UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET U SARAJEVU

-ZAVRŠNI RAD -

Praktikum za laboratorijske vježbe iz

„Signala i sistema“

Mentor:
R. prof.dr Melita Ahić-Đokić, dipl.ing.el

Kandidat:
Bukva Emina

Sarajevo, septembar 2010. godine

SAŽETAK

Praktikum je izrađen za potrebe odvijanja nastave na predmetima Teorija signala i Analiza signala i sistema na Elektrotehničkom fakultetu u Sarajevu, Univerziteta u Sarajevu. Tematske cjeline koje obrađuje ovaj praktikum odgovaraju onim obrađenim unutar kursa. Osnovni cilj ovog praktikuma je pomoći studentima da kroz samostalan rad u vidu pripreme za laboratorijsku vježbu i interaktivne zadatke koji će biti izvedeni u toku sati rada u laboratoriji primjene stečeno teoretsko znanje na praktične probleme u obradi signala.

Praktikum se sastoji iz dva poglavlja. Prvo poglavlje se bavi osnovnim pojmovima i konceptima rada u programskom okruženju MATLAB-a i Simulink-a. Razvojna okruženje MATLAB-a i Simulink-a su izabrana zbog velikog broja mogućnosti za obradu signala koju pružaju kao i zato što su u širokoj upotrebi u polju procesiranja signala. S obzirom da se studenti prije ovog kursa nisu imali priliku susresti sa radom u MATLAB-u, ovo poglavlje je predviđeno da im pomogne da se upoznaju sa radnim okruženjem i mogućnostima koje nudi MATLAB.

Drugo poglavlje sadrži šest laboratorijskih vježbi koje se sastoje iz dva dijela. Prvi dio je uvod u laboratorijsku vježbu u kojem su navedene potrebne teoreme, matematičke formule i sl. potrebne za pravilno izvođenje laboratorijskih vježbi. Ovaj dio sadrži i riješene primjere iz razmatrane oblasti koje pojašnjavaju metode korištenje MATLAB-ovih resursa, odnosno preddefiniranih funkcija koje ćemo koristiti za rješavanje pojedinih problema. Drugi segment je sam laboratorijski zadatak, koji je podijeljen u niz manjih zadataka i eventualnu pripremu potrebnu za uspješno obavljanje zadataka, koju je student dužan ponijeti sa sobom na laboratorijsku vježbu.

ABSTRACT

This laboratory manual is designed for the course of instruction in the subjects Theory of Signals and Signals and Systems Analysis in Electrical Engineering Faculty in Sarajevo, University of Sarajevo. Teaching units processed in this manual are all covered in the course. The main objective of this laboratory manual is to help students apply acquired theoretical knowledge of signal processing to practical problems, through individual work in the form of preparation exercises and interactive tasks to be performed during the hours in the laboratory.

The manual consists of two chapters. The first chapter deals with basic terms and concepts of the programming in the MATLAB and Simulink environment. MATLAB and Simulink are chosen because of the large number of methods for processing signals they provide and because they are widely used in the field of signal processing. Given that most students have no experience with MATLAB or Simulink prior to this course, this chapter is intended to help them familiarize with the working environment and opportunities provided by MATLAB.

The second chapter contains six laboratory excercises that consist of two parts. The first part is an introduction to the laboratory exercise, which contains the necessary theorems, mathematical formulas, etc. necessary for the proper performance of laboratory exercises. This section contains solved examples from the selected topic that help demonstrate how to use MATLAB's resources, like predefined functions that will be used to solve particular problems. The second part is the laboratory task, which is divided into several smaller tasks, and the preparation necessary to successfully perform the task that the student is required to bring with them to the laboratory exercise.

SADRŽAJ

INDEX SLIKA

UVOD

Praktikum za laboratorijske vježbe iz “Signala i sistema”, ima za cilj ilustrirati metode i probleme koji se javljaju u području akvizicije i obrade signala, obrađene na satima predavanja.

Praktikum predstavlja dopunu predavanjima, te se pretpostavlja da su studenti prije početka vježbi savladali gradivo obrađeno na predavanjima i u prethodnim kursevima. Umjesto teoretskih izlaganja, u uvodu laboratorijskih vježbi je dat samo kratak pregled matematičkih relacija i teorema koje će biti demonstrirane u sklopu date vježbe.

Kao alat obrade signala odabran je programski paket MATLAB/Simulink. Iako su integrisani u jedinstveno razvojno okruženje, važno je naglasiti da postoji velika razlika u samom pristupu zadatku između MATLAB-a i Simulink-a. MATLAB predstavlja programski jezik koji je više orijentisan na obradu podataka predstavljenih u vidu matrica, te omogućava sekvencijalno izvođenje naredbi za procesiranje i prikaz podataka, ili kreiranje novih signala kombiniranjem postojećih. Simulink, s druge strane pruža velike mogućnosti analize i ispitivanja veoma složenih sistema kreiranih u vidu blok dijagrama. Mogućnosti obrade signala koje MATLAB i Simulink posjeduju nadzilaze one izložene u ovom praktikumu, te zbog svog obima mogu čak biti predmetom doživotnog izučavanja jednog inženjera.

Vježbe su sačinjene iz dva segmenta. Prvi segment predstavlja uvod u problem koji će biti razmotren u sklopu laboratorijske vježbe. Predviđeno je da će ovaj dio laboratorijske vježbe student proučiti samostalno prije dolaska na laboratorijske vježbe. Cilj mu je studenta podsjetiti na osnove relacije, izraze i definicije koje su potrebne za razumijevanje vježbe. Pored navedenog sadrži i nekoliko riješenih primjera iz razmatrane oblasti, koji ilustrativno prikazuju način upotrebe funkcija kojima raspolaže MATLAB koje studentu pomažu pri rješavanju problema.

Drugi dio laboratorijske vježbe predstavlja laboratorijski zadatak. Cilj je bio pronaći i odabrati kao laboratorijske zadatke primjere iz prakse, adekvatne kompleksnosti koji na interesantan način demonstriraju načine prikupljanja i obrade signala obrađene na nastavi. Korištene metode obrade signala su uključivale metode obrade signala u vremenskom i frekventnom domenu. Laboratorijske vježbe uključuju pripremu za laboratorijsku vježbu, koju čine jednostavni zadaci koji studentima služe da se bolje upoznaju sa izučavanom tematikom. Sam laboratorijski zadatak je većeg obima, ali je uvijek postavljen u vidu više manjih zadataka radi bolje razumljivosti problema.

POSTAVKA ZADATKA

Red. prof. dr Melita Ahić-Đokić, dipl.el.inž.
Asistent Emir Sokić, dipl.el.inž.
Odsjek za automatiku i elektroniku
Sarajevo, 05.11.2009.

Tema za završni rad
studenta I ciklusa studija koji studira na ETF-u u skladu sa principima Bolonjskog procesa
na Odsjeku za automatiku i elektroniku (šk.2009/10)

Tema: Izrada Praktikuma za laboratorijske vježbe iz “Signala i sistema”

Student: Bukva Emina

Sažetak:
U okviru ovog rada potrebno je dizajnirati Praktikum za laboratorijske vježbe iz ”Signala i sistema”. Praktikum treba da obradi sve laboratorijske vježbe koje se već drže na predmetima ”Teorija signala” i ”Analiza signala i sistema”, koje prate gradivo koje se obrađuje u sklopu predmeta. Naglasak Praktikuma treba da bude na upotrebi i primjeni MATLAB-a kao alata za upoznavanje sa osnovnim konceptima teorije obrade i analize signala.

On treba da sadrži sljedeće elemente:

1. Upoznavanje sa radnim okruženjem MATLAB-a
2. Rad sa matricama u MATLAB-u
3. Crtanje analognih i diskretnih signala u MATLAB-u
4. Rad sa m-fajlovima i funkcijama u MATLAB-u
5. Korištenje Simulinka
6. Osnovni elementi obrade zvuka u MATLAB-u – kao diskretnog signala
7. Osnovni elementi obrade slike u MATLAB-u – kao višedimenzionalnog signala
8. Crtanje signala i aproksimacionih funkcija – razvoj u Fourierov Red
9. Fourierova, Hilbertova i Z-Transformacija u MATLAB-u
10. DFT i FFT u MATLAB-u – ilustracija na primjerima.
11. Ilustracija Nyquistove teoreme odabiranja na primjerima slike, zvuka i realnog fizičkog signala
12. Osnove akvizicije signala sa MATLAB-om – obrada signala u realnom vremenu

Svako poglavlje treba se sastojati iz 3 dijela:

1.Kratak teorijski uvod (Pregled osnovnih MATLAB funkcija)
2.Urađeni primjeri
3.Zadaci sa samostalan rad

Koncept i metode rješavanja:

Za izradu softverskog dijela rada i računarsku analizu obavezna je upotreba MATLAB softverskog paketa.

Literatura:

1. **Melita Ahić-Đokić**, *"Signali i sistemi"*, Elektrotehnički fakultet Sarajevo, 2010.
2. **Alan V. Oppenheim, Alan S. Willsky**, *“Signals and systems”*, Prentice Hall, 1997.
3. **Alan V. Oppenheim, Roland W. Schafer**, *“Discrete-time Signal and Processing”*, Prentice Hall, 1999.
4. **The MathWorks, Inc**., “*Getting Started with MATLAB*”, http://www.mathworks.com/help/ pdf_doc/matlab/getstart.pdf , 11.12.2009. godine
5. **Nasser Kehtarnavaz, Philipos Loizou, Mohammad Rahman**, *"An Interactive Approach to Signals and Systems Laboratory,"* Connexions, http://cnx.org/content/col10667/1.10/, 12. 12. 2010. godine.
6. **Adnan Tahirović, Mujo Hebibović,** *“MATLAB u teoriji automatskog upravljanja – praktikum za laboratorijske vježbe ”,* Elektrotehnički fakultet Sarajevo, 2002.

Sarajevo, septembar 2010. Mentor:

Red. prof. dr Melita Ahić-Đokić, dipl.el.inž.

1. Uvod u razvojno okruženje MATLAB-a i Simulink-a

MATLAB

MATLAB predstavlja veoma razvijen jezik tehnološkog proračuna koji objedinjuje proračun, vizualizaciju i programiranje u jednostavnom razvojnom okruženju u kome se problem i rješenje izražavaju u poznatoj matematičkoj notaciji.

MATLAB predstavlja interaktivni sistem čiji je osnovni tip podataka matrica koji ne zahtijeva prethodno dimenzioniranje, te je vrlo pogodan za rješavanje tehničkih problema koji zahtijevaju rad sa matricama. Interakcija s korisnikom se odvija direktnim unošenjem komandi na promptu u komandnom prozoru. Pored toga omogućeno je i pisanje programa u vidu m-fajlova koje MATLAB potom interpretira i izvršava. MATLAB sadrži i kolekcije već predefiniranih funkcija koje su grupisane u alatne grupe (*eng. Toolbox*) namijenjenih za rješavanje problema iz neke naučne oblasti npr. Optimization Toolbox, Neural Network Toolbox, Fuzzy Logic Toolbox, Control System Toolbox, Signal Processing Toolbox, Statistics Toolbox itd. Postoji još mnogo primjena ovakvih alata, ali mi ćemo se fokusurati na akviziciju, analizu i vizualizaciju podataka.

Pokretanje i zaustavljanje MATLAB-a

Pokretanje MATLAB-a se vrši odabirom MATLAB-a iz Programs menija ili dvostrukim klikom na ikonu na desktopu. Kada se se otvori grafički interfejs MATLAB-a sačekamo da se pojavi znak ‘`>>`’ koji označava da je MATLAB spreman za unos podataka.

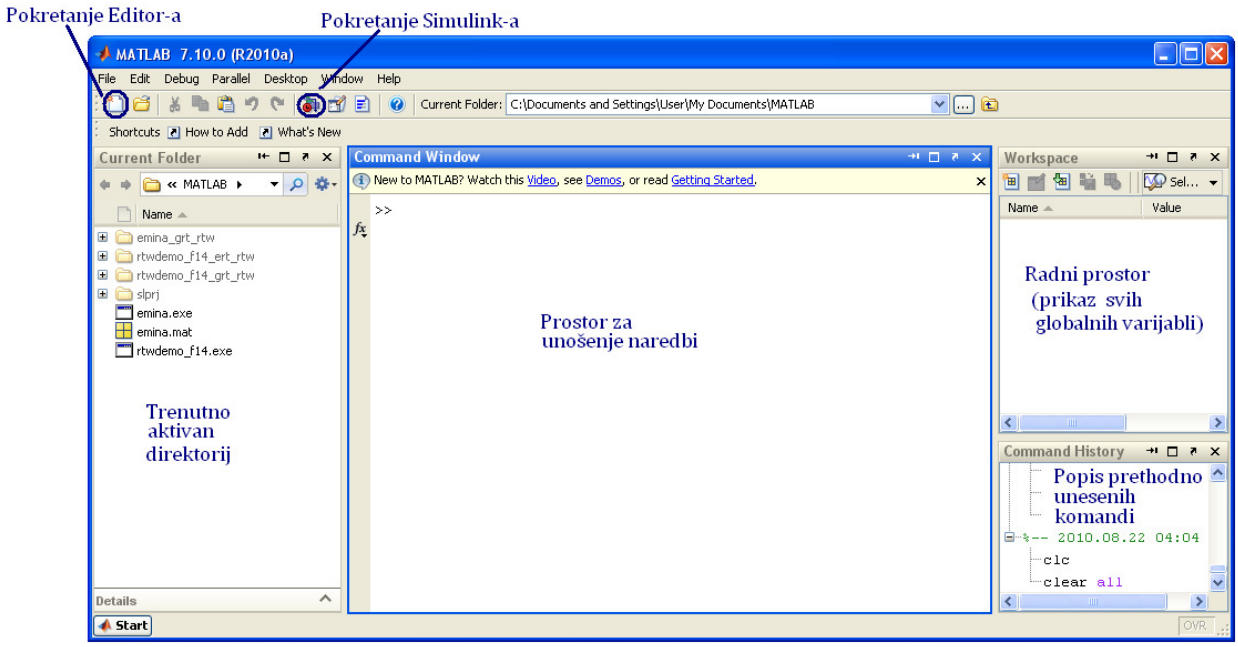
Grafički interfejs MATLAB-a pored standardizovanih menija sačinjavaju i prozor za unošenje komandi, prozor koji vodi evidenciju o svim ranije unesenim komandama, prozor za manipulaciju folderima i radni prostor u kome su prikazane sve globalne varijable koju su trenutno dostupne unutar MATLAB-a. Izgled grafičkog interfejsa MATLAB-a je prikazan na slici 1.1.

MATLAB korisniku nudi veoma kvalitetan sistem pomoći koji se aktivira unosom komadne riječi '`help`' u komandni prozor. MATLAB posjeduje i veliku kolekciju demonstracija korištenja pojedinih alatnih skupina, kojima je moguće pristupiti korištenjem naredbe ‘`demo`’. Ukoliko znamo ime funkcije, ali ne znamo kako da je upotrijebimo, koje parametre prima i slično, moguće je jednostavno unijeti sljedeći zahtjev za pomoć u komandni prozor:

```
>> help ime_funkcije
```

Da bi napustili MATLAB potrebno je unijeti ‘`quit`’ ili ‘`exit`’. Ukoliko želite obustaviti rad prije završetka i sačuvati podatke ukucajte ‘`save`’ prije napuštanja programa i vaš radni postor će biti sačuvan u tekućem direktoriju (naziv fajla je po defaultu *matlab.mat*). Tekući direktorij predstavlja direktorij unutar kojeg će MATLAB spašavati sve podatke, dnevnike, m-fajlove, slike, modele i sl. ukoliko eksplicitno ne naznačite drugačije. Također, unutar tekućeg direktorija je moguće pozivati sve sačuvane fajlove unošenjem samo njihovog imena,

bez navođenja pune memorijske adrese. Ukoliko želite nastaviti raditi na svom projektu nakon pokretanja MATLAB-a ukucajte ‘`load`’ da povratite vaš prijašnji radni prostor i nastavite tamo gdje ste se prethodno zaustavili.



Slika 1.1 MATLAB grafički interfejs

Varijable u MATLAB-u

Matlab se može upotrijebiti kao običan linijski kalkulator. Ako u komandnom prozoru MATLAB-a ukucate:

```
>> 1.2*sin(3.4^2+log10(5))
```

i pritisnete tipku ‘*Enter*’ dobili biste vrijednost izraza $1.2 \times \sin(3.4^2 + \log_{10} 5)$[1]. Kako nije specificirana varijabla kojoj će biti dodijeljen dobijeni rezultat, rezultat operacije se po default-u privremeno pohranjuje u varijablu ‘`ans`’. Da bi ovu vrijednost sačuvali za kasniju upotrebu, potrebno ju je imenovati npr.

```
>> rezultat=1.2*sin(3.4^2+log10(5))
```

Sada smo kreirali varijablu ‘`rezultat`’ koju vidimo u našem radnom prostoru. Primjetimo da varijablu ‘`rezultat`’ nismo morali porethodno kreirati, dimenzionirati niti navesti tip varijable, već smo joj samo pridružili vrijednost. Znak za pridruživanje je ‘=’ , a ime varijable može biti bilo koja riječ koja počinje slovom i ne sadrži razmake niti specijalne znakove, pri čemu se mala i velika slova razlikuju.

Kreiranu varijablu možemo koristiti za nove operacije:

```
>> novi_rezultat=3+rezultat/2
```
[2]

[1] *Logaritam po bazi e se dobije korištenjem funkcije 'log(x)', dok se logaritam po bazi 10 dobija korištenjem funkcije 'log10(x)'. Decimalna tačka je predstavljena znakom '.'.*

[2] *Operacije računanja se vrše prema prioritetu. Ukoliko se želi naglasiti drukačiji redoslijed izvršenja potrebno je to naglasiti upotrebom malih zagrada.*

Matlab poznaje i rad sa kompleksnim brojevima. Da bi dobili $\sqrt{-1}$ možete ukucati ‘`sqrt(-1)`’ ili jednostavnije ‘`i`’ (podržana je i notacija ‘`j`’). Kompleksne brojeve je moguće unijeti u algebarskom obliku preko njihovog realnog i imaginarnog dijela $z = x + iy$ ili u Eulerovom obliku, preko modula i argumenta kompleksnog broja $z = re^{i\theta}$.

```
>> z=3+4i
```

ili

```
>> z=5*exp(i*0.9273)
```

U oba slučaja MATLAB kao potvrdu ispisuje kompleksni broj z u obliku $x + yi$. Kompleksni broj nije moguće zadati kao ‘`z=3+i4`’, jer MATLAB pretpostavlja da je ‘`i4`’ naziv varijable ili funkcije (`??? Undefined function or variable 'i4'.`).

Da bi upotrijebili ovakav redoslijed zapisivanja potrebno je naglasiti da se radi o množenju ‘`z=3+i*4`’. Primjetimo takođe da se argument θ zadaje u radijanima, a ne u stepenima. Korištenjem sljedećih funkcija možete dobiti vrijednost realnog dijela ‘`real(z)`’, imaginarnog dijela ‘`imag(z)`’, modula kompleksnog broja ‘`abs(z)`’ i argumenta kompleksnog broja ‘`angle(z)`’.

Predefinirana vrijednost ‘`i`’ je $\sqrt{-1}$ no moguće je (iako veoma nepreporučljivo) promijeniti ovu vrijednost jednostavnim dodjeljivanjem neke druge vrijednosti varijabli i, npr `i=10`. Pored pomenutih ‘`ans`’ i ‘`i`’(ili ‘`j`’) u preddefiniranie varijable MATLAB-a spadaju i:

- `pi` $= 3.14$
- `eps` $=$ najmanji pozitivan broj (preciznost floating point formata 2^{-52})
- `realmax` $=$ najveći broj u floating point formatu (2^{1023})
- `realmin` $=$ najmanji broj u floating point formatu (2^{-1022})
- `Inf` $=$ broj veći od realmax
- `NaN` $=$ nije broj npr. $0/0$, Inf/Inf ili drugi izrazi koji nemaju definirano matematičko značenje

Veoma korisna osobina MATLAB-a je je što funkcije poput ‘`sin`’, ’`log`’, ‘`abs`’ i mnogih drugih mogu da se računaju za više različitih argumenata istovremeno. Pretpostavimo npr. da se žele izračunati vrijednosti $\sin(\frac{-\pi}{2})$, $\sin 0$, $\sin\frac{\pi}{2}$, $\sin\pi$. Ove vrijednosti možemo naći unoseći sljedeću komandu u radni prostor MATLAB-a:

```
>> sin([-pi/2,0,pi/2,pi])
```

Uočavamo da se argument funkcije piše unutar malih zagrada, a ukoliko imamo listu argumenata onda nju navodimo u uglastim zagradama. S obzirom da su elementi razdvojeni zarezom lista argumenata je horizontalna (predstavlja jedan red), pa je i rezultat horizontalan. Listu argumenta možemo definisati i izvan funkcije. Ukoliko elemente razdvojimo tačka-zarezom lista, i shodno tome rezultat, će biti vertikalni:

```
>> vlista=[-pi/2; 0; pi/2; pi]
>> sin(vlista)
```

Vektori i matrice

Elementi vektora, u opštem slučaju matrice se navode nabrajanjem unutar uglastih zagrada `[]`. Pri tome koristimo ‘`,`’ ili ‘`Space`’ za navođenje elemenata u jednom redu dok ‘`;`’ ili ‘`Enter`’ služi za razdvanje elemenata po redovima.

Matricu $A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & -2 & -3 \end{bmatrix}$ možete unijeti u MATLAB koristeći neku od navedenih naredbi:

```
>> A=[1,2,3;-1,-2,-3]
```

ili

```
>> A=[1 2 3
-1 -2 -3]
```

Ukoliko želimo izračunati :

$$\begin{bmatrix} \ln 1 & \ln 2 & \ln 3 \\ \ln(-1) & \ln(-2) & \ln(-3) \end{bmatrix}$$

To možemo uraditi koristeći sljedeći kod:

```
>> log(A)
```

Specijalno, matricu koja sadrži samo jedan red nazivamo **vektor red** ili **vektor vrsta**, a matricu koja sadrži samo jednu kolonu *vektor kolonom*. Dobro poznavanje matričnog računa može značajno pojednostaviti rad u Matlabu, ali ne predstavlja nužan preduslov.

Ukoliko u komandni prozor ukucamo:

```
>> B=[A ; 2*A]
```

rezultujća matrica će imati 4 reda i 3 kolone. Da bi vidjeli koji se elemenat nalazi u presjeku trećeg reda i druge kolone matrice ‘`B`‘ ukucamo:

```
>> B(3,2)
```

U opštem slučaju, element u presjeku i-tog reda i j-te kolone matrice ‘`B`’, adresiramo sa ‘`B(i,j)`’. Želimo li adresirati i više od jednog elementa, to možemo učinti pobrojavanjem odgovarujućih redova odnosno kolona. Ukoliko sve elemente prve i treće kolone želimo prepsati u novu matricu ‘`C`’ to možem postići sljedećim kodom:

```
>> C=B(:,[1,3])
```

ili

```
>> C=B(1:end,[1,3]);
```

Matrica ‘`C`’ ima 4 reda i 2 kolone. Znak ‘`:`’ je korišten da označi sve kolone, tako da nismo morali pisati ‘`C=B([1,2,3,4],[1,3])`’. Umjesto nabrajanja operator ‘`:`’ možemo koristiti da zadamo opseg, npr. ‘`1:3:10`’ daje kao rezultat se cijele brojeve između 1 i 10 sa korakom 3 (1,4,7). Korak može biti i negativan. Dozvoljeno je da korak bude i racionalan broj, npr. ‘`0:0.1:1`’, ali pri indeksiranju elemenata matrica takav korak nije smislen, pa nije ni dozvoljen. Zadnji red ili kolonu u matrici možemo indeksirati sa ‘*`end`*’.

Između dvije matrice istih dimenzija defisani su sljedeći poredbeni operatori koji vrše poređenje matrica element po element. Rezultat ovih operacija je nova matrica istih dimenzija popunjena nulama i jedinicama. Ukoliko je elemenat

rezultujuće matrice jednak nuli to označava da elementi matrica na toj poziciji nisu u relaciji, a ukoliko elementi zadovoljavaju zadanu relaciju onda se u rezultujućoj matrici na njihovoj poziciji nalazi jedinica. Relacioni operatori su:

`<`	manje	`<=`	manje ili jednako
`>`	veće	`>=`	veće ili jednako
`==`	jednako	`~=`	različito

Neke od osnovnih funkcija za rad sa matricama koristimo za zadavanje često korištenih oblika matrica, npr. nul matricu dimenzija mxn možemo kreirati pozivanjem naredbe ‘`zeros(m,n)`’, jediničnu matricu dimenzija nxn kreiramo naredbom ‘`eye(n)`’, a matricu popunjenu jedinicama dimenzija mxn pomoću naredbe ‘`ones(mxn)`’.

Nad matricama je moguće vršiti elementarne artimetičke operacije upotrebom operatora ‘`+`’, ’`-`‘. Pri tome treba voditi računa da matrice moraju biti istih dimenzija. Množenje matrica u matričnom smislu se obavlja operatorom ‘`*`’ i obavlja se pod uslovom da je broj kolona prve matrice jednak broju redova druge. Korištenjem operatora ‘`.*`’ moguće je izvesti skalarno množenje matrica tako da svaki element prve matrice bude pomnožen elementom druge matrice na istoj poziciji. Uslov je da matrice moraju biti jednake. Analogno prethodnom postoje i dvije vrste stepenovanja matrica, stepenovanje kvadratne matrice operatorom ’`^`’ i skalarno stepenovanje elemenata matrice operatorom ‘`.^`’ koje ne podrazumijeva ograničenja na dimenzije.

Dijeljenje matrica A i B zapravo predstavlja matrično množenje matrice A sa inverznom matricom matrice B. Inverznu matricu je moguće dobiti direktno korištenjem naredbe ‘`inv(B)`’. Operator ‘`/`’ predstavlja matrično dijeljenje s desna i izraz ‘`C=A/B`’ je jednak ‘`C=A*inv(B)`’, dok operator ‘`\`’ predstavlja matrično dijeljenje s lijeva pri čemu je ‘`C=A\B`’ jednako je ‘`C=inv(B)*A`’. Dijeljenje odgovarajućih elemenata dvije matrice istih dimenzija je omogućeno upotrebom operatora ‘`./`’. Potrebno je navesti i operator ‘`'`’ koji vrši transponovanje matrice A.

Korisniku su na raspolaganju različite funkcije za manipulaciju nad pohranjenim podacima u matrici, kao što su:

`det(A)` – vraća determinantu matrice A

`sum(A)` – vraća vektor-red čiji su elementi sume kolona matrice A

`diag(A)` – vraća vektor-kolonu koja sadrži elemente glavne dijagonale matrice

`poly(A)` – vraća vektor-kolonu koja sadrži koeficijente karakterističnog polinoma matrice A, tj. *det(A-ΛE)*

`size(A)` – vraća vektor čiji su elementi dimenzije matrice A

`size(A,1), size(A,2)` – vraća broj redova, odnosno kolona matrice A

`finite(A)` – vraća matricu koja sadrži samo konačne elemente matrice A

`fliplr(A), flipud(A)` – obrće matricu A sa lijeva na desno, odnosno odozgo na dole

`rot90(A)` – rotira matricu za 90˚ u smjeru suprotnom od kazaljke na satu

Ukoliko želimo naći jedinične vektore i jedinične vrijednosti matrice A ukucamo:

```
>> [W D]=eig(A)
```

Varijabla ‘W’ je matrica čije kolone predstavljaju jedinične vektore, a matrica ‘D’ je dijagonalna matrica čiji elementi glavne dijagonale predstavljaju sopstvene vrijednosti matrice ‘A’. ž

Polinomi u MATLAB-u se mogu također opisivati matricama, tačnije vektorima. Polinom $p(x) = 2x^2 - 3x + 1$ može biti zapisan kao:

```
>> p=[2 -3 1]
```

Omogućeno je vršenje različitih operacija nad polinomima, npr. za nalaženje korijena polinoma p možemo upotrijebiti funkciju ‘roots(p)’.

U MATLAB-u često radimo sa matricama velikih dimenzija, pa bi ispisivanje rezultata nakon svakog koraka bilo nepraktično. Jednostavan način da spriječimo MATLAB da ispisuje rezultat obavljene operacije je da stavimo ‘;’ na kraju komande. Kreirajmo vektor od 100 tačaka ravnomjerno raspodjeljenih između 0 i π (ne zaboravite ‘;’):

```
>> x=linspace(0,pi,100);
```

Sada možemo ispisati svaku desetu vrijednost, samo ukucamo ‘x(1:10:end)’ i izračunati sinus ovih vrijednosti komandom ‘sin(x)’.

```
>> sinx=sin(x);
```

Ukoliko za ispis podataka umjesto standardne MATLAB-ove notacije želimo koristiti notaciju u obliku “mantisa×exponent”. Posmatrajmo vrijednosti sinusa oko 1/2, npr. između 49-og i 52-og elementa, da biste promjenili notaciju unesite sljedeći kod:

```
>> sinx(49:52)
>> format short e
>> sinx(49:52)
```

Sada možemo uočiti vrijednosti prve 4 decimale i eksponente od 10 (oznaka e za eksponent), a ako želimo prikaz sa punom tačnošću izraza:

```
>> format long e
>> sinx(49:52)
```

Ukoliko se želite vratiti na standardnu notaciju u radni prostor MATLAB-a unesite komandu ‘format’:

```
>> format
```

Grafički prikaz

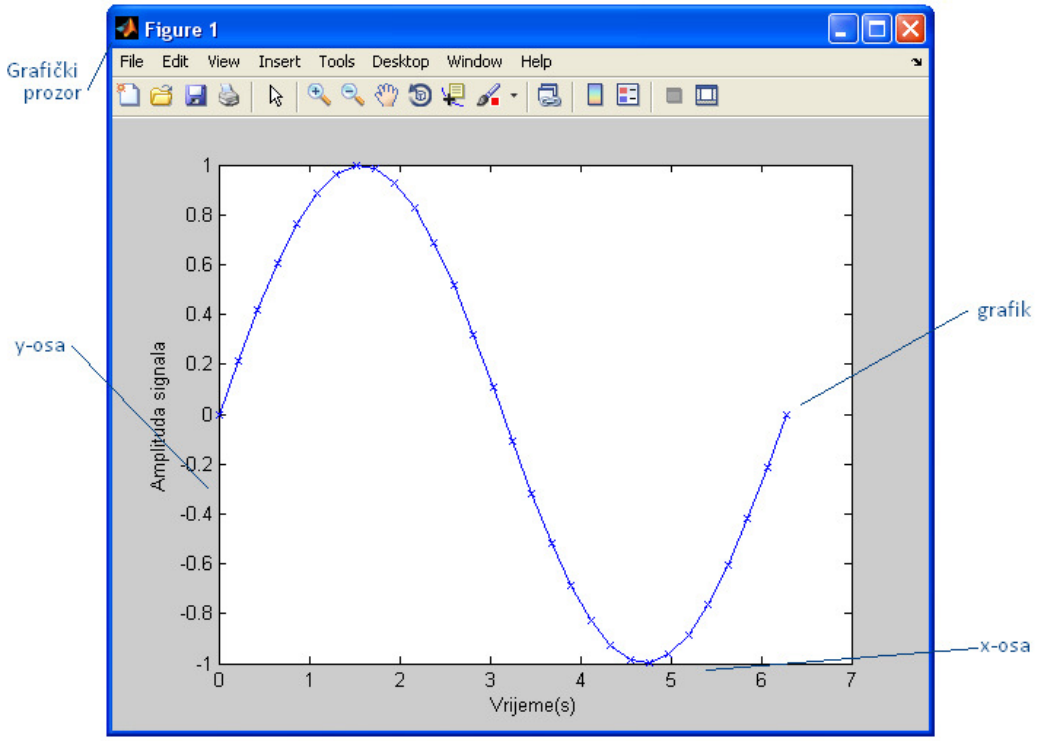
Dobijanje grafičkog prikaza podataka u MATLAB-u je veoma jednostavno. Osnovna sintaksa je ‘plot(x,y)’ koja otvara grafički prozor u kojem crta vrijednosti vektora podataka y (ordinata) u odnosu na vrijednosti vektora podataka x (apcisa). Vektori x i y moraju imati iste dimenzije. Grafik se prikazuje u zasebno otvorenom prozoru. Ukoliko želimo otvoriti novi grafički prozor to možemo učiniti unosom naredbe ‘figure’. Sve naredne komande za obradu grafika koje unesemo u MATLAB će se odnose na posljednji otvoreni grafički prozor. Neke od osnovnih komandi za rukovanje grafičkim prikazom su:

`xlabel('oznaka x ose')`	– oznaka x ose
`ylabel('oznaka y ose')`	– oznaka y ose

`title('tekst naslova')`	– naslov grafikona
`grid on/off`	– uključuje/isključuje prikaz koordinatna mreže
`hold on/off`	– uključuje/isključuje zadržavanje prikaza
`axis on/off/auto/square/equal`	– rukovanje prikazom osa
`axis([xmin xmax ymin ymax zmin zmax])`	– definira opsege osa
`text(x,y,'tekst')`	– prikazuje tekst na poziciji x, y
`legend('1. grafik','2. grafik')`	– ispisivanje legende na grafiku
`subplot(m,n,k)`	– crta k-ti grafik u grafičkom prozoru koji sadrži mxn grafika podijeljenih u m redova i n kolona
`close(n)/close all`	- zatvori n-ti grafički prozor/ zatvori sve

Da bi grafički prikazali ranije kreiranu sinusnu funkciju sinus ukucamo u komandni prozor sljedeće naredbe:

```
>> x=linspace(0,2*pi,30);        %definišemo vrijednosti vektora x
>> sinx=sin(x);                  %definišemo vrijednosti vektora y
>> plot(x,sinx,'b-x');           %crtanje grafika
>> xlabel('Vrijeme(s)');         %oznaka x ose
>> ylabel('Amplituda signala');  %oznaka y ose
```



Slika 1.2 Primjer crtanja grafika funkcije sinx=sin(x)

Kao posljednji argument funkciji ‘`plot`’ je proslijeđen niz znakova ‘`b-x`’ koji redom označavaju boju grafika (`b` plava boja), vrstu linije(`-` puna linija), te tip markera (`x` križić) koji prikazuje podatke. Pregled mogućnosti vizuelnog prikaza 2D grafikona je moguće dobiti unosom naredbe ‘`help plot`’ u komandni prozor.

Jednostavne naredbe za crtanje 2D grafika pored ‘plot’ uključuju:

`stem` – crtannje diskretnih uzoraka funkcije u linearnim koordinatama
`loglog` – cratnje grafika uz logaritamske x i y ose
`semilogx` – cratnje grafika uz linearnu y logaritamsku x osu
`semilogy` – cratnje grafika uz linearnu x logaritamsku y osu
`hist` – crtanje grafika u formi histograma
`polar` – crtanje grafika u polarnim koordinatama

Još neke važne funkcije za crtanje 2D grafika se mogu naći sa ‘`help graph2d`’.

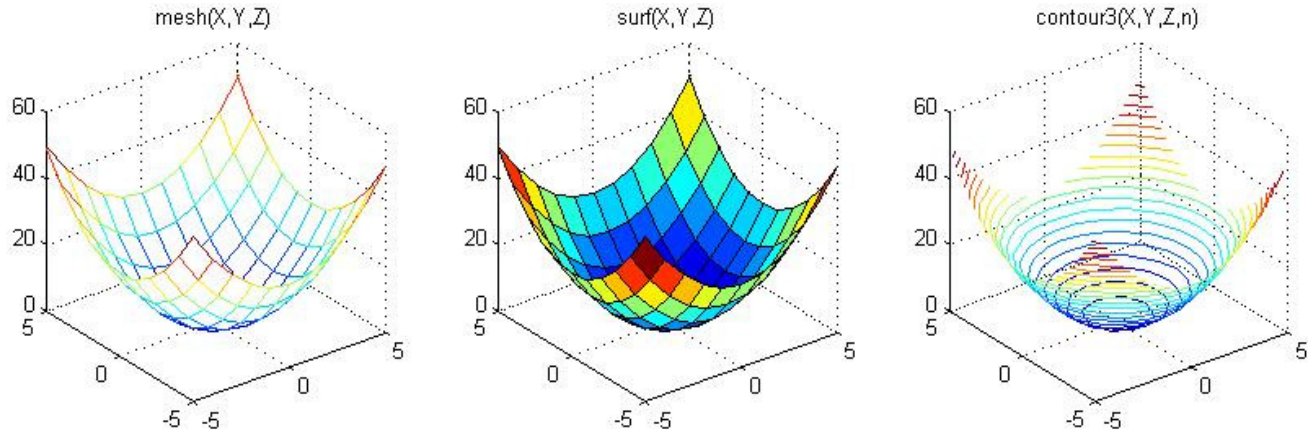
Za crtanje 3D grafika `z=z(x,y)` potrebno je prilagoditi ulazni set podataka sadržanih u vektorima ‘`x`’ i ‘`y`’ tako da kreiraju mrežu tačaka u xy ravni za koje su vrijednosti funkcije z izračunate i grafički prikazane. Generisanje ove mreže tačaka za zadane vrijednosti ose x i ose y za koje će funkcija biti prikazana se vrši naredbom ‘`meshgrid`’:

```
>> x=-5: 1:5;
>> y=-5:1:5;
>>[X,Y]=meshgrid(x,y);
```

Sada se umjesto funkcije ‘`z=z(x,y)`’ računa funkcija ‘`Z=Z(X,Y)`’, tako da vrijednosti funkcije $z = x^2 + y^2$ možemo izračunati i grafički prikazati koristeći sljedeću naredbu:

```
>> Z=X.^2+Y.^2;
```

Pri tome je potrebno koristiti skalarne operatore, a ne matrične. Grafički prikaz 3D grafika za ovu funkciju je moguće postići koristeći niz funkcija koje se mogu naći sa sa ‘`help graph3d`’. Neke od mogućnosti grafičkog prikaza su predstavljene na slici ispod.



Slika 1.3 3D grafikoni funkcije Z(X,Y)=X.^2+Y.^2

Kreiranje m-fajlova i pisanje vlastitih funkcija u MATLAB-u

Unošenjem komande ‘`edit`’ u MATLAB-u se otvara editor sa praznim m-fajlom. Unutar ovog m-fajla moguće je upisati niz naredbi koje želimo izvršiti. Nakon upisivanja komandi m-fajl je potrebno imenovati i sačuvati. Uz ime fajla obavezno ide ekstenzija .m. Kasnije kada želite izvršiti komande upisane u m-fajl, dovoljno je samo unijeti ime fajla u editor (bez ekstenzije.m) i on će sa podacima pohranjenim u globalne varijable sadržane u radnom prostoru izvršiti komande upisane u m-fajl. Ovaj način pristupanja je moguć samo ukoliko je m-fajl sačuvan u trenutno aktivnom direktoriju, u suprotnom je potrebno promijeniti trenutno aktivni direktorij u direktorij koji sadrži m-fajl ili upisati potpunu adresu m-fajla. Drugi način pokretanja m-fajla je direktno iz editora pritiskom na tipku `F5`. M-fajlovi koji samo izvršavaju komande koje su u njima sadržane bez prihvatanja i vraćanja podataka se nazivaju skripte.

Prilikom pisanja m-fajla većeg obima od velikog je praktičnog značaja pisati komentare radi pojednostavljenja izvođenja budućih korekcija u skripti. Komentar započinjemo znakom ’`%`’. MATLAB tumači kao komantar svaki karakter koji se nalazi desno od ovog znaka pa sve do kraja reda.

Funkcije predstavljaju m-fajlove koji imaju posebno zaglavlje u kome se navodi ime funkcije, ulazni argumenti i vrijednosti koje funkcija vraća. Pri tome bi ime funkcije trebalo biti isto kao i ime m-fajla. Izgled zaglavlja funkcije:

```
function izl_var=ime_funkcije(ul_var1,ul_var2,...)
```

gdje su:

`izl_var`	– varijabla koju funkcija vraća (ili više njih-matrica)
`ime_funkcije`	– ime funkcije koje odgovara imenu m-fajla
`ul_var1, ul_var2, ...`	– argumenti koji se prenose u funkciju pri pozivu.

Broj argumenata koji se prosljeđuju funkciji može biti manji od zadanog ukoliko su za neproslijeđene argumente unutar funkcije određene default-ne vrijednosti. Ukoliko se neki od argumenata ne koristi to označavamo pisanjem ‘`[]`’ na mjestu njegove vrijednosti. Korištenjem naredbe ‘`nargin`’ je moguće provjeriti koliko je argumenata proslijeđeno funkciji.

Varijable unutar funkcije imaju lokalni karakter, tako da njihove izmjene unutar funkcije ne utiču na njihove vrijednosti u radnom prostoru MATLAB-a. Ukoliko želimo da ove izmjene budu vidljive i izvan funkcije potrebno je varijable prije prve upotrebe deklarisati kao globalne:

```
global ime_varijable
```

Ukoliko funkcija vrati veći broj vrijednosti od predviđenog MATLAB neće preuzeti sve vrijednosti, već samo njihov ranije određen broj i pri tome preuzima vraćene vrijednosti redom s lijeva na desno. Korištenjem naredbe ‘`nargout`’ je moguće uticati na to koje će vrijednosti funkcija vratiti.

Kontrola toka izvršavanja

MATLAB posjeduje pet struktura za kontrolu toka. Sintaksa struktura za kontolu toka je veoma slična onoj u C programskom jeziku odakle je i preuzeta.

if struktura

```
if  uslov1 kod1
elseif  uslov2 kod2
else kod3
end
```

Ukoliko su varijable koje se porede skalarne veličine kao uslove je moguće koristiti ranije navedene relacione i logičke operatore. Međutim u slučaju da su veličine koje se porede matrice, potrebno je umjesto relacionih operatora koristiti logičke funkcije, jer relacioni operatori ne vraćaju logičku vrijednost već matricu vrijednosti 0 i 1. Često korištene logičke funkcije uključuju:

`isequal(A,B)` - rezultat je 1 ili 0 ovisno da li su svi elementi A jednaki elementima B ili nisu

`isempty(A)` - rezultat je 1 ako je matrica A prazna

`all(A)` - rezultat je 1 ako su svi elementi matrice A nenulti

`any(A)` - rezultat je 1 ako je bar jedan element matrice A različit od nule

switch struktura

```
switch varijabla
    case vrijednost1
         kod1
    case vrijednost2
         kod2
    ...
    otherwise
         kod
end
```

Kod MATLAB-ove ‘`switch`’ strukture program se završava čim je jedan od uslova zadovoljen, što je značajna razlika u odnosu na ‘`switch`’ strukturu programskog jezika C.

for petlja

```
for n=pocetna_vrijednost:korak:krajnja_vrijednost
    kod
end
```

Omogućava izvršavanje dijela koda tačno određeni broj puta.

while petlja

```
while uslov
    kod
end
```

Omogućava izvršavanje dijela koda sve dok je ispunjen navedeni uslov.

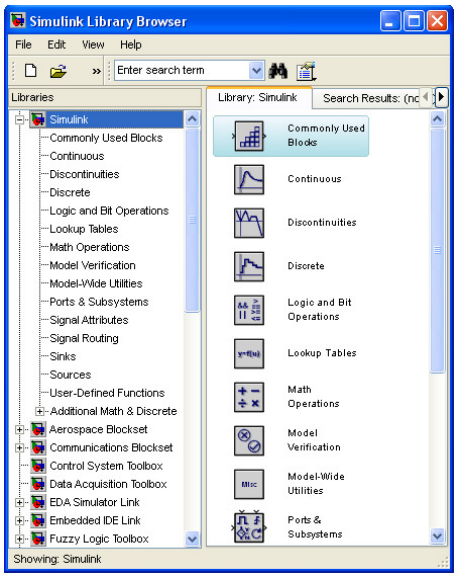
break

Komanda koja prekida izvršavanje neke od ovih struktura. Ukoliko je više ovakvih struktura ugnježdeno prekida se samo unutarnja struktura koja sadrži ‘`break`’.

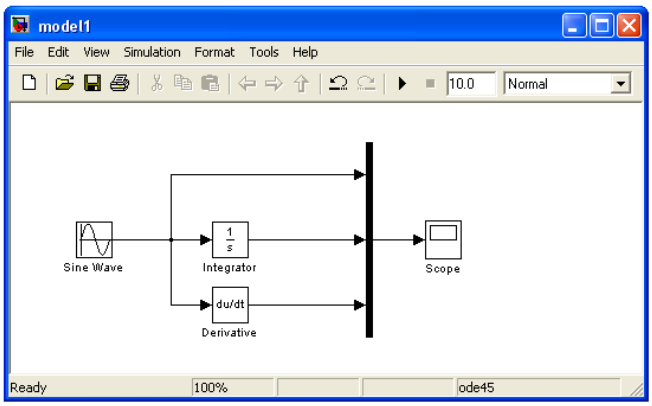
Simulink

Simulink u suštini predstavlja grafički korisnički interfejs (GUI) koji kao dodatak MATLAB-u pomaže lakšem i jednostavnijem kreiranju i dizajnu složenijih sistema i vizualizacije rezultata. On omogućava korisniku da jednostavnom ‘drag and drop’ metodom odabire potrebne blokove i povezuje ih linijama toka signala.

Da bi pokrenuli Simulink jednostavno ukucamo komandu ‘simulink’ u MATLAB-ovom prozoru za unošenje komandi ili dvostrukim klikom miša odaberemo ikonicu sa znakom Simulink na izbornoj traci. Nakon toga otvara se Simulink-ov pretraživač (Library Browser). Za početak je potrebno odabrati ikonicu sa praznim bijelim listom u gornjem lijevom uglu koja otvara novi model.



Slika 1.5 Simulink-ov pretraživač biblioteka



Slika 1.4 Jednostavan Simulink model sa koji vrši integriranje i diferenciranje ulaznog sinusoidalnog signala

Da bismo napravili jednostavan Simulnik model prikazan na slici 1.5 potrebno je prikazane blokove pronaći u Simulink-ovom pretraživaču. U sekciji *Sources* ćete pronaći blok *Sine Wave* koji predstavlja izvor sinusoidalnog signala, koji trebate označiti i potom prevući u vaš prazan model. Blokovi *Integrator* i *Derivative* se nalaze u *Continuous* sekciji, a blok *Mux* u *Signal Routing* sekciji. *Scope* ćete pronaći u sekciji *Sinks*. Blokovi se povezuju tako što klikom miša odaberemo izlazni port bloka i vučemo vezu do ulaznog porta željenog bloka. Veze između blokova je potrebno urediti tako da odgovaraju onima prikazanim na slici 1.5. Nakon što ste prenijeli blokove u model možete ga sačuvati. MATLAB uz odabrano ime fajla dodaje ekstenziju *.mdl*. Pozicionirajte miša na dugme *Play* i pritiskom lijeve tipke miša pokrenite simulaciju. Dvostrukim klikom miša na Scope dobijate grafički prikaz ulaznog i izlaznih signala.

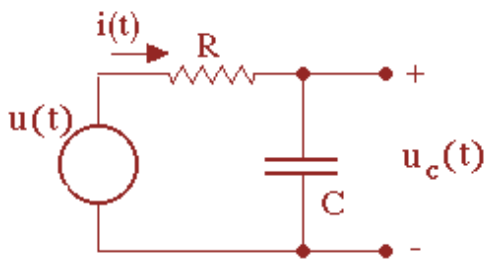
Dvostrukim klikom na bilo koji blok otvara se novi prozor *Parametars* unutar kojeg je moguće vršiti podešenja vrijednosti parametara tog bloka. Tako je izvoru sinusoidalnog signala između ostalog moguće podešavati amplitudu, frekvenciju, fazni pomak sigala. U donjem desnom uglu se nalazi tipka Help. Pritiskom na ovu tipku otvara se sistem pomoći i prikazuje informacije o mogućnostima podešenja izabranog bloka.

Simulacija sistema prvog reda u kontinualnom vremenu

Kao primjer sistema prvog reda razmotrimo jednostavno RC kolo prikazano na slici 1.6. Diferencijalna jednačina koja opisuje ovo kolo je:

(1.1)

gdje je kapacitet kondenzatora , a otpornost otpornika prikazanog na slici.

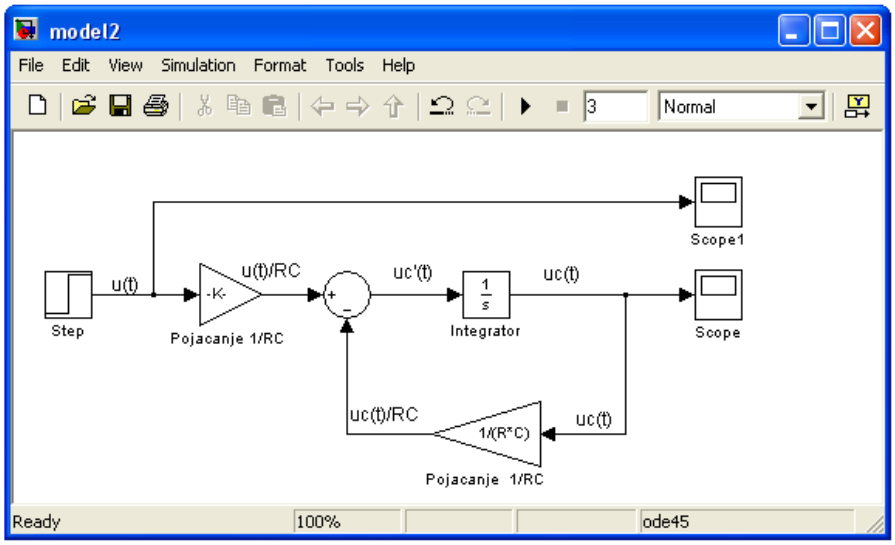


Slika 1.6 Serijsko RC kolo

Da bi kreirali simulink model koje će simulirati ponašanje gore opisanog sistema potrebno je iz diferencijalne jednačine koja ga opisuje odrediti najviši izvod izlazne veličine koji se pojavljuje u jednačini. Jednačinu (1.1) je moguće zapisati kao:

— — (1.2)

Ukoliko pretpostavimo da je na ulaz sistema dovedena jedinična step funkcija koju možemo predstaviti *Step* blokom iz *Source* sekcije, za izgradnju modela će nam još biti potreban blok sumatora (*Sum*), blok integratora i dva bloka pojačanja (*Gain*). Kreirani model treba da izgleda kao model na slici 1.7.



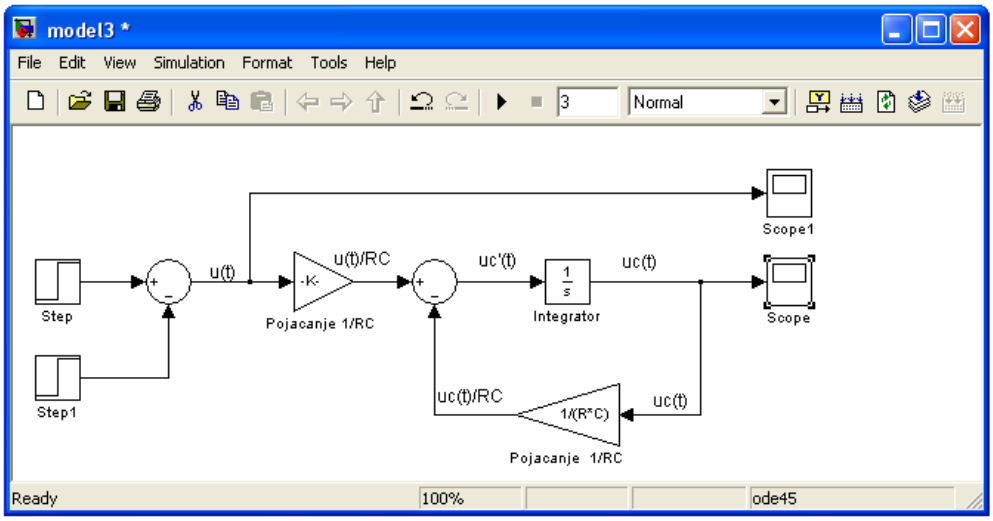
Slika 1.7 Simulink model sistema prvog reda sa odzivom na jediničnu step funkciju

Podešavanje parametara pojačanja možemo izvršiti tako što ćemo izračunati iznos konstante pojačanja i direktno je unijeti u blok pojačanja. Alternativno tome pojačanje je moguće zadati preko parametara R i C kao 1/(R*C) pri čemu Simulink za uspješno pokretanja modela zahtijeva da su unutar MATLAB-ovog radnog prostora

unešene dvije skalrne varijable ‘`R`’ i ‘`C`’ sa definisanim numeričkim vrijednostima. Ukoliko dimenzije bloka pojačanje ne omogućavaju prikaz koeficijenta pojačanja, on će biti simbolično predstavljen oznakom ‘`-K-`’.

Prije pokretanja modela potrebno je podesiti vrijeme trajanja simulacije na 3 sekunde umjesto uobičajenih 10 sekundi. Parametri simulacije poput trajanja simulacije, izbora numeričke metode rješavanja, trajanja koraka simulacije i sl. se mijenjaju u ‘*Simulation*’→‘*Configuration Parameters*’ prozoru. Otvorite prozor i podesite vrijeme trajanja simulacije od 0 do 3 sekunde. Potvrdite načinjene izmjene i pokrenite simulaciju.

Da bi snimili odziv prikazanog sistema na impulsnu pobudu možemo na ulaz sistema dovesti razliku dva jedinična step signala kao što je prikazano na slici. Pri tome je parametre drugog step signala potrebno podesiti tako da se skok dešava 0.001 sekundu kasnije nego skok prvog step signala, dakle u trenutku 1.001.



Slika 1.8 Odziv sistema prvog reda na impulsnu pobudu

Dvostrukim klikom na ‘*Scope*’ možete vidjeti rezultate simulacije. Desnom tipkom miša kliknite na ikonicu ‘*Parameters*’ (pored ‘*Print*’ ikonice) i otvorite ‘*Data history*’ tab. Isključite opciju koja ograničava prikaz samo zadnjih 5000 tačaka (‘*Limit data points to last*’), te ponovo pokrenite simulaciju. Na ovaj način možete pogledati cijelu sliku. Pritiskom na ikonicu ‘*Autoscale*’ (ikonica u obliku dvogleda) automatski se podešavaju ose na osciloskopu tako da se prikaže cijeli grafik.

Pored ovih osobina moguće je koristiti ‘*Scope*’ da podatke dobivene simulacijom sačuvamo podatke o signalima koje smo obrađivali u radno okruženje MATLAB-a gdje ih je moguće sačuvati, grafički obraditi i sl. Ukoliko želite sačuvati vrijednosti signala u radni prostor MATLAB-a onda je potrebno u ‘*Data history*’ tab-u odaberati opciju ’*Save data to Workspace’,* imenovati niz podataka i odabrati formu u kojoj želimo da se sačuvaju. Ukoliko odaberemo ‘*Array*’ podaci će biti sačuvani kao matrica sa dvije kolone, gdje prva kolona predstavlja diskretne vrijednosti vremena, a druga kolona predstavlja vrijednosti signala na ulazu ‘*Scope’*-a u tim vremenskim trenutcima.

Altrenativan način pohranjivanja podataka dobivenih simulacijom pružaju blokovi ‘*To File’* i ‘*To Workspace’.* Blok ‘*To File*’ služi za čuvanje podataka iz simulacije u zaseban .mat koji ima isti naziv kao i varijabla. Ove podatke je moguće prenijeti u workspace pokretanjem .mat fajla naredbom ‘`load ime_fajla`’. Blok ‘*To Workspace’* pohranjuje podatke direktno u radnu prostor MATLAB-a.

Simulacija sistema drugog reda u kontinualnom vremenu

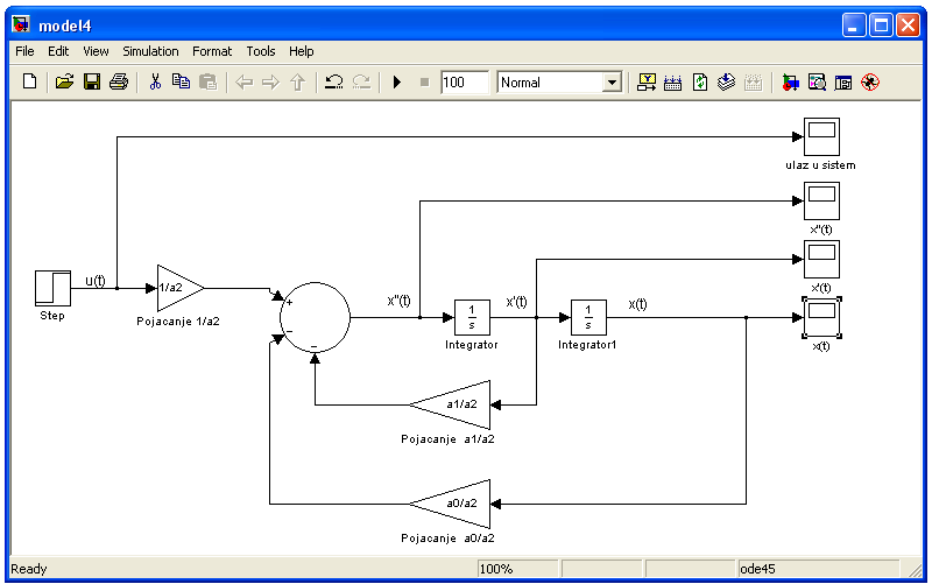
Razmotrimo sistem drugog reda koji je u opštem slučaju opisan sljedećom diferencijalnom jednačinom:

$$a_2 \frac{d^2x(t)}{dt^2} + a_1 \frac{dx(t)}{dt} + a_0 x(t) = u(t) \tag{1.3}$$

Da bi napravili Simulink model ovog sistema, prvo jednačinu (1.3) rješavamo po $d^2x(t)/dt^2$,a zatim dva puta integriramo.

$$\frac{d^2x(t)}{dt^2} = -\frac{a_1}{a_2}\frac{dx(t)}{dt} - \frac{a_0}{a_2}x(t) - \frac{1}{a_2}u(t) \tag{1.4}$$

Simulink model sistema je prikazan na slici 1.9.



Slika 1.9 Simulink model sistema drugog reda sa kontinualnim vremenom

Neka su date vrijednosti parametara $a_0 = 0.2$, $a_1 = 0.15$ i $a_2 = 0.2$. Ukoliko pokrenete simulaciju i pogledate odziv uočavate da je signal odziva ‘izlomljen’. Razlog za to je velika dozvoljena relativna tolerancija simulacije. Granice tolerancije možemo smanjiti u ‘*Simulation*’→‘*Configuration Parameters*’ prozoru. Umjesto postavljenjih 1e-3 odabraćemo 1e-5. Nakon što ponovo pokrenemo simulaciju uočavamo da je linija signala glatka.

Promatrajte odzive sistema za različite vrijednosti parametra a_1 (od 0.2 do 0.1)!

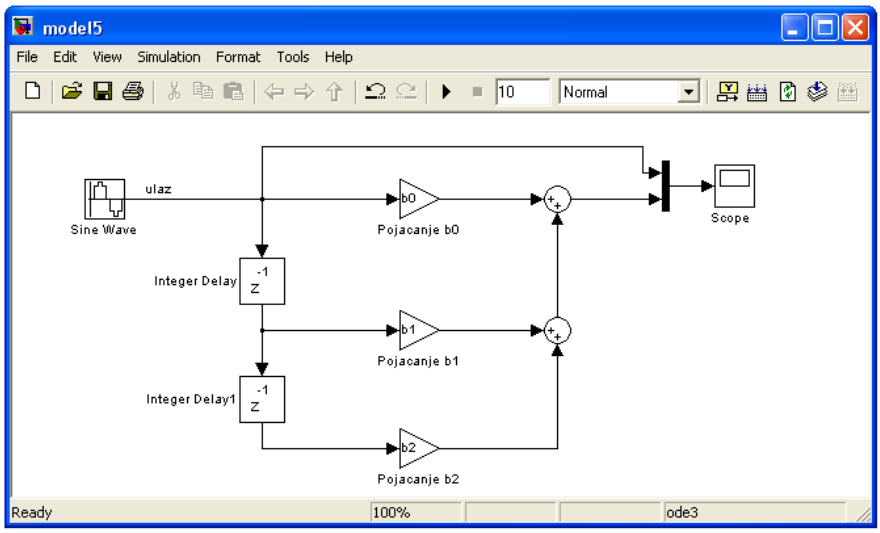
Simulacija sistema drugog reda u diskretnom vremenu

U opštem slučaju sistem drugog reda u diskretom vremenu je opisan diferentnom jednačinom:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] \tag{1.5}$$

S obzirom da je diferentna jednačina već riješena po sekvenci $y[n]$ možemo pristupiti pravljenju Simulink modela prikazanog na slici 10. Koeficijenti $b_0 = 0.5$, $b_1 = 1$ i $b_2 = 0.5$. Za pravljenje modela umjesto bloka '*Integrator*'-a koristimo blok '*Integer Delay*' iz '*Discrete*' grupe alata. Kašnjenje ‘*Integer delay*’ blokova podesimo na jedan. Izvor sinusoidalnog signala je podešen tako da mu je amplituda 5, a frekvenvcija 1Hz . Da bi sinusni signal bio diskretiziran postavićemo vrijeme uzorkovanja (‘*Sample time*’) na $ts = 0.1$. Prije nego pokrenemo

simulaciju potrebno je ‘*Simulation*’→‘*Configuration Parameters*’ prozoru promjeniti vrstu Solver-a iz ‘*Variable-step*’ u ‘*Fixed-step*’ i postaviti korak simulacije na $ts = 0.1$. Vrijeme trajanja simulacije je $10\ s$.



Slika 1.10 Simulink model sistema drugog reda sa diskretnim vremenom

2. Laboratorijske vježbe

Laboratorijska vježba br. 1
Osnovni elementi obrade zvuka u MATLAB-u – kao diskretnog signala

Uvod

Osnova razvoja današnje globalne/lokalne komunikacije je zasnovana na prijenosu govora, općenito prijenosu audio informacije. U 19. stoljeću su napravljeni prvi značajni koraci ka razvoju buduće industrije pružanja telekomunikacijskih usluga i industrije za pohranjivanje audio zapisa izumom telefona (*Alexandar Graham Bell* 1876.god.) i uređaja za pohranjivanje audio zapisa tzv. “fonografa” (*Thomas A. Edison 1877.god.*). Danas pohranjivanje, obrada, prenos i reprodukcija zvuka igraju značajnu ulogu u mnogim aspektima ljudskog života, od zabave i edukacije, do naučnoistraživačkih projekata i medicinskih usluga.

Zvuk sam po sebi predstavlja analognu fizikalnu veličinu, dakle vremenski kontinualan signal, kontinualne amplitude. Da bi mogli pohraniti ovaj signal u digitalnom računaru potrebno je snimiti zvuk pomoću elektromehaničkog pretvarača (mikrofon) i analogno digitalnog konvertora koji će izvršiti vremensku i amplitudnu diskretizaciju.

U MATLAB-u je zvuk poput svake druge varijable predstavljen kao matrica, preciznije vektor kolona (`mx1`). Za kreiranje zvučnog signala je dovoljan unos elemenata vektor kolone koja predstavlja vrijednost zvučnog signala u datom trenutku, pri čemu vrijednosti pripadaju segmentu `[-1,1]` i pri čemu se pretpostavlja da nam je poznata frekvencija uzorkovanja signala. MATLAB omogućava čitanje već kreiranih audio zapisa u *.wav* formatu naredbom ‘`wavread`’. Izvršavanje ove naredbe omogućavaju naredne sintakse:

```
>> zvuk = wavread(‘ime_fajla.wav’);
>> [zvuk, Fs] = wavread(‘ime_fajla.wav’);
```

gdje je ‘`Fs`’ frekvencija uzorkovanja zvuka, a ‘`zvuk`’ ime varijable u koju ćemo pohraniti zvučni signal.

Za snimanje audio zapisa u *.wav* formatu koristimo naredbu ‘`wavwrite`’, koja kao parametar vektor kolonu zvuka ‘`zvuk`’ i ime fajla u koji pohranjuje zvučni signal ‘`ime_fajla.wav`’. Moguće je (mada ne i neophodno) proslijediti i frekvenciju uzorkovanja ‘`Fs`’ i broj bita po uzorku ‘`N`’, kao što pokazuju sljedeće sintakse ove naredbe:

```
>> wavwrite(zvuk,‘ime_fajla.wav’);
>> wavwrite(zvuk,Fs,‘ime_fajla.wav’)
>> wavwrite(zvuk, Fs, N, ‘ime_fajla.wav’)
```

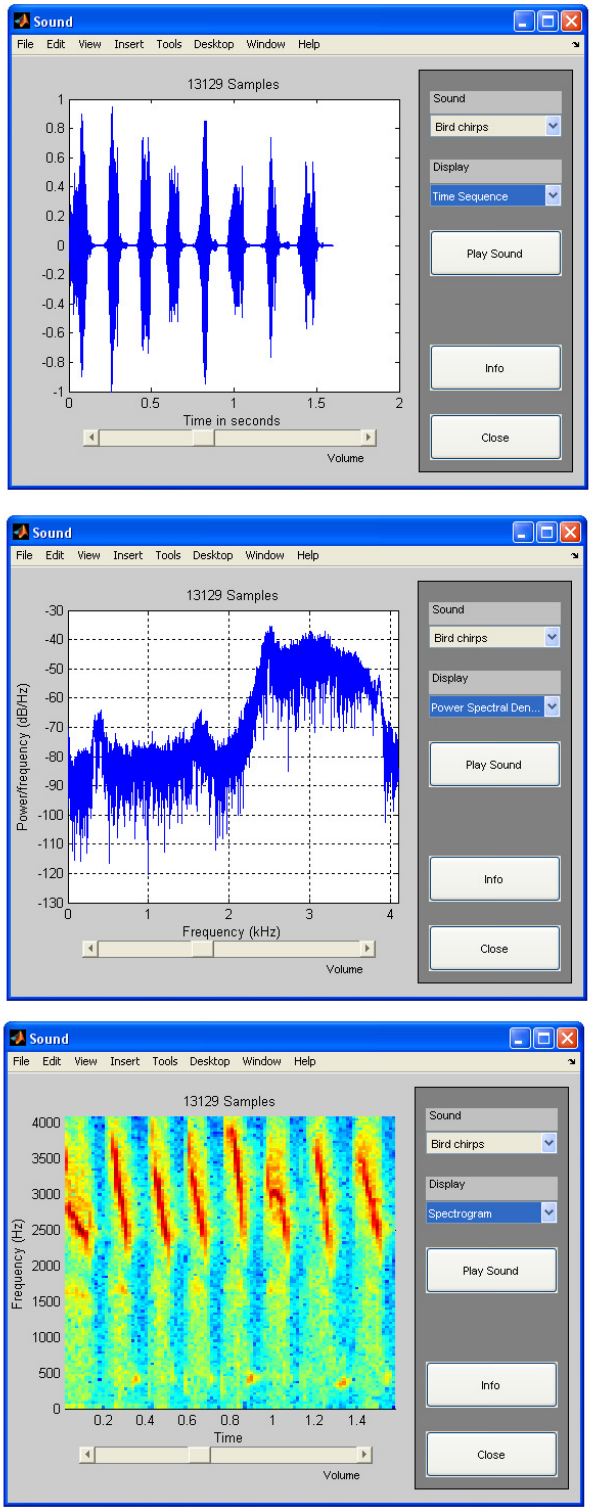
MATLAB omogućava i reprodukciju zvuka na zvučnoj kartici ukoliko je poznata frekvencija uzorkovanja pomoću naredbe ‘`sound`’.

```
>>  sound(zvuk,Fs)
```

Za vizualizaciju informacije sadržane unutar zvučnog signala MATLAB posjeduje grafički korisnički interfejs koji demonstrira mogućnosti obrade zvučnog signala sadržane u MATLAB-u nad nekoliko karakterističnih signala.

Pokreće se naredbom ‘`xpsound`’ i omogućava prikaz zvuka na tri načina:

1. Vremenski signal (‘*Time sequence*’) – prikazuje zvuk kao 2D grafik zvuka u funkciji vremena
2. PSD (‘*Power spectral density*’) – prikazuje jačinu snage signala na frekvencijama sadržanim u signalu
3. Spektrogram (‘*Spectrogram*’) –prikazuje promjenu frekventnog sadržaja signala u vremenu



Slika 2.1 GUI za demonstraciju mogućnosti vizualizacije zvučnog signala u MATLAB-u

Primjer 1.1:

Koristeći naredbe ‘`plot`’ i ‘`subplot`’ grafički prikazati sljedeće vremenski kontinualne funkcije $f: [-1,1] \to \mathbb{R}$.

$$\forall t \in [-1,1], \qquad f_1(t) = \sin(2\pi * 100t)$$
$$\forall t \in [-1,1], \qquad f_2(t) = e^{10t}\sin(2\pi * 100t)$$
$$\forall t \in [-1,1], \qquad f_3(t) = e^{-10t}\sin(2\pi * 100t)$$

Rješenje:

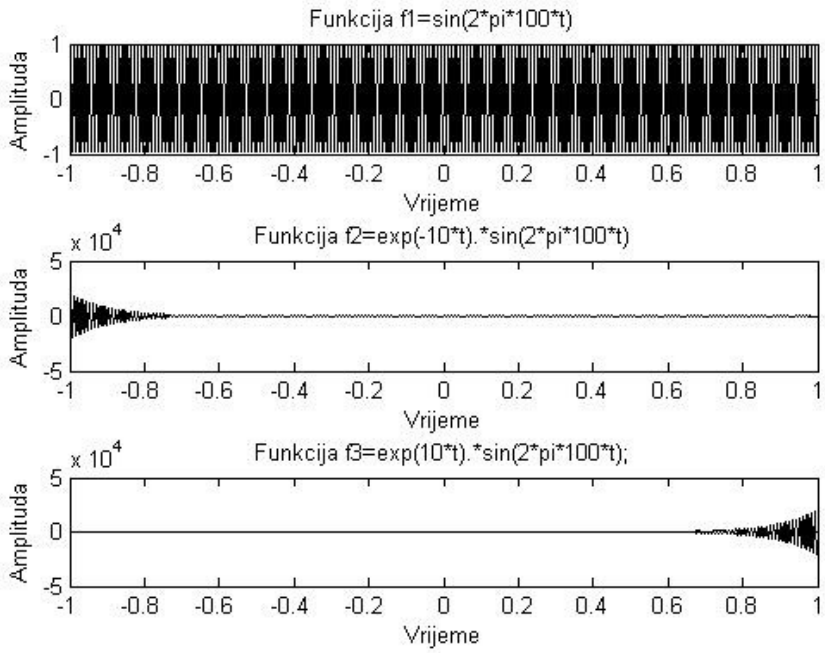
Kako u Matlabu nije moguće raditi sa kontinualnim signalima potrebno je diskretizirati vremenski interval i izračunati vrijednosti funkcije u odabranim trenucima. Da bi izvršili diskretizaciju po vremenu prvo moramo odabrati i unijeti neku frekvenciju uzorkovanja ‘`Fs`’ koja će omogućiti da dobijeni digitalni signal *nalikuje* analognom. Diskretizaciju vremena vršimo sljedećom naredbom:

```
>> t=-1:1/Fs:1;          %t=pocetak:korak:kraj, korak=period=1/frekvencija
```

Za računanje vrijednosti funkcija i crtanje funkcija u komandni prozor MATLAB-a unesite sljedeće komande:

```
>> f1=sin(2*pi*100*t);
>> f2=exp(-10*t).*sin(2*pi*100*t);
>> f3=exp(10*t).*sin(2*pi*100*t);
>> subplot(3,1,1), plot(t,f1,'k'),xlabel('Vrijeme'),
ylabel('Amplituda'),title('Funkcija f1=sin(2*pi*100*t)');
>> subplot(3,1,2), plot(t,f2,'k'),xlabel('Vrijeme'),
ylabel('Amplituda'),title('Funkcija f2=exp(-10*t).*sin(2*pi*100*t)');
>> subplot(3,1,3), plot(t,f3,'k'),xlabel('Vrijeme'),
ylabel('Amplituda'),title('Funkcija f3=exp(10*t).*sin(2*pi*100*t);');
```

Obrazložite svoj izbor frekvencije uzorkovanja ‘`Fs`’!



Slika 2.2 Grafički prikaz navedenih funkcija pomoću naredbi `plot` i `subplot`

Primjer 1.2:

Koristeći naredbu `sound` reprodukujte zvuk sinusoidalnog signala u trajanju 1 sekunde opisanog funkcijom:

$$\forall t \in [0,1]\,, \qquad f_2(t) = sin\,(2\pi * 440t)$$

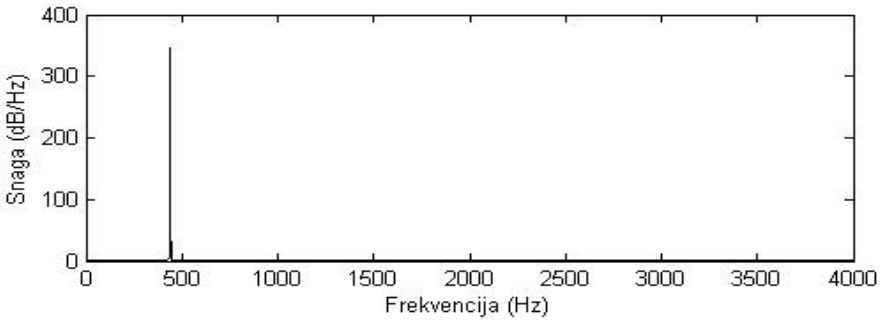
Pri tome koristite frekvenciju uzorkovanja 8000 uzoraka u sekundi. Potom pomnožite ovu funkciju funkcijom e^{-5t}, i reprodukujte novodobijeni signal. Pomoću funkcije ‘`periodogram`’ nacrtajte spektar snage signala. Više informacija o ovoj funkciji možete naći u Help-u.

Rješenje:

Sljedeći niz komandi daje željeno rješenje:

```
>> Fs=8000;
>> t=0:1/Fs:1;
>> f=sin(2*pi*440*t);
>> sound(f,Fs);
>> f=exp(5*t).*sin(2*pi*440*t);
>> sound(f,Fs);
>> [P,F] = periodogram(f,[],[],Fs)
>> plot(F,P,'k'),xlabel('Frekvencija (Hz)'),ylabel('Snaga (dB/Hz)');
```

Sljedeća slika prikazuje podjelu snage po frekvencijama u signalu:



Slika 2.3 Prikaz snage signala po frekvencijama

Primjer 1.3:

Konstruišite zvučni signal koji se sastoji od niza sinusoidalnih signala sa eksponencijalno opadajućom amplitudom, kao u prethodnom primjeru, svaki u trajanju pola sekunde. Frekvencije signala su 261, 261, 392, 392, 440, 440, 392, 349, 349, 329, 329, 293, 293 i 261 respektivno. Koristeći naredbu ‘`sound`’ reprodukujte ovaj zvučni signal.

Rješenje:

Za generisanje željenog zvučnog signala potrebno je unijeti sljedeći niz naredbi u radni prostor MATLAB-a:

```
>> Fs=8000;
>> t=0:1/Fs:0.5;
>> c4=exp(-5*t).*sin(2*pi*261*t);
>> g4=exp(-5*t).*sin(2*pi*392*t);
>> a4=exp(-5*t).*sin(2*pi*440*t);
```

```
>> d4=exp(-5*t).*sin(2*pi*293*t);
>> e4=exp(-5*t).*sin(2*pi*329*t);
>> f4=exp(-5*t).*sin(2*pi*349*t);
>> f=[c4 c4 g4 g4 a4 a4 g4 f4 f4 e4 e4 d4 d4 c4];
>>sound(f,Fs);
```

Možete li prepoznati melodiju?

Laboratorijski zadatak

Priprema:

Uz pomoć računara i programa Sound Recorder (*Start →All Programs →Accessories →Entertainment →Sound Recorder*) napravite snimak ljudskog govora ne dužeg od 10 sekundi. Možete uz pomoć mikrofona npr. snimiti sebe kako izgovarate vaše ime i prezime i broj indexa, ili napraviti isječak iz nekog filmskog dijaloga. Snimak sačuvajte kao ‘*govor.wav*’. Napravite isječak iz neke pjesme ne duži od 10 sekundi i pomoću Sound Recorder-a ga snimite kao ‘*muzika.wav*’. Prilikom snimanja obavezno koristite najveći mogući kvalitet snimanja zvuka.

OBAVEZNO ponijeti slušalice na laboratorijske vježbe!!!

Rad u laboratoriji:

a) Učitajte podatke iz audio snimka koji sadrži muziku u radno okruženje MATLABA koristeći naredbu ‘`wavread`’. Pomoću naredbe ‘`sound`’ možete reprodukovati muzički signal na zvučnoj kartici vašeg računara.

b) Preslušajte zvuk uz pomoć komande ‘`soundsc`’. Za više informacija o ovoj komandi u radni prostor MATLAB-a unesite:
```
>> help soundsc
```

c) Reprodukujte zvuk koji u odnosu na vaš snimljeni govor ima:
   - dvostruko veću amplitudu
   - dvostruko veću frekvenciju uzorkovanja
   - dvostruko manju amplitudu
   - dvostruko manju frekvencij uzorkovanja

d) Učitajte podatke iz audio snimka koji sadrži govor u radno okruženje MATLABA koristeći naredbu ‘`wavread`’. Pomoću naredbe ‘`sound`’ možete reprodukovati govorni signal na zvučnoj kartici vašeg računara.

e) Napravite m-file koji generiše novi signal tako što uzima svaki n-ti uzorak iz vašeg snimljenog signala, pri čemu vi određujete n.

f) Koje je maksimalno n za koje se još uvijek može razumjeti sadržaj govora?

g) Nacrtajte spektar snage govornog i muzičkog zapisa? Koje frekvencije sadrže?

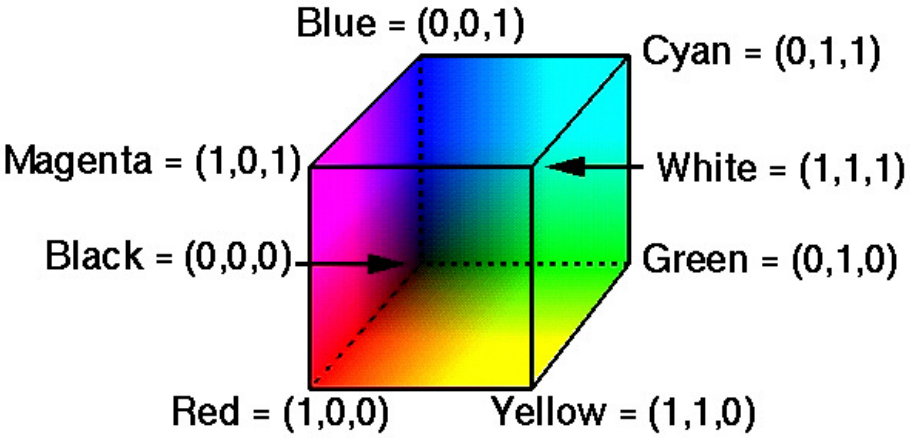
h) Koristeći naredbu ‘`conv(muzika,vektor)`’ napravite konvoluciju vašeg muzičkog signala sa vektor kolonom ispunjenim nulama dimenzija (50000,1) koji sadrži dvije jedinice, pri čemu je prva na 10000-toj poziciji, a druga na 20000-poziciji.

Laboratorijska vježba br. 2
Osnovni elementi obrade slike u MATLAB-u – kao višedimenzionalnog signala

Uvod

Veoma često u modernoj industriji telekomunikacija se javlja potreba za obradom slike, odnosno video signala. Prikupljanje i obrada ovih sinala su složeni i zahtjevni zbog obima informacija koje svaka slika sadrži. MATLAB je razvio '*Image Processing Toolbox*' koji predstavlja skup alata namjenski razvijenih za potrebe obrade slike.

Za unos slike u MATLAB koristimo funkciju '`imread('ime_slike.jpg')`' . Osim JPEG formata zapisa slike, MATLAB podržava učitavanja slika i u formatima BMP, PNG, GIF, TIFF,ICO... Više informacija možete pronaći u sistemu pomoći naredbom '`help imread`'. Prije nego što pređemo na opis funkcija za obradu slike koje su ugrađene u MATLAB, potrebno je posvetiti pažnju načinu na koji je slika predstavljena u MATLAB-u. Slika se učitava u radno okruženje MATLAB-a kao RGB slika. U radnom prostoru slika je predstavljena kao matrica dimenzija `mxnx3`, gdje m i n predstavljaju dimenzije slike u pikselima. Vrijednosti matrice predstavlju zastupljenost svake od tri boje (crvene, zelene i plave) u kreiranju boje posmatranog piksela.



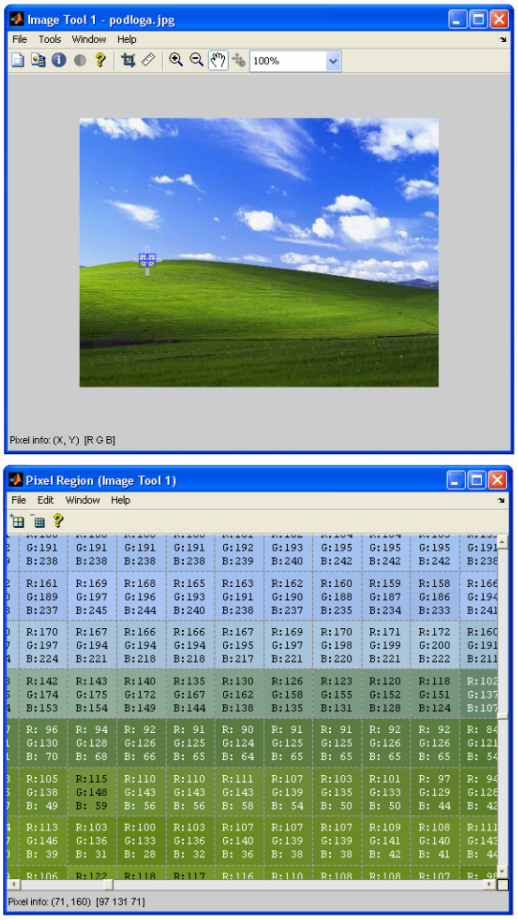
Slika 2.4 RGB paleta boja

Paleta boja u MATLAB-u je matrica dimenzija `mx3` popunjena realnim brojevima koji pripadaju segmentu [0, 1]. Svaki red je jedna boja iz palete predstavljena RGB vektorom, tako k-ti red matrice predstavlja k-tu boju u paleti gdje '`map(k,:) = [r(k) g(k) b(k)]`' označava intenzitet crvene, zelene i plave boje respektivno.

Kada govorimo o predstavljanju crno-bijelih slika u MATLAB-u, očito je da je njihovo čuvanje u RGB paleti boja nepraktično, jer zauzimaju više memorijskog prostora nego što je potrebno. Zbog toga njih čuvamo u paleti sivih tonova, u kojoj je zapis slike predstavljen kao dvodimenzionalna matrica dimenzija `mxn` gdje m i n predstavljaju dimenzije slike izražene u pikselima.

Konverziju slike iz RGB palete boja u paletu sivih tonova vršimo komandom '`rgb2gray(slika)`', gdje je '`slika`' matrični zapis slike učitan naredbom '`imread`'. Prikaz slike vršimo pomoću naredbe '`imshow`'.

MATLAB posjeduje i GUI za vizualizaciju podataka o slici koji se pokreće naredbom '`imtool`'. Sljedeća slika prikazuje primjer upotrebe Image Tool-a za pregled vrijednosti boje piksela slike u RGB paleti boja.



Slika 2.5 Pregled vrijednosti boje piksela slike u RGB paleti boja upotrebom Image Tool-a

MATLAB podržava 13 paleta boja jet, hsv, hot, cool, spring, summer, autumn, winter, gray, bone, copper, pink i lines, čiji se odabir vrši naredbom ‘`colormap`’. Odabir colormap-e za jedan grafički prozor se odnosi na sve sadržaje u tom grafičkom prozoru.

Slika 2.6 Preddefinirane palete boja sadržane u MATLAB-u

Primjer 2.1:

Kreirajte vektor red dimenzija koji sadrži vrijednosti od nula do jedan sa korakom 0.01 i prikažite ga kao sliku koristeći naredbu ‘`imshow`’. Odabrite paletu boja za koje ćete grafički prikazati vašu vektor kolonu kao sliku.

Napravite novi vektor red koji sadrži vrijednost od 0 do 64 sa korakom 1 i prikažite ga u zasebnom grafičkom prozoru kao sliku koristeći naredbu ‘`image`’.

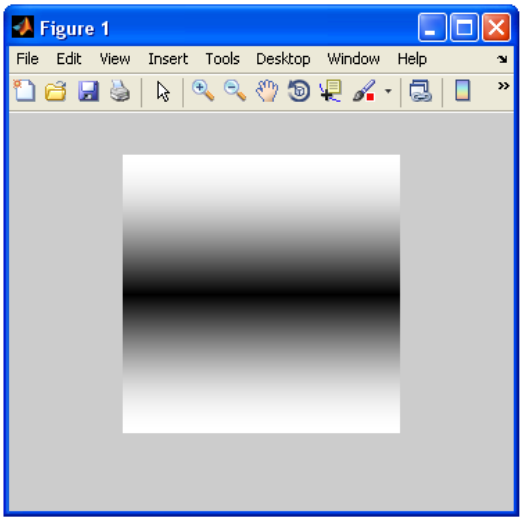
Rješenje:

Kako se odabir colormap-e odnosi na cijeli grafički prozor potrebno je svaku colormapu prikazati u zasebnom grafičkom prozoru. Zadavanje vektora reda i prikaz izbora palete boja vršimo sljedećim naredbama:

```
>> v1=[0:0.01:1];
>> figure
>> imshow(v1),colormap('jet');
```

Primjer 2.2:

Napravite m-file koji kreira sliku prikaznu na slici ispod dimenzija 200x200 piksela. Pikseli koji se nalaze u istom redu imaju istu vrijednost, dok se vrijednosti vertikalno susjednih piksela mijenja prema sinusoidalnom zakonu. Odrediti potrebnu frekvenciju sinusoide.



Slika 2.7 Grafički prikaz sinusoidalne ovisnosti vrijednosti piksela na slici

Rješenje:

Za generisanje tražene slike možemo koristiti ‘`for`’ petlju ili ‘`repmat`’ komandu. Više podataka o ovoj komandi možete pronaći u odgovarajućem Help file-u.

```
v=zeros(200,200);
for i=1:1:200
    for j=1:1:200
    v(i,j)=abs(cos(2*pi*(i-1)/400));
    end
end
imshow(v);
```

Funkcija ‘`abs`’ nam je potrebna jer MATLAB pri prikazu slike, sve vrijednosti koje izlaze iz opsega (0,1) odsjeca, odnosno zaokružuje na maksimumalnu/minimalnu dozvoljenu vrijednost.

M-file koji korištenjem naredbe ‘`repmat`’ generiše traženu sliku je dat ispod:

```
t=1:1:200;
x=abs(sin(2*pi/400*t));
v=repmat(x',1,200);
imshow(v)
```

Često je u praksi neophodno odrediti ivice predmeta na slici. Jednostavan metod određivanja ivica je konvolucija matričnog predstavljanja slike sa matricom koja predstavlja filter. Ova matrica je data u sljedećem obliku:

$$h = \begin{bmatrix} -1 & -1 & -1 \\ -1 & a & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Pri čemu je, a realan parametar. Po uzoru na dati filter možete kreirati vlastite filtre mijenjam vrijednosti članova matrice ‘`h`’ i posmatrati njihov učinak na sliku.

Primjer 2.3:

Odrediti ivice predmeta sa slike (odabir slike je proizvoljan) i grafički prikazati rezultat:

Slika 2.8 Primjer slike sign.jpg

Rješenje:

Postavite vrijednost varijable ‘`a`’ i vodite računa o tipu varijabli. U radno okruženje MATLAB-a je potrebno unijeti sljedeći kod:

```
>> a=9;
>> s=imread('sign.jpg');
>> sbw=rgb2gray(s);
>> sd=double(sbw)+1;        % pretvaranje u tip double
>> h = [-1,-1,-1; -1, a ,-1; -1,-1,-1];
>> Xd=conv2(sd,h);
>> X=int8(Xd -1);           % pretvaranje u tip int8
>> figure
>> subplot(1,2,1),imshow(s),xlabel('Originalna slika');
>> subplot(1,2,2),imshow(X),xlabel('Detektovane ivice');
```

Laboratorijski zadatak

Priprema:

Kao pripremu za ovu vježbu obavezno je ponijeti dvije slike u JPEG formatu. Prva slika treaba da ima dimenzije (300x400) piksela i tu sliku ćete koristiti kao podlogu. Sliku nazvati *'podloga.jpg'*. Druga slika treba predstavljati neki objekat, loptu ili sl. i treba imati dimenzije 64x64 piksela. Pri tome svi pikseli koji ne predstavljau sam objekat trebaju biti obojeni u crno. Sliku nazovite *'objekat.jpg'*.

Rad u laboratoriji:

a) Napravite sliku kao u primjeru 2.2 čiji je sadržaj vertikalno konstanta, a horizontalno se mijenja po sinuoidalnom zakonu.
b) Promjenite frekvenciju sinusoide tako da na slici dobijete 4 crne pruge.
c) Napravite sliku čiji se sadržaj mijenja i horizontalno i vertikalno tako da piksel na poziji (i,j) predstavlja proizvod vrijednosti horizontalne i vertikalne promjene. Pogledajte skalarno množenje matrica.
d) Učitajte slike *'podloga.jpg' i 'objekat.jpg'* pomoću naredbe '`imread`'. Kojim tipom podataka je slika predstavljena u MATLAB-u. Koje su dimenzije matrica kojima su slike predstavljene?
e) Izdvojite količinu crvene boje u svakom pikselu u zasebno matricu. Prikažite novokreiranu matricu u zasebnom prozoru. Ponovite postupak za plavu i zelenu boju. Prikazite tako kreirane slike zajedno u jednom prozoru koristeći naredbu '`subplot`'.
f) Pretvorite obje vaše slike *'podloga.jpg' i 'objekat.jpg'* iz RGB u grayscale pomoću naredbe '`rgb2gray`'. Koje su nove dimenzije matrica kojima su slike predstavljene?
g) Izvršite konverziju tipa podataka kojima su slike predstavljene u double. Konverzija podataka iz double u int8 i obratno se vrši na sljedeći način:

```
varijabla_double=double(varijbla_int8)+1;
varijabla_int8=int8(varijbla_double -1);
```

h) Napravite crnu sliku dimezija 300x400 piksela, sa samo jednom bijelom tačkom na slici (npr. na poziciji 100x100). Koristeći naredbu '`conv2`' napravite konvoluciju vašeg objekta sa ovom slikum. Prikažite rezultat pomoću naredbe '`imshow`'. Obrazložite dimenzije nastale slike.
i) Ponovite prethodni korak ali ovaj put odaberite dimenzije crne slike tako da slika nastala konvolucijom ima tačno dimenzije 300x400 piksela.
j) Saberite sliku nastalu konvolucijom sa slikom podloge. Prikažite rezultat u zasebnom grafičkom prozoru.

Laboratorijska vježba br. 3
Crtanje signala i aproksimacionih funkcija – razvoj funkcije u Fourierov red

Uvod

Poznato je da svaku periodičnu funkciju koja zadovoljava Dirichlet-ove uvjete možemo razviti u Fourierov red. Čak i funkcije koje nisu periodične, na konačnom intervalu možemo razviti u Fourierov red, pri čemu se razvoj funkcije u Fourierov red poklapa sa samom funkcijom samo na tom intervalu.

Aproksimacija signala $f(t)$ na intervalu $[a, b]$ kompleksnim harmonijskim funkcijama data je izrazom:

$$f(t) = \sum_{k=-\infty}^{\infty} C_k f_k(t) = \sum_{k=-\infty}^{\infty} C_k e^{-jk\omega_0 t} \qquad a \leq t \leq b \tag{2.3.1}$$

Pri čemu je $\omega_0 = \frac{2\pi}{T_0}$, gdje je $T_0 = b - a$ osnovni (fundamentalni) period periodičkog signala koji se aproksimira. Koeficijenti Fourierovog reda C_k signala $f(t)$ određuju udio funkcije $f_k(t) = e^{-jk\omega_0 t}$ u formiranju funkcije $f(t)$, a računaju se iz relacije:

$$C_k = \frac{1}{T_0} \int_a^{a+T_0} f(t) e^{-jk\omega_0 t}\, dt \tag{2.3.2}$$

Alternativno, funkciju predstavljenu preko Fourierovog reda možemo zapisati na sljedeći način:

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t)] \tag{2.3.3}$$

Pri čemu se koeficijenti a_0, a_k i b_k dobiju iz sljedećih formula:

$$a_0 = \frac{2}{T_0} \int_{-T_0/2}^{T_0/2} f(t)\, dt$$

$$a_k = \frac{2}{T_0} \int_{-T_0/2}^{T_0/2} f(t) \cos(k\omega_0 t)\, dt \tag{2.3.4}$$

$$b_k = \frac{2}{T_0} \int_{-T_0/2}^{T_0/2} f(t) \sin(k\omega_0 t)\, dt$$

Između spektralnih koeficijenata C_k i realnih koeficijenata razvoja funkcije u Fourierov red a_k i b_k važe sljedeće relacije:

$$C_k = \frac{a_k - jb_k}{2} \tag{2.3.5}$$

$$C_{-k} = \frac{a_k + jb_k}{2} \tag{2.3.6}$$

Kao provjeru, da li smo tačno izračunali koeficijente Fourierovog reda možemo koristiti Parsevalovu relaciju za kontinualne periodičke signale koja ima oblik:

$$\frac{1}{T_0}\int_{T_0}|f(t)|^2\,dt = \sum_{k=-\infty}^{\infty}|C_k|^2 \tag{2.3.7}$$

Odnosno,

$$\frac{1}{T_0}\int_{T_0}|f(t)|^2\,dt = a_0^2 + \frac{1}{2}\sum_{k=1}^{\infty}(a_k^2 + b_k^2) \tag{2.3.8}$$

Primjer 3.1:

Potrebno je razviti sljedeću funkciju u Fourierov red:

$$f(x) = \begin{cases} 1, & m \le t < m + \frac{1}{2}, \quad m \in \mathbb{Z} \\ -1, & m + \frac{1}{2} \le t < m + 1\,, \quad m \in \mathbb{Z} \end{cases}$$

Nacrtati stvarnu funkciju i funkciju dobivenu aproksimacijom funkcije $f(t)$ sa prvih 50 članova Fourireovog reda funkcije $f(t)$, na istom grafiku, na intervalu $\left[-\frac{1}{2}, \frac{1}{2}\right]$ i uporediti ih.

Rješenje:

Zadana funkcija predstavlja periodičnu funkciju sa periodom $T_0 = 1$, preciznije povorku četvrtki jedinične amplitude. Fourierov red je predstavljen sa:

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty}[a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t)]$$

Gdje je:

$$a_0 = \frac{2}{T_0}\int_{-T_0/2}^{T_0/2} f(t)\,dt$$

$$a_k = \frac{2}{T_0}\int_{-T_0/2}^{T_0/2} f(t)\cos(k\omega_0 t)\,dt$$

$$b_k = \frac{2}{T_0}\int_{-T_0/2}^{T_0/2} f(t)\sin(k\omega_0 t)\,dt$$

Kako je funkcija $f(t)$neparna, svi koeficijenti a_k su jednaki nuli, $\forall k \ge 0$. Zbog svojstva simetrije slijedi da je:

$$b_k = \frac{2}{1} * 2\int_0^{0.5}\sin(2k\pi t)\,dt = -4\frac{\cos(2k\pi t)}{2k\pi}\Bigg|_{t=0}^{t=\frac{1}{2}} = \begin{cases} \frac{4}{k\pi}\,, & za\ neparno\ k \\ 0, & za\ parno\ k \end{cases}$$

Uvrštavajući dobivene rezultate, Fourireov red funkcije $f(t)$ dobijemo u sljedećem obliku:

$$f(t) = \sum_{k=1}^{\infty}\frac{4}{(2k+1)\pi}\sin(2\pi(2k+1)t)$$

Sada tačnost izračunatih koeficijenta možete provjeriti koristeći Parsevalovu jednakost. Za razmatranu funkciju treba da vrijedi:

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} f^2(t)dt = a_0^2 + \frac{1}{2}\sum_{k=1}^{\infty}(a_k^2 + b_k^2) = \frac{1}{2}\sum_{k=1}^{\infty}\left(\frac{4}{(2k+1)\pi}\right)^2 = \frac{8}{\pi^2}\sum_{k=1}^{\infty}\left(\frac{1}{2k+1}\right)^2$$

U MATLAB-u ovu provjeru vršimo unošenjem sljedećih komandi:

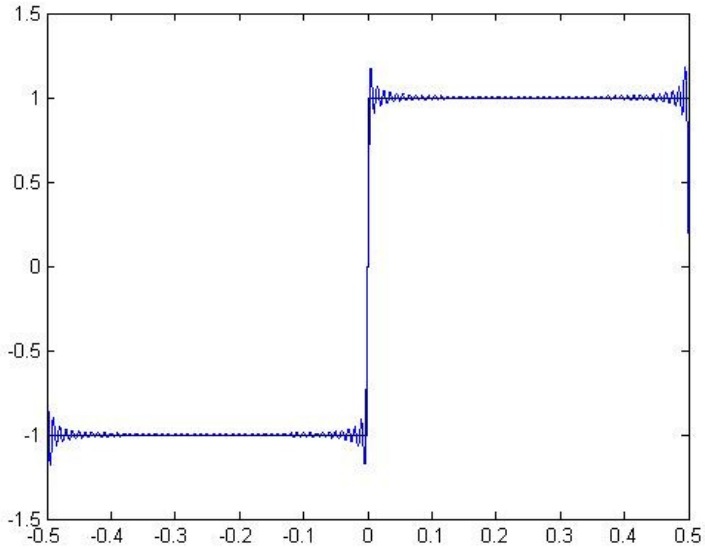
```
>> k=[1:2:10000];
>> S=(8/pi^2)*sum(1./k.^2)
```

Rezultat sumiranja treba biti 1, čime potvrđujemo tačnost izračunatih koeficijenata. Alternativni način dobijanja ovog rezultata je korištenjem simboličkih varijabli, kao što prikazuje naredni blok komandi.

```
>> syms k
>> S=(sym('8')/sym('pi')^2)*symsum(1/(2*k+1)^2, k,0,Inf)
```

Unesite u radno okruženje MATLAB-a sljedeći niz naredbi koji vrši prikaz funkcije $f(t)$ i funkcije dobivene aproksimacijom funkcije $f(t)$ sa prvih 50 članova Fourireovog reda funkcije $f(t)$.

```
>> t=linspace(-0.5,0.5,1000);
>> f=[-ones(1,500)  ones(1,500)];
>> fF=zeros(size(t));
>> for k=1:2:101
fF=fF+4/pi*(sin(2*pi*k*t))/k;
end
>> plot(t,f,'k',t,fF,'b');
```

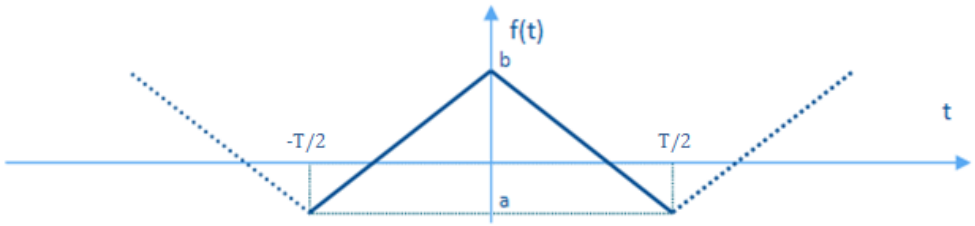


Slika 2.9 Originalna funkcija f(t) i suma prvih 50 člana Fourireovog reda funkcije f(t)

Laboratorijski zadatak

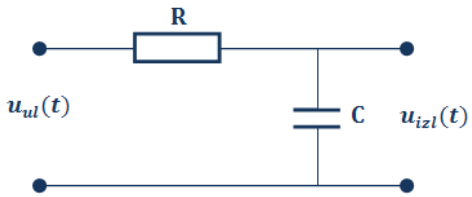
Priprema:

Izračunajte na papiru koeficijente razvoja funkcije $f(t)$ prikazane na slici 2.9, za $-\frac{T}{2} < t \leq \frac{T}{2}$. Definišite njoj odgovarajuću izvedenu periodičku funkciju $f_T(t)$. Izračunajte koeficijente razvoja izvedene periodičke funkcije u Fourierov red c_k i nacrtajte odgovarajuće spektre (realni, imaginarni, amplitudni i fazni). Na osnovu izračunatih koeficijenata razvoja c_k, odredite koeficijente razvoja funkcije u Fourierov red a_k i b_k.



Slika 2.10 Posmatrani signal f(t)

Dat je idealni niskopropusni filter sa graničnom frekvencijom ω_c. Odrediti graničnu frekvenciju $\omega_c^{(1)}$ tako da se propusti samo nulta komponenta i prvi harmonik signala $f_T(t)$. Skicirati izlaze $g(t)$ (u vremenskom domenu!) iz filtera ako se na ulaz dovode signal $f_T(t)$ sa slike 2.9. (Uputstvo: izvršiti rekonstrukciju signala g(t) na osnovu propuštenih harmonika!)



Slika 2.11 RC filter

Dat je RC filter kao na slici 2.10. Napisati diferencijalnu jednačinu koja opisuje dati sistem. Naći frekventnu funkciju sistema. Skicirati $|H(j\omega)|$. Izračunati koeficijente razvoja u Fourierov red signala na izlazu $h(t)$ ako se na ulaz dovede signal $f_T(t)$. Vrijednosti za R i C odaberite proizvoljno, ali tako da vrijedi $R * C = T/8$. (Npr. $T = 5$, pa je $R * C = 5/8$, usvajamo recimo $R = 500$, $C = 1/800$).

Rad u laboratoriji:

a) Korištenjem MATLAB-a izračunajte koeficijente razvoja funkcije $f_T(t)$ u Fourierov red.
b) Provjerite da li izračunati koeficijenti zadovoljavaju Parsevalovu jednakost.
c) Saberite prvih deset članova dobivenog Fourierovog reda i grafički uporedite dobivenu sumu sa početnom funkcijom.
d) Saberite prvih pedeset članova dobivenog Fourierovog reda i grafički uporedite dobivenu sumu sa početnom funkcijom.
e) Prikažite grafički signal g(t) dobiven na izlazu niskopropusnog filtra. Uporedite ga grafički sa funkcijom $f_T(t)$.

f) Prikažite grafički realni, imaginarni, amplitudni i fazni spektar signala na ulazu i signala na izlazu niskopropusnog filtra $g(t)$.
g) Prikažite grafički realni, imaginarni, amplitudni i fazni spektar signala na ulazu i signala na izlazu RC filtra $h(t)$.

Laboratorijska vježba br. 4
Fourierova i Hilbertova transformacija u MATLAB-u

Uvod

Fourierova transformacija

Za analizu karakteristika kontinualnih signala i sistema i veza između kontinualnih signala i sistema često je pogodno koristiti Fourierovu transformaciju. Fourierovu transformaciju koristimo da predstavimo vremenski kontinualan signal $f(t)$, koji nije nužno periodički, u kompleksnom domenu. Da bi na funkciju $f(t)$ mogli primjeniti Fourierovu transformaciju dovoljno je da ispunjava Dirichlet-ove uslove, te da je apsolutno integrabilna. Direktna Fourierova transformacija $F(j\omega)$ vremenski kontinualne funkcije $f(t)$ je data jednačinom (2.4.1). Inverzna Fourierova transformacija $f(t)$ kompleksne funkcije $F(j\omega)$ je data jednačinom (2.4.2).

Lik: $$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{2.4.1}$$

Original: $$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega \tag{2.4.2}$$

Veza između originala i lika funkcije u Fourierovoj transformaciji je jednoznačna.

Primjer 4.1:

Odrediti Fourierovu transformaciju pravougaonog signala $f(t) = p\left(\frac{t}{a}\right)$:

$$f(t) = p\left(\frac{t}{a}\right) = \begin{cases} 1, & t \leq \left|\frac{a}{2}\right|, \quad a \in \mathbb{Z} \\ 0, & t > \left|\frac{a}{2}\right|, \quad a \in \mathbb{Z} \end{cases}$$

Prikazati grafički zadanu funkciju te realni i imaginarni spektar Fourierove transformacije funkcije.

Rješenje:

Da bi našli Fourierovu transformaciju, potrebno je prije svega definisati zadanu funkciju u MATLAB-u. Ova funkcija predstavlja razliku dvije odskočne funkcije. Pri tome koristimo sljedće poznate realcije:

$$p\left(\frac{t}{a}\right) = u\left(t + \frac{a}{2}\right) - u\left(t - \frac{a}{2}\right)$$

$$u(t) = \frac{1 + sgn(t)}{2}$$

$$sgn(t) = \frac{t}{|t|}$$

Definisanje zadane funkcije korištenjem simboličkih varijabli je moguće korištenjem datog koda:

```
>> syms t a                          % definisanje varijabli
>> sgn =t/abs(t)                     % signum funkcija
>> stp=(sym('1')+sgn)/2              % odskočna funkcija
>> rect=subs(stp,t+a/sym('2'))-subs(stp,t-a/sym('2'))
```

Rezultat koji ispusuje MATLAB je uglavnom nerazumljiv i nepregledan, kao što je prikazano. Ukoliko želimo da forma u kojoj je rezultat ispisan bude jasnija i čitljivija možemo koristiti naredbu ‘`pretty`’.

```
rect =

(a/2 - t)/(2*abs(a/2 - t)) + (a/2 + t)/(2*abs(a/2 + t))

>> pretty(rect)

    a           a
    - - t       - + t
    2           2
  --------- + ---------
    |a    |     |a    |
  2 |- - t|   2 |- + t|
    |2    |     |2    |
```

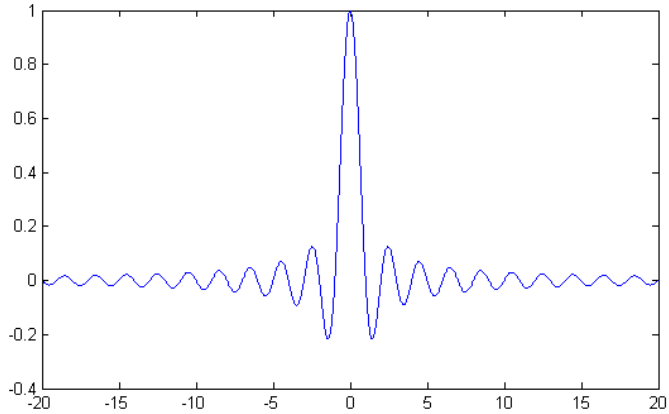
Za računanje Fourierove transformacije signala $rect(t)$ za datu vrijednost $a = 1$ potrebno je izračunati integral dat jednačinom (2.4.1). Prije toga je neophodno definisati frekvenciju u kompleksnom domenu ‘`f`’.

```
>> syms f
>> F=int(exp(-j*2*pi*f*t)*subs(rect,a,1),t,-inf,inf);
>> pretty(F)

  sin(pi f)
  ---------
    pi f
```

Za dobijanje grafičkog prikaza ove funkcije na širem opsegu frekvencija, kao na slici, koristimo sljedeće komande:

```
>> t=-20:0.005:20;
>> F1=subs(F,t);
>> plot(t,F1)
```



Slika 2.12 Realni/amplitudni spektar signala rect(t)

Alternativno, moguće je naći Fourierovu transformaciju signala koristeći isključivo preddefinisane funkcije za rad sa simboličkim varijablama, kao što pokazuje sljedeći niz naredbi:

```
>> syms t
>> rect=heaviside(t+sym('1/2'))-heaviside(t-sym('1/2'));
>> F=fourier(rect);
>> simplify(F)

ans =
(2*sin(w/2))/w

>> ezplot(F)
```

Primjer 4.2:

Odrediti i grafički prikazati signal, ako je data njegova Fourierova transformacija:

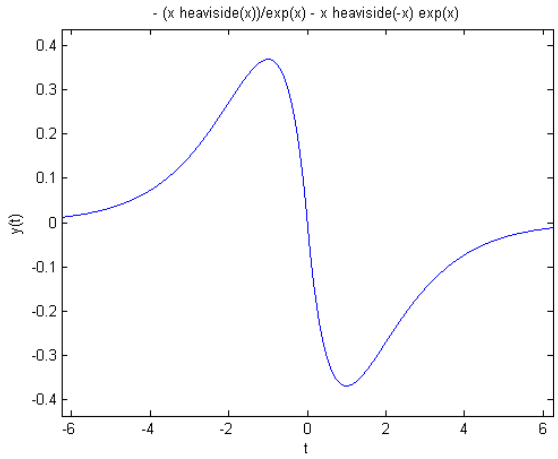
$$F(j\omega) = \frac{4j\omega}{(1+\omega^2)^2}$$

Rješenje:

Da bi odredili inverznu Fourierovu transformaciju date funkcije, u radni prostor MATLAB-a je potrebno unijeti sljedeće komande:

```
>> syms w
>> F=4*i*w/((1+w^2)^2);
>> f=ifourier(F);
>> pretty(simplify(f))

    x heaviside(x)
  - -------------- - x heaviside(-x) exp(x)
        exp(x)
>> ezplot(f), xlabel('t'),ylabel('f(t)');
```



Slika 2.13 Inverzna Fourierova transformacija funkcije $F(j\omega) = \frac{4j\omega}{(1+\omega^2)^2}$

Hilbertova transformacija

Hilbertova transformacija za razliku od Fourireove transformacije transformiše realnu vremenski kontinualnu funkciju $f(t)$ u realnu vremenski kontinualnu funkciju $\hat{f}(t)$. Hilbertova transformacija predstavlja konvoluciju realne funkcije $f(t)$ sa funkcijom $\frac{1}{\pi\, t}$ i data je jednačinom (2.4.3):

$$H\{f(t)\} = \hat{f}(t) = f(t) * \frac{1}{\pi\, t} \tag{2.4.3}$$

Od posebnog značaja u analizi signala i sistema je Hilbertov filter koji mijenja fazu ulaznog harmonijskog signala za $\frac{\pi}{2}$, odnosno za jedan kvadrant, zbog čega se još naziva i kvadraturni filter. Od posebnog značaja nam je i Fourierova transformacija Hilbertove transformacije signala data jednačinom (2.4.4):

$$F\{\hat{f}(t)\} = \hat{F}(j\omega) = F(j\omega) \cdot F\left\{\frac{1}{\pi t}\right\} \tag{2.4.4}$$

Primjer 4.3:

Odrediti Hilbertovu transformaciju sinusoidalnog signala $f(t) = \cos(t)$. Prikazati grafički zadanu i dobivenu funkciju.

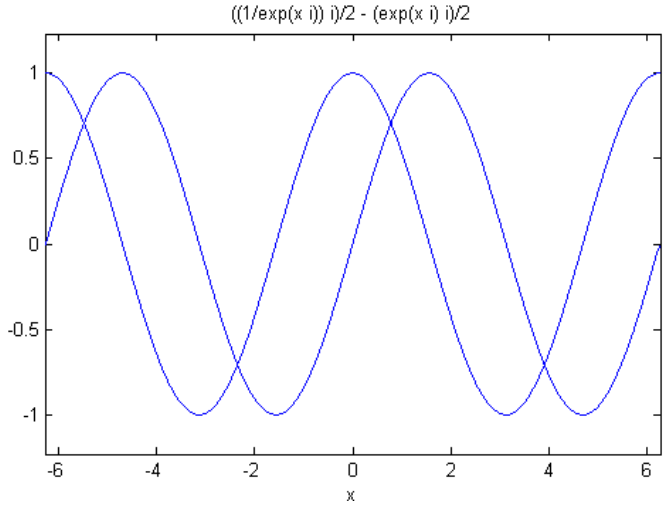
Rješenje:

U radni prostor MATLAB-a je potrebno unijeti sljedeće komande:

```
>> G=F*H;
>> g=ifourier(G);
>> pretty(simplify(g))

  sin(x)

>> ezplot(g)
>> hold on
>> ezplot(f)
```



Slika 2.14 Signal f(t)=cos(t) i njegova Hilbertova transformacija

Laboratorijski zadatak

Priprema:

Posmatrajmo signal $f(t)$ opisan izrazom:

$$f(t) = \begin{cases} -0.1\,t + 0.5t, & 0 \le t \le 5 \\ 0, & za\ svako\ drugo\ t \end{cases}$$

na koji djeluje smetnja $s(t)$ data izrazom:

$$s(t) = \begin{cases} 0.03\cos(10t) - 0.01\sin(20t) & 0 \le t \le 10 \\ 0, & za\ svako\ drugo\ t \end{cases}$$

Prikazati na istom grafiku ova dva signala, kao i za rezultujući signal $x(t) = f(t) + s(t)$. Izračunati Fourireovu transformaciju signala bez upotrebe MATLAB-a i nacrtati realne i imaginarne spektre signala.

Rad u laboratoriji:

a) Korištenjem simboličkog računa u MATLABU izračunajte Fourierovu transformaciju sljedećih funkcija:

$$f(t) = \begin{cases} e^{-t} & t \ge 0 \\ 0 & t < 0 \end{cases}$$

$$g(t) = e^{-|t|}$$

b) Grafički predstavite realni, imaginarni, amplitudni i fazni spektar transformacija funkcija izračunatih u dijelu zadatka pod a)

c) Simbolički račun u MATLAB-u posjeduje određena ograničenja. Stoga ćete za određivanje Fourierove transformacije signala $f(t)$, $s(t)$ i $x(t)$ datih u pripremi koristiti sljedeći m-fajlove:

```
function F=fourier(t,f,w)
% Funkcija Fourier računa F(jw)=F{f(t)} preko definicionog integrala
% Ulazni parametri funkcije su:
% t - vrijeme                (npr. t=0:0.001:5)
% f - funkcija
% w - kružna učestanost      (npr. w=-20:0.001:20)
dt=t(2)-t(1);
for k=1:length(w)
     F(k)=dt*sum(f.*exp(-i*w(k)*t));
end
```

d) Nacrtati spektre (realni, imaginarni, amplitudni i fazni) za ova tri signala.

e) Na osnovu amplitudnih spektara signala odredite prenosnu funkciju filtra kojim možete izfiltrirati dobiveni signal $x(t)$, tako da na izlazu iz filtra dobijete signal $f_1(t)$, što bliži signalu $f(t)$. Grafički prikažite amplitudni i fazni spektar prenosne funkcije filtra.

f) Po uzoru na dati m-fajl za računanje Fourireove transformacije napišite m-fajl za računanje inverzne Fourierove transformacije.

g) Napisanu funkciju primjenite da odredite i grafički prikažete funkciju $f_1(t)$. Grafički uporedite početnu funkciju $f(t)$ i filtriranjem dobivenu funkciju $f_1(t)$.

Laboratorijska vježba br. 5
DFT i FFT u MATLAB-u

Uvod

U laboratorijskoj vježbi br. 4 smo uvidjeli da su mogućnosti analitičkog računanja Fourierove transformacije vremenski kontinualnog signala, te njeno predstavljanje kao kontinualne funkcije kompleksne promjenjive ograničene, jer MATLAB prilagođen radu sa digitalnim veličinama. U praksi se prije početka obrade kontinualnog signala $f(t)$ prvo vrši vremenska diskretizacija i amplitudna kvantizacija signala tako da se dobiju uzorci signala u trenutcima $f(kT_1)$ koji predstavljaju sekvencu vrijednosti konačne dužine. Ovako dobivena funkcija se naziva zvjezdasta funkcija $f^*(t)$. Pri tome vrijeme uzorkovanja signala T_1 mora biti odabrano tako da se sačuva informacija pohranjena u signalu. Prilikom numeričkog računanja Fourierove transformacije signala, rezultujući spektar je takođe diskretizovan.

Inverzna diskretna Fourierova transformacija definiše odnos između uzoraka funkcije $f_{T_0}(kT_1)$, gdje T_0 period izvedene periodičke funkcije, a T_1 vrijeme uzorkovanja signala ($k = 0,1,\dots,N-1$, $T_1 = \frac{2\pi}{\omega_1}$) i uzoraka spektra $F^*(jm\omega_0)$ zvjezdaste funkcije $f^*(t)$ ($m = 0,1,\dots,N-1$, $\omega_0 = 2\pi T_0$) , dat relacijom (2.5.1).

$$f_{T_0}(kT_1) = \frac{1}{N}\sum_{m=0}^{N-1} F^*(jm\omega_0)W_N^{-mk} \tag{2.5.1}$$

Direktna diskretna Fourierova transformacija je data relacijom (2.5.2).

$$F^*(jm\omega_0) = \sum_{k=0}^{N-1} f_{T_0}(kT_1)W_N^{mk} \tag{2.5.2}$$

Pri čemu je $W_N^{mk} = e^{-j\frac{2\pi}{N}}$. Relacije (2.5.1) i (2.5.2) skraćeno zapisujemo kao:

$$x(k) = \frac{1}{N}\sum_{m=0}^{N-1} X(m)W_N^{-mk} \tag{2.5.3}$$

$$X(m) = \sum_{k=0}^{N-1} x(k)W_N^{mk} \tag{2.5.4}$$

Diskretna Fourierova transformacija transformiše periodički diskretni signal $x(k)$ u periodički diskretni signal $X(m)$, pri čemu su oba signala periodična sa periodom N. Uočavamo da se u relacijama (2.5.3) i (2.5.4) vrše samo operacije sabiranja i množenja (nema integriranja!). Međutim za računanje DFT sekvence sa velikim brojem uzoraka potrebno je izvršiti veliki broj operacija. Zbog toga se u praksi umjesto računanja DFT po definiciji koristi algoritam za brzo nalaženje diskretne fourierove transformacije FFT (*Fast Fourier Transform*).

U programskom okruženju MATLAB-a je implementiran FFT algoritam za nalaženje diskretne Fourierove transformacije signala. FFT algoritam je najefikasniji pri radu sa sekvencama čija je dužina stepen broja 2. Naravno moguće je naći FFT i sekvenci čija dužina ne zadovoljava ovaj uslov.

Ukoliko imamo sekvencu $x(k)$ od 8192 uzorka ($8192 = 2^{13}$) uzorkovanu sa 8000 uzoraka po sekundi, ona odgovara signalu u trajanju nešto dužem od 1 s. Korištenjem FFT algoritma možemo odrediti uzorke $X(m)$, za svako $m \in \mathbb{Z}$. Na osnovu relacije (2.5.4) možemo izračunati vrijednosti uzoraka za $m = 0,1,\dots 8191$, a iz osobina DFT vrijedi:

$$X_{-1} = X_{-1+8192} = X_{8191} \tag{2.5.5}$$

Frekvencija $X(m)$ je data u radijanima po uzorku.

Frekvencija u Hz kojoj odgovara m-ti član sekvence je data kao: $\frac{mF_s}{N}$.

Primjer 5.1:

Neka je zadan signal $x[n]$, za čije članove vrijedi sljedeća relacija:

$$x(n) = \begin{cases} \sin(2\pi * 100t + 2\pi * 100t^2) & 0 \leq n < 8192 \\ 0 \quad , & za\ ostale\ n\ \in \mathbb{N} \end{cases}$$

Uzorkujte signal sa 8000 uzoraka u sekundi. Reproducirajte ovaj signal pomoću zvučne kartice. Izračunajte DFT ovog signala koristeći ugrađeni FFT algoritam. Grafički prikažite amplitudni spektar signala pri čemu će horizontalna osa prikazivati frekvenciju u Hz $(0 - 8000\ Hz)$.

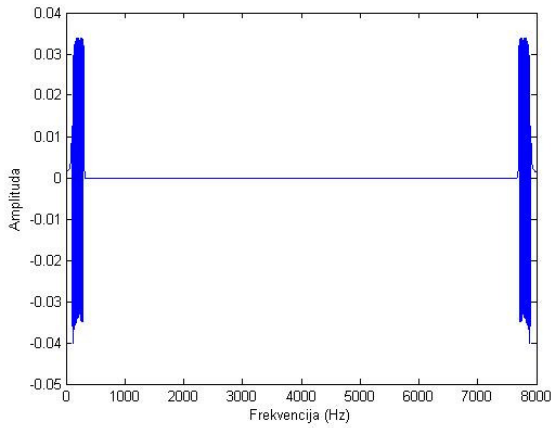
Rješenje:

Ovaj signal je još poznat pod nazivom *chirp* signal. Za generisanje signala i reprodukciju zvuka u radni prostor MATLAB-a unosimo sljedeće komande:

```
>> Fs=8000;
>> N=8192;
>> t=[0:1/Fs:(N-1)/Fs];
>> x=sin(2*pi*100*t + 2*pi*100*t.*t);
>> sound(x,Fs);
```

Za računanje DFT unesite sljedeće komande u radni prostor MATLAB-a:

```
>> m=[0:1:N-1]
>> X=1/N*fft(x,N);
>> w=m*Fs/N;
>> figure
>> plot(w,X)
>> xlabel('Frekvencija (Hz)')
>> ylabel('Amplituda');
```



Slika 2.15 Amplitudni spektar signala x[n] prikazan na frekvencijama od 0 do 8000 Hz

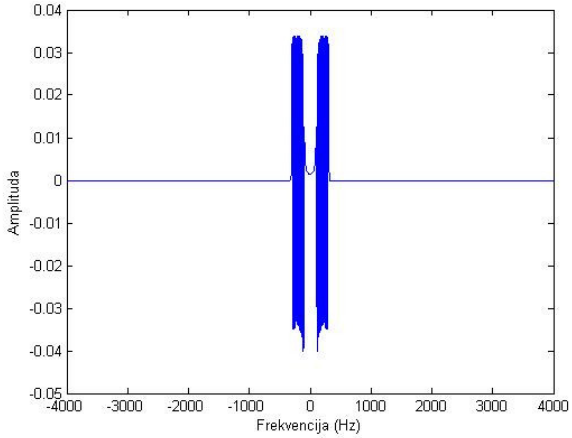
Uočavamo da pored očekivanog frekventnog sadržaja na frekvencijama od 100 do oko 300. Hz, prisutne su i frekventne komponente na frekvencijama 7700 do 7900 Hz. Ovo pojava je uzrokovana svojstvom simetrije DFT i predstavlja komponente sa frekvencijama -100 do -300 Hz.

Primjer 5.2:

Za signal dat u prethodnom primjeru potrebno je nacrtati aplitudni spektar za vrijednosti frekvencija od -4000 Hz do 4000 Hz. Uočite simetričnost amplitudnog spektra.

Rješenje:

```
>> m_n=[-N/2:1:N/2-1];
>> f_n=m_n*Fs/N;
>> X=1/N*fft(x,N);
>> Y=X(1:N/2-1);
>> Z=X(N/2:N);
>> X_n=[Z,Y];
>> plot(f_n,abs(X_n)); xlabel('Frekvencija (Hz)'); ylabel('Amplituda');
```



Slika 2.16 Amplitudni spektar signala x[n] prikazan na frekvencijama od -4000 do 4000 Hz

Funkcija ‘`filter`’ računa odziv LTI sistema opisanog diferentnom jednačinom u sljedećem obliku:

$$a_1 y(n) = b_1 x(n) + b_2 x(n-1) + \cdots + b_M x(n-M+1) - a_2 y(n-1) - \cdots - a_N y(n-N+1)$$

Da bi odredili odziv sistema na pobudu x potrebno je definisati vektor koeficijenata a dužine N i vektor koeficijenata b dužine M i unijeti sljedeću komandu u radni prostor MATLAB-a:

```
>> y=filter(B,A,x);
```

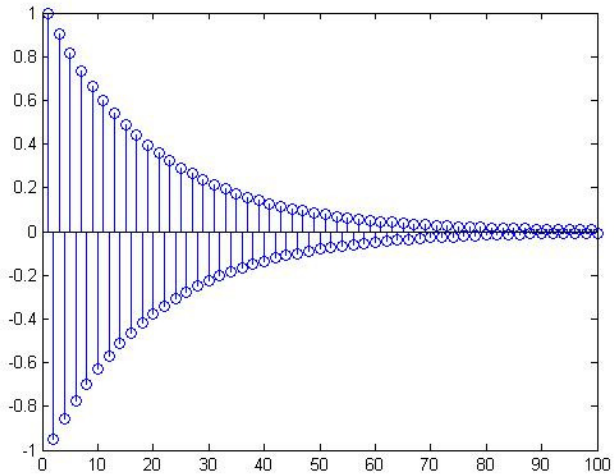
Primjer 5.3:

Odrediti impulsni odziv LTI sistema opisanog diferentnom jednačinom $y(n) = x(n) - 0.95y(n-1)$ i grafički ga prikazati.

Rješenje:

U radni prostor MATLAB-a unesite sljedeće komande:

```
>> x=[1 zeros(1,99)];
>> y=filter([1],[1 0.95],x);
>> stem(y)
```



Slika 2.17 Impulsni odziv sistema opisanog diferentnom jednačinom $y(n) = x(n) - 0.95y(n-1)$

Amplitudna modulacija

Amplitudna modulacija je prvi uspješno implementiran način prenosa komercijalnog radio signala, koji se i danas zadržao u upotrebi. AM se vrši translacijom spektra signala poruke $f_m(t)$ za vrijednost frekvencije ω_0 sadržane u signalu kojim vršimo modulaciju, najčešće $\cos(\omega_0 t)$. Dobiveni modulirani signal je:

$$f(t) = f_m(t)\cos(\omega_0 t) \tag{2.5.6}$$

Spektri modulišućeg i moduliranog signala su povezani relacijom (2.5.7).

$$\mathcal{F}\{f_m(t)\} = F_m(j\omega)\,, \quad \mathcal{F}\{f(t)\} = F(j\omega)$$

$$F(j\omega) = \mathcal{F}\{f_m(t)\cos(\omega_0 t)\} = \frac{1}{2}F_m[j(\omega-\omega_0)] - \frac{1}{2}F_m[j(\omega+\omega_0)] \tag{2.5.7}$$

Za dobijanje originalnog signala iz amplitudno moduliranog signala, potrebno je signal ponovo pomnožiti sa kosinusoidom iste frekvencije. Međutim, samo ovaj postupak obično ne daje željeni rezultat pa je potrebno isfiltrirati tako dobiveni signal.

Unutar Signal Processing Toolbox-a preddefinirane su različite funkcije koje nam računaju koeficijente sistema u ovisnosti o tome kakav frekventni odziv sistema želimo postići. Npr., ukoliko želimo napraviti niskopropusni filter koji će propustiti sve frekvencije ispod 1kHz signala uzorkovanog frekvencijom 8000 Hz to možemo postići sljedećom komandom:

```
>> [B A]=butter(10,0.25);
```

Funkcija ‘`butter`’ kao prvi argument prima dužinu signala M koja određuje red filtera (ovaj filter zahtjeva da je $M = N$), a drugi argument predstavlja frekvenciju odsjecanja kao dio polovine frekvencije uzorkovanja (frekvenciju na kojoj amplituda odzivapada na $\frac{1}{\sqrt{2}}$). U našem slučaju vrijedi $0.25 * \frac{8000}{2} = 1000\ Hz$.

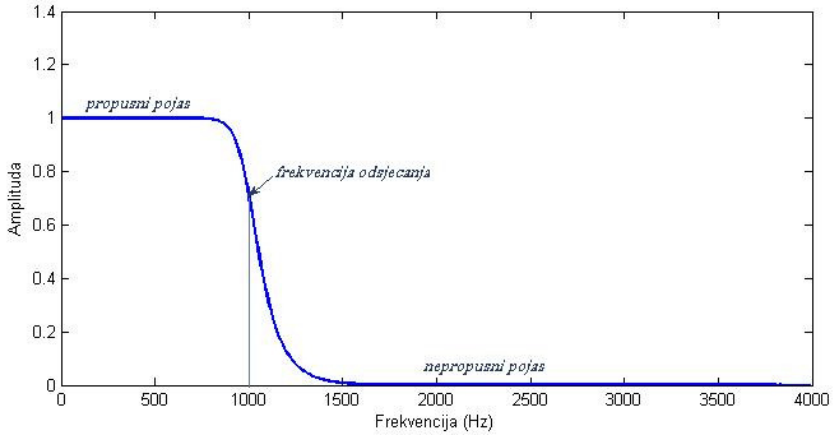
Primjer 5.4:

Grafički prikazati frekventni i impulsni odziv Butterworth-ovog filtra.

Rješenje:

Frekventni odziv filter možemo dobiti pomoću komande ‘`freqz`’ . Unesite u radni prostor MATLAB-a:

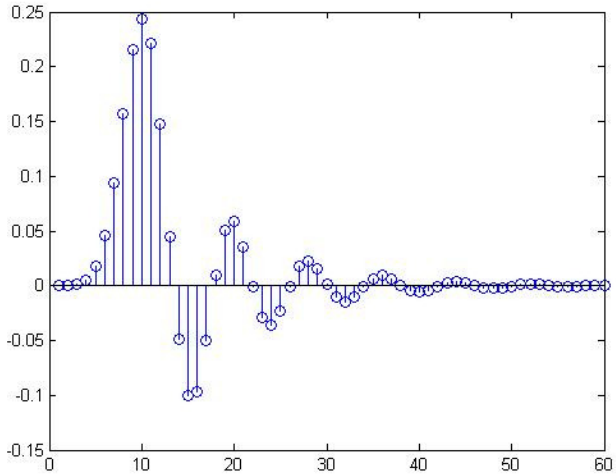
```
>> Fs=8000;
>> [B,A]=butter(10,0.25);
>> [H,W]=freqz(B,A,512);
>> plot(W*Fs/(2*pi), abs(H)), xlabel('Frekvencija (Hz)'),
ylabel('Amplituda');
```



Slika 2.18 Frekventni odziv Butterworth filtera

Za dobijanje impulsnog odziva u radni prostor MATLAB-a unesite sljedeće komande:

```
>> x=[1 zeros(1,59)];
>> y=filter(B,A,x);
>> stem(y)
```



Slika 2.19 Impulsni odziv Butterworth-ovog filtra

Laboratorijski zadatak

Priprema:

Koristeći relaciju (2.5.4) pokažite da je $X(m) = X(m + N)$ za svaku cjelobrojnu vrijednost promjenjive m, pretpostavljajući da je sekvenca $x[k]$ realna.

AM radio

Potrebno je emitovati signal radio stanice. Opseg frekvencija na kojima je dozvoljeno emitovati signal je od 5 kHz do 10 kHz, ukupno 5 kHz. Međutim signal koji želimo emitovati je govorni signal i on sadrži frekvencije u opsegu 50 Hz do 10 kHz. Za razumljivi ljudski govor dovoljno je prenositi frekvencijsko područje 300-3400 Hz. Izvršite modulaciju signala tako da se može emitovati datim kanalom, a zatim i demodulaciju tako da se može reprodukovati. Snimite govorni signal u trajanju ne dužem od 5 sekundi kao *govor.wav* i frekvencijom uzorkovanja 8 kHz i ponesite ga sobom na laboratorijsku vježbu.

OBAVEZNO ponijeti slušalice na laboratorijske vježbe!!!

Rad u laboratoriji:

a) Napravite amplitudnu modulaciju signal cvrkuta (chirp) tako što ćete ga pomnožiti a sinusoidom frekvencije 1kHz. Grafički prikažite amplitudni spektar modulisanog signala na frekvencijama -4000 Hz do 4000 Hz. Da li je ovaj signal dozvoljeno emitovati putem opisane radio stanice? Reprodukujte dobiveni signal na vašoj zvučnoj kartici.
b) Modulirani signal možemo demodulirati tako što ćemo ga ponovo pomnožiti sa sinusnim signalom frekvencije 1kHz. . Grafički prikažite amplitudni spektar demodulisanog signala na frekvencijama -4000 Hz do 4000 Hz. Reprodukujte dobiveni signal na vašoj zvučnoj kartici. Da li postoji razlika između originalnog i demoduliranog signala?
c) Primjenite Butterworth-ov filter (npr. reda 5) da poboljšate kvalitet dobivenog signala. Grafički prikažite amplitudni spektar izfiltriranog signala na frekvencijama 4000 Hz do 4000 Hz.
d) Napravite amplitudnu modulaciju vašeg govornog signala tako da se može emitovati pute opisane radio stanice. Grafički prikažite amplitudni spektar modulisanog signala. Reprodukujte dobiveni signal na vašoj zvučnoj kartici