers. The signals coming from the real world, which points to the need that the processing is performed in real time and that the signals are converted into digital form. The signals are discrete, which means that the information from discrete values lost and should be restored. There are various algorithms and software packages for digital signal processing.

In the program's package Matlab, Signal Processing Toolbox provides functions and applications to generate, measure, transform, filter and visualize signals. The toolbox includes algorithms for resampling, smoothing, and synchronizing signals, designing and analyzing filters, estimating power spectra, and measuring peaks, bandwidth, and distortion. The toolbox also includes parametric and linear predictive modeling algorithms. You can use Signal Processing Toolbox to analyze and compare signals in time, frequency, and time-frequency domains, identify patterns and trends, extract features, and develop and validate custom algorithms to gain insight into your data. FFT algorithm in DOS are widely used as a built-in function in the software package Matlab. [4]

4. FAST FOURIER TRANSFORMATION

Fast Fourier transform (FFT) algorithm has a "fast" calculating the value of discrete Fourier transform (DFT). The acceleration compared to the usual procedure of calculating the discrete Fourier transformation is achieved by avoiding recalculation terms that negate each other. Algorithm "divide and conquer" is attributed to James W. Cooley and John W. Tukey who published it in 1965. [5]

Three main steps principle "The Divide and Conquer Paradigm" are:

• divide sequence into two or more subsequences,

• solve each sequence recursively using the same algorithm and set the boundary conditions to end the recursion when the length of subsequence is sufficiently small and

• Prepare a solutions of the original sequence by combining solutions subsequence.

Intuitively, it is clear that the division algorithm and resolve to have the best results if the length of the sequence N, the number of exponent 2, because the division of the sequence into successively smaller groups can continue until only one couple in the group. Also, there are many cases where N is not an exponent of the number 2 and it is necessary to modify the algorithm. Starts from the assumption that $N = 2^n$.

FFT algorithm the base 2 is a recursive algorithm, which consists of dividing the obtained sequences (and any subsequence) in the two subsequences of half the length. There are two commonly used FFT algorithm, which differ in the way in which the defined subsequence. These are the fast Fourier transform with decimation in time (DIT FFT) and fast Fourier transform with decimation in frequency (DIF FFT). [6]

4.1. Decimation by time

One-dimensional array x (n) decompose in two, so that the first array of f1 (n) are just a array of vaporous samples x (n), and the second f2 (n) odd uzorci.Then the N-point DFT has the form: [7]

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} =$$

$$k = 0,1,\dots,N-1$$

$$= \sum_{n\ je\ paran} x(n) W_N^{kn} + \sum_{n\ je\ neparan} x(n) W_N^{kn} =$$

$$= \sum_{m=0}^{\frac{N}{2}-1} x(2m) W_N^{2km} + \sum_{m=0}^{\frac{N}{2}-1} x(2m+1) W_N^{(2m+1)k} =$$

$$= \sum_{m=0}^{\frac{N}{2}-1} f_1(n) W_{\frac{N}{2}}^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_2(n) W_{\frac{N}{2}}^{km} =$$

$$= F_1(k) + W_N^k F_2(k), \quad k = 0,1,\dots,N/2-1$$

$F_1(k)=\sum_{m=0}^{\frac{N}{2}-1} f_1(n) W_{\frac{N}{2}}^{km}$, i

$F_2(k) = \sum_{m=0}^{\frac{N}{2}-1} f_2(n) W_{\frac{N}{2}}^{km}$

the values of the DFT in N/2 points and periodic with period N/2. $W_N^k = e^{\frac{jk2\pi}{N}}$ is the rotational factor.

Using this procedure, N-member DFT is reduced to two N/2-members DFT. Following the same procedure decimation, each N/2-member DFT can be reduced to two N/4-members DFT, and so on until the calculation is reduced to 2-members DFT. Below is a detailed view of calculating the FFT decimation in time for the case where N = 8.

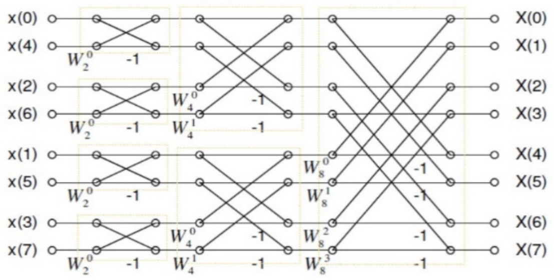


Figure 1. DIT FFT

4.2. Decimation by frequency

One-dimensional array x (n) decompose in two, so that the first array is first half n = N / 2 elements of the array x (n) in the second part is second half of x (n) . Then the N-point DFT has the form: [7]

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} =$$

$$k = 0,1,\dots,N-1$$

$$= \sum_{prvih\ n} x(n) W_N^{kn} + \sum_{drugih\ n} x(n) W_N^{kn} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=N/2}^{N-1} x(n) W_N^{kn} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{(n+\frac{N}{2})k} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + W_N^{\frac{N}{2}k} \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{nk} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{kn} + (-1)^k \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)W_N^{nk},$$
$$k = 0,1,\dots,N/2-1$$

The procedure is analoguos to the execution of an algorithm as in the decimation in time. Below is a detailed view of calculating the FFT decimation in frequency for the case where N=8.

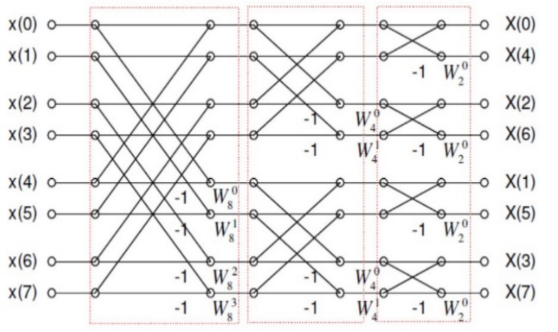


Figure 2. DIF FFT

5. DESCRIPTION OF FUNCTION FFT IN MATLAB

Y = fft(X) computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.[8]

• If X is a vector, then fft(X) returns the Fourier transform of the vector.

• If X is a matrix, then fft(X) treats the columns of X as vectors and returns the Fourier transform of each column.

• If X is a multidimensional array, then fft(X) treats the values along the first array dimension whose size does not equal 1 as vectors and returns the Fourier transform of each vector.

Example: Noisy Signal

Use Fourier transforms to find the frequency components of a signal buried in noise.

Specify the parameters of a signal with a sampling frequency of 1 kHz and a signal duration of 1 second.

```
Fs = 1000;          % Sampling frequency
T = 1/Fs;           % Sampling period
L = 1000;           % Length of signal
t = (0:L-1)*T;      % Time vector
```

Form a signal containing a 50 Hz sinusoid of amplitude 0.7 and a 120 Hz sinusoid of amplitude 1.

```
S = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
```

Corrupt the signal with zero-mean white noise with a variance of 4.

```
X = S + 2*randn(size(t));
```

Plot the noisy signal in the time domain. It is difficult to identify the frequency components by looking at the signal X(t).

```
plot(1000*t(1:50),X(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('t (milliseconds)')
ylabel('X(t)')
```

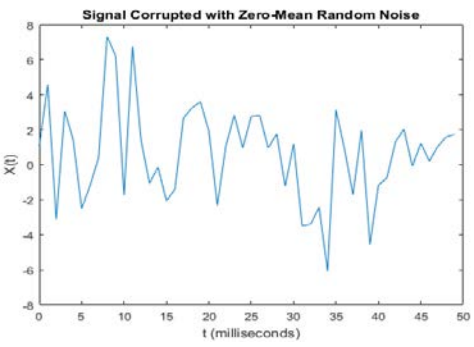


Figure 3: Noisy Signal- Decimation by time

Compute the Fourier transform of the signal.

```
Y = fft(X);
```

Compute the two-sided spectrum P2. Then compute the single-sided spectrum P1 based on P2 and the even-valued signal length L.

```
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
```

Define the frequency domain f and plot the single-sided amplitude spectrum P1. The amplitudes are not exactly at 0.7 and 1, as expected, because of the added noise. On average, longer signals produce better frequency approximations.

```
f = Fs*(0:(L/2))/L;
plot(f,P1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

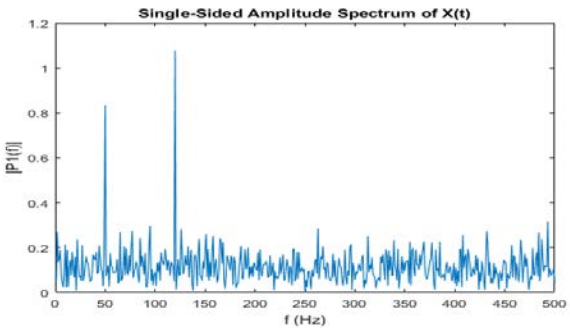


Figure 4: Noisy Signal- Decimation by frequency X(t)

Now, take the Fourier transform of the original, uncorrupted signal and retrieve the exact amplitudes, 0.7 and 1.0.

```
Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);

plot(f,P1)
title('Single-Sided Amplitude Spectrum of S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

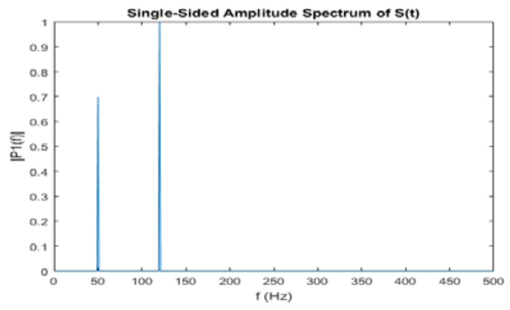


Figure 5: Noisy Signal- Decimation by frequency S(t)

Y = fft(X,n) returns the n-point DFT. If no value is specified, Y is the same size as X.[8]

• If X is a vector and the length of X is less than n, then X is padded with trailing zeros to length n.

• If X is a vector and the length of X is greater than n, then X is truncated to length n.

• If X is a matrix, then each column is treated as in the vector case.

• If X is a multidimensional array, then the first array dimension whose size does not equal 1 is treated as in the vector case.

Example: Gaussian Pulse

Convert a Gaussian pulse from the time domain to the frequency domain.

Define signal parameters and a Gaussian pulse, X.

```
Fs = 100;          % Sampling frequency
t = -0.5:1/Fs:0.5;  % Time vector
L = length(t);      % Signal length
X = 1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));
```

Plot the pulse in the time domain.

```
plot(t,X)
title('Gaussian Pulse in Time Domain')
xlabel('Time (t)')
ylabel('X(t)')
```

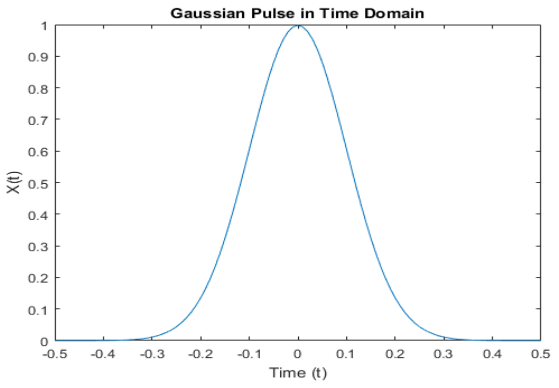


Figure 6: Gaussian Pulse- Decimation by time

To use the fft function to convert the signal to the frequency domain, first identify a new input length that is the next power of 2 from the original signal length. This will pad the signal X with trailing zeros in order to improve the performance of fft.

```
n = 2^nextpow2(L);
```

Convert the Gaussian pulse to the frequency domain.

```
Y = fft(X,n);
```

Define the frequency domain and plot the unique frequencies.

```
f = Fs*(0:(n/2))/n;
P = abs(Y/n);
plot(f,P(1:n/2+1))
title('Gaussian Pulse in Frequency Domain')
xlabel('Frequency (f)')
ylabel('|P(f)|')
```

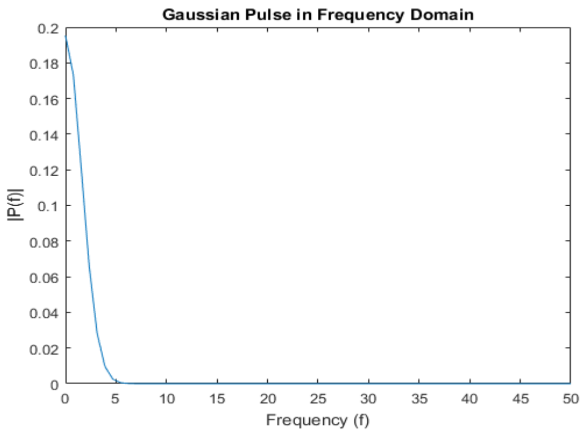


Figure 7: Gaussian Pulse- Decimation by frequency

Y = fft(X,n,dim) returns the Fourier transform along the dimension dim. For example, if X is a matrix, then fft(X,n,2) returns the n-point Fourier transform of each row.[8]

Example: Cosine Waves

Compare cosine waves in the time domain and the frequency domain.

Specify the parameters of a signal with a sampling frequency of 1kHz and a signal duration of 1 second.

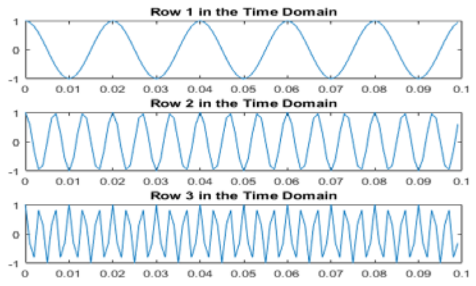
```
Fs = 1000;              % Sampling frequency
T = 1/Fs;               % Sampling period
L = 1000;               % Length of signal
t = (0:L-1)*T;           % Time vector
```

Create a matrix where each row represents a cosine wave with scaled frequency. The result, X, is a 3-by-1000 matrix. The first row has a wave frequency of 50, the second row has a wave frequency of 150, and the third row has a wave frequency of 300.

```
x1 = cos(2*pi*50*t);        % First row wave
x2 = cos(2*pi*150*t);       % Second row wave
x3 = cos(2*pi*300*t);       % Third row wave
X = [x1; x2; x3];
```

Plot the first 100 entries from each row of X in a single figure in order and compare their frequencies.

```
for i = 1:3
   subplot(3,1,i)
   plot(t(1:100),X(i,1:100))
   title(['Row ',num2str(i),' in the Time Domain'])
end
```



Slika 8: Cosine Waves- Decimation by time

For algorithm performance purposes, fft allows you to pad the input with trailing zeros. In this case, pad each row of X with zeros so that the length of each row is the next higher power of 2 from the current length. Define the new length using the nextpow2 function.

n = 2^nextpow2(L);

Specify the dim argument to use fft along the rows of X, that is, for each signal.

dim = 2;

Compute the Fourier transform of the signals.

Y = fft(X,n,dim);

Calculate the double-sided spectrum and single-sided spectrum of each signal.

P2 = abs(Y/n);

P1 = P2(:,1:n/2+1);

P1(:,2:end-1) = 2*P1(:,2:end-1);

In the frequency domain, plot the single-sided amplitude spectrum for each row in a single figure.

```
for i=1:3
   subplot(3,1,i)
   plot(0:(Fs/n):(Fs/2-Fs/n),P1(i,1:n/2))
   title(['Row ',num2str(i), ' in the Frequency Domain'])
end
```

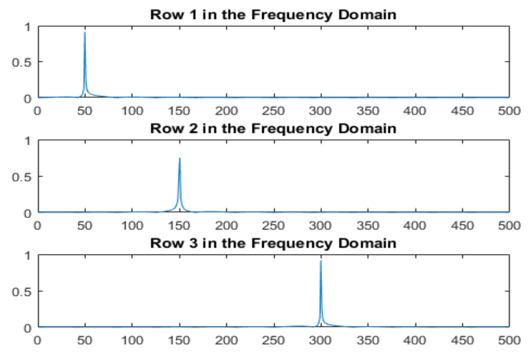
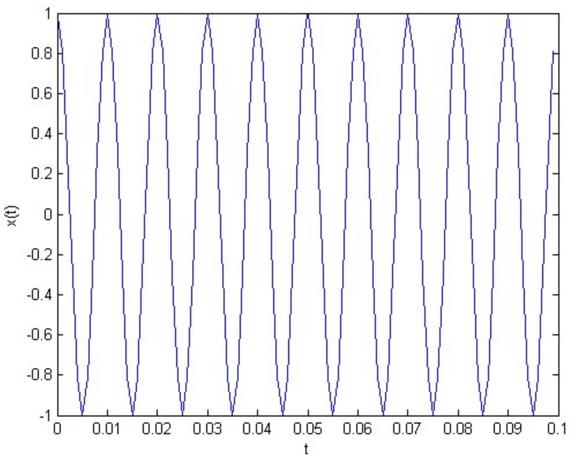


Figure 9: Cosine Waves- Decimation by frequency

6. EXAMPLES OF VISUALIZATION DIGITAL SIGNAL

Example 1: Sine wave- discretization

```
close all; clear
fs=1000;
f=100;
N=100;
t=(0:N-1)'/fs; x=cos(2*pi*f*t);
figure,plot(t,x),xlabel('t'),ylabel('x(t)');
X=fft(x);
figure,subplot(3,1,1),stem((0:length(X)-1),abs(X)),
xlabel('k'),ylabel('fft(x)');
subplot(3,1,2),stem((0:length(X)-1)/length(X)*fs,abs(X)),
xlabel('f'),ylabel('fft(x)');
subplot(3,1,3),stem((0:length(X)-1)/length(X)*fs-fs/2,fftshift(abs(X))),
xlabel('f'),ylabel('fft(x)');
```



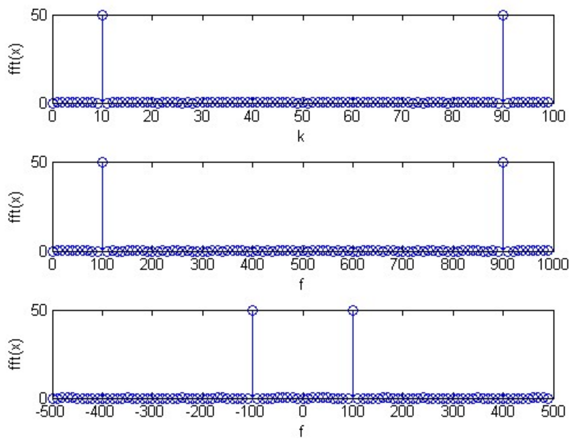


Figure 10: Visualization of example 1

Example 2: Sine wave - digital signal

```
close all; clear
N=100;
n=(0:N-1)';
x=cos(2*pi*0.05*n);
figure,stem(n,x),xlabel('n'),ylabel('x(n)');
X=fft(x);
figure,subplot(3,1,1),stem((0:length(X)-1),abs(X)),
xlabel('k'),ylabel('fft(x)');
subplot(3,1,2),stem((0:length(X)-1)/length(X)*2*pi,abs(X)),xlim([0 2*pi]),
xlabel('\omega'),ylabel('fft(x)');
subplot(3,1,3),stem((0:length(X)-1)/length(X)*2,abs(X)),xlim([0 2]),
xlabel('\omega/\pi'),ylabel('fft(x)');
```

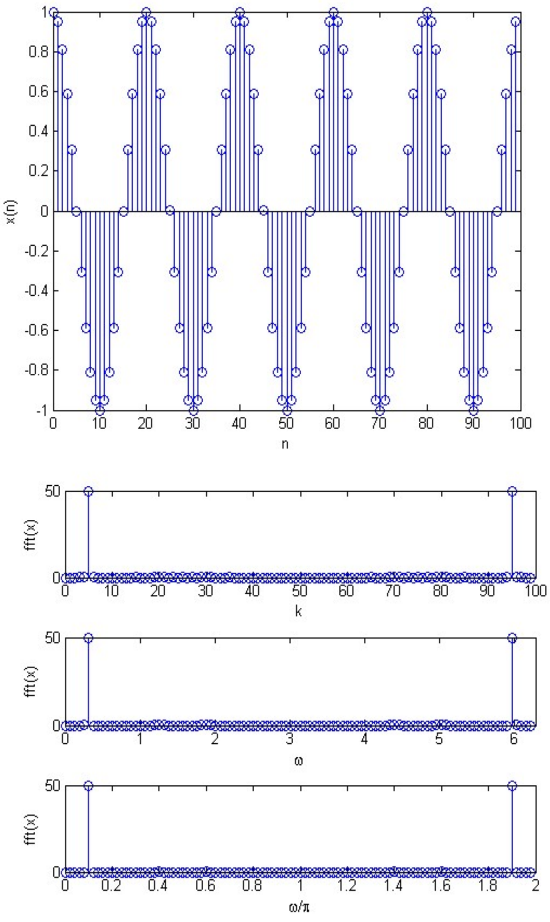


Figure 11: Visualization of example 2

Example 3: Spectral leakage [9]

```
close all; clear fs=1000;
f1=100; f2=103; N=100;
t=(0:N-1)'/fs; x1=cos(2*pi*f1*t); x2=cos(2*pi*f2*t);
figure,subplot(2,1,1),plot(t,x1),xlim([0 t(end)]),
xlabel('t'),ylabel('x_1(t)');
subplot(2,1,2),plot(t,x2),xlim([0
t(end)]),xlabel('t'),ylabel('x_1(t)'); X1=fft(x1);
X2=fft(x2);
figure,subplot(2,1,1),stem((0:length(X1)-1),abs(X1)),
xlabel('k'),ylabel('fft(x_1), f=100, N=1000, fs=1000');
subplot(2,1,2),stem((0:length(X2)-1),abs(X2)),
xlabel('k'),ylabel('fft(x_2), f=103, N=1000, fs=1000');
tp=(0:2*N-1)'/fs;
xp1=[x1;x1];
xp2=[x2;x2];
figure,subplot(2,1,1),plot(tp,xp1),xlim([0 tp(end)]),
xlabel('t'),ylabel('x_{1p}(t)');
subplot(2,1,2),plot(tp,xp2),xlim([0 tp(end)]),
xlabel('t'),ylabel('x_{2p}(t)');
Xp1=fft(xp1);
Xp2=fft(xp2);
figure,subplot(2,2,1),stem((0:length(X1)-1), abs(X1)),
xlabel('k'), ylabel('fft(x_1), f=100, N=1000, fs=1000'),
xlim([0 length(X1)-1]);
subplot(2,2,2),stem((0:length(Xp1)-1), abs(Xp1)),
xlabel('k'),
ylabel('fft(x_{1p}), f=100, N=1000, fs=1000'),
xlim([0 length(Xp1)-1]);
subplot(2,2,3),stem((0:length(X2)-1),abs(X2)),xlabel('k'),
ylabel('fft(x_2), f=100, N=1000, fs=1000'),
xlim([0 length(X2)-1]);
subplot(2,2,4),stem((0:length(Xp2)-1), abs(Xp2)),
xlabel('k'), ylabel('fft(x_{2p}), f=103, N=1000, fs=1000'),
xlim([0 length(Xp2)-1]);
```

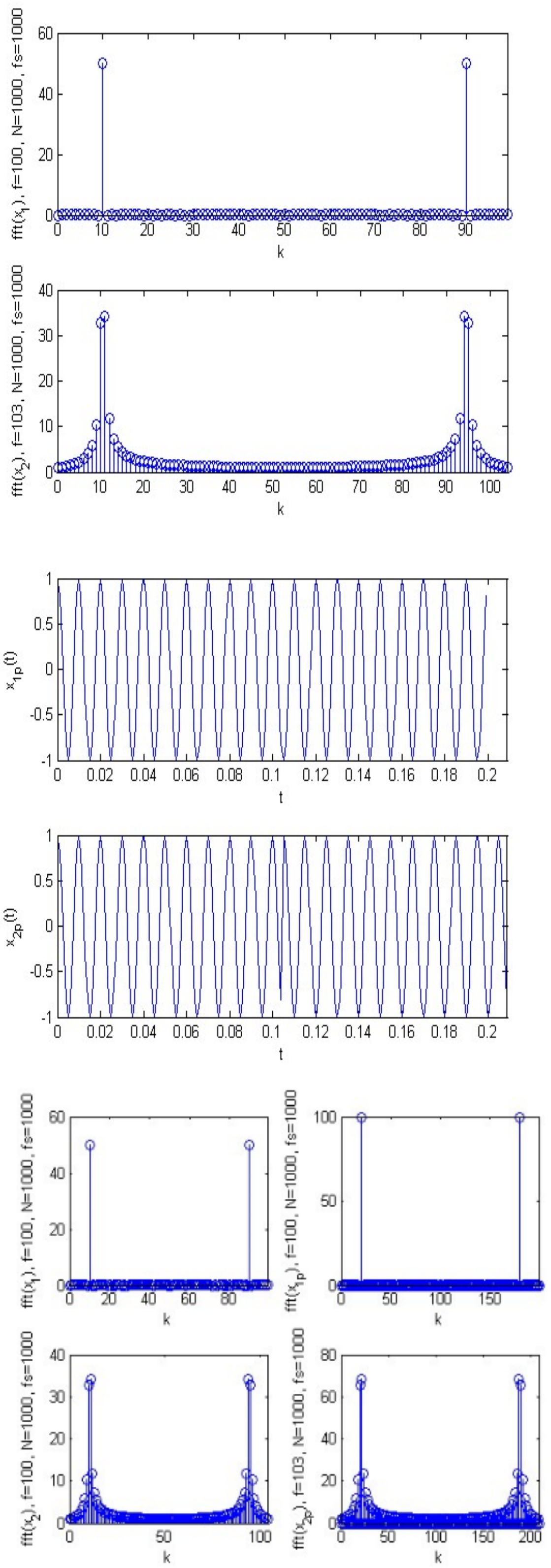


Figure 12: Visualization of example 3

Example 4: Resolution DFT

```
close all; clear fs=1000;
N1=100; N2=105;
t1=(0:N1-1)'/fs; t2=(0:N2-1)'/fs; f=50; x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1);
X2=fft(x2,N2);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
   fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```

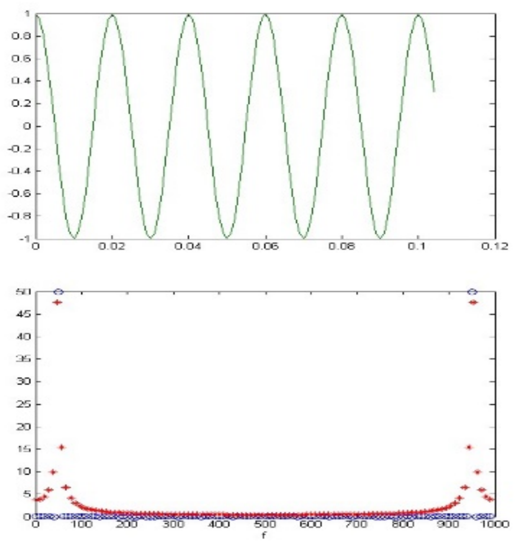


Figure 13: Visualization of example 4

Example 5: FFT with two arguments

```
close all; clear fs=1000;
N1=100; N2=105;
t1=(0:N1-1)'/fs; t2=(0:N2-1)'/fs; f=50; x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1);
X2=fft(x2,N1);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
   fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```

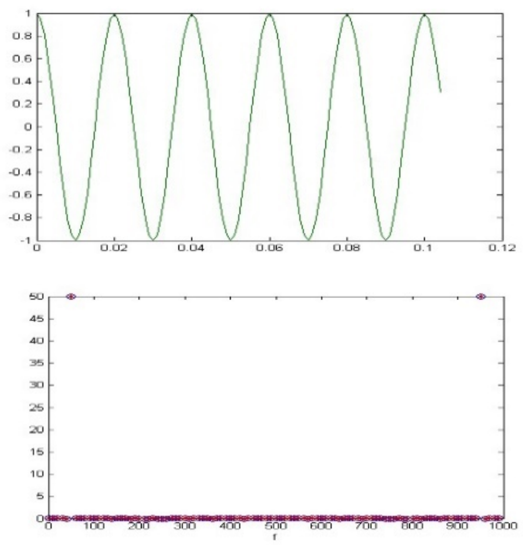


Figure 14: Visualization of example 5

Example 6: FFT with two arguments

```
close all; clear fs=1000;
N1=100;
N2=105;
t1=(0:N1-1)'/fs;
t2=(0:N2-1)'/fs; f=50;
x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1,N2);
X2=fft(x2);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
   fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```

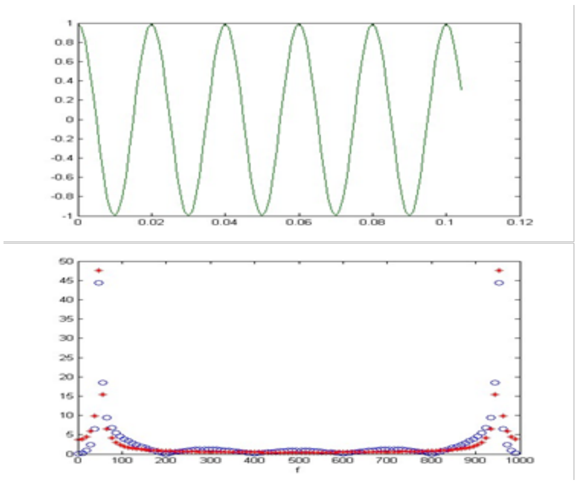


Figure 15: Visualization of example 6

Example 7: FFT with two arguments

```
close all; clear fs=1000;
N1=100; N2=105;
t1=(0:N1-1)'/fs; t2=(0:N2-1)'/fs; f=50; x1=cos(2*pi*f*t1);
x2=cos(2*pi*f*t2);
figure,plot(t1,x1,t2,x2); X1=fft(x1,1000);
X2=fft(x2,1000);
figure,plot(fs*(0:length(X1)-1)/length(X1),abs(X1),'bo',...
fs*(0:length(X2)-1)/length(X2),abs(X2),'r*'),xlabel('f');
```

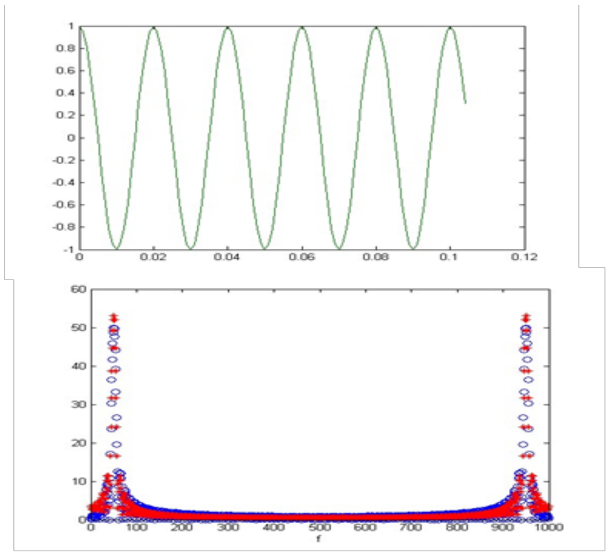


Figure 16: Visualization of example 7

Example 8: Unfrequented rectangular impulse [9]

```
close all; clear;
tau=20;
fa=(-0.5:0.001:0.5);
Xa=tau*sin(2*pi*fa*tau/2)./(2*pi*fa*tau/2);
figure,plot(fa,abs(Xa));
ta=(0:0.0001:40);
xa=(ta<=tau);
figure,plot(ta,xa);
```

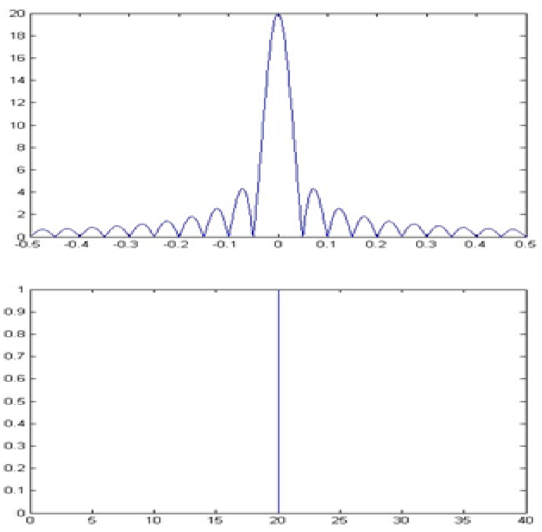


Figure 17: Visualization of example 8

Example 9: Complement of zeros [10]

```
close all; clear; N=10;
x=ones(N,1); X=fft(x);
figure,stem((0:length(X)-1),abs(X));
x1=x;
x1(2*N)=0;
X1=fft(x1);
figure,stem((0:length(X1)-1),abs(X1));
x2=x;
x2(5*N)=0;
X2=fft(x2);
figure,stem((0:length(X2)-1),abs(X2));
x3=x;
x3(10*N)=0;
X3=fft(x3);
figure,stem((0:length(X3)-1),abs(X3));
tau=0.1;
fa=[-50:0.01:50];
Xauc=tau*sin(2*pi*fa*tau/2)./(2*pi*fa*tau/2);
Xapc=tau*sin(2*pi*(-50:50)*tau/2)/(2*pi*(-50:50)*tau/2);
figure,plot((0:length(X3)-1)/length(X3)*2*50-50,
fftshift(abs(X3))/100,... fa,abs(Xauc));
figure,stem((0:length(X3)-1)/length(X3)*2*50-50,
fftshift(abs(X3))/100);
hold on
stem((-50:50),abs(Xapc),'filled');
```

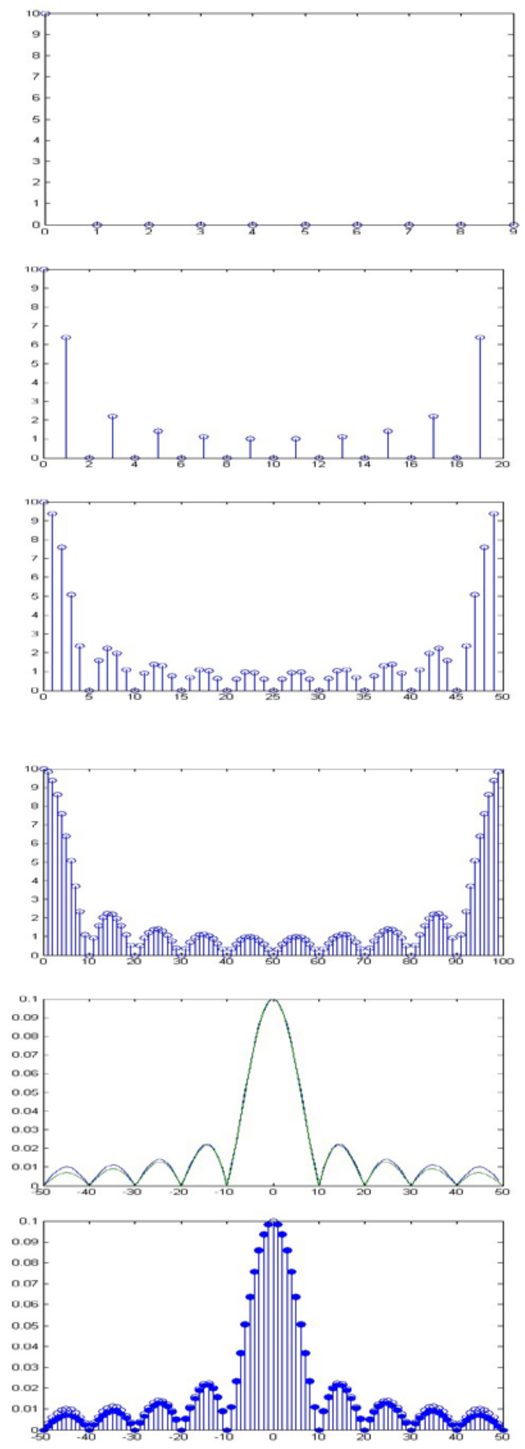


Figure 18: Visualization of example 9

7. CONCLUSION

This paper gives an overview of some of the most basic examples of digital signal processing using Fast Fourier Transformation. All of the above examples have been implemented in the software package Matlab. Each example is visualized and for each example is given a detailed program code. Application of Matlab in digital signal processing, in this way, greatly helps the unification of theory and practice, and it facilitates an understanding of the problems exposed.

8.REFERENCES

[1] Lj. Milić, Z. Dobrosavljević, J. Ćertić, "Uvod u digitalnu obradu signala", Akademska misao, Beograd, (2015)

[2] D. Radunović, A. Samardžić, F. Marić, "Numeričke metode – Zbirka zadataka kroz C, Fortran i Matlab", Akademska Misao, Beograd,(2005)

[3] T. Petrović, A. Rakić, "Signali i sistemi", Univerzitetski udžbenik, DEXIN, Beograd, (2005)

[4] Hwang, K., F. A. Briggs, "Computer Architecture and Parallel Processing", New York: McGraw-Hill, (1984)

[5] Solowiejczk, Y., J. Petzinger, "Large 1-D Fast Fourier Transforms on a Shared-Memory System", ICPP91 Proceedings, (1991)

[6] www.wikipedia.org

[7] E. Chu, "Discrete and Continuous Fourier Transforms", USA, Chapman & Hall/CRC, (2008)

[8] Frigo, M., and S. G. Johnson. "FFTW: An Adaptive Software Architecture for the FFT." *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3, 1998, pp. 1381-1384

[9] http://telekomunikacije.etf.bg.ac.rs/lab54/os1/vezbe_09_10/v_2009_11_10.html

[10] Mujo Hebibović, Predavanja iz predmeta Digitalni sistemi upravljanja, Elektrotehnički fakultet, Sarajevo, www.etf.unsa.ba, akademska 2008/2009. godina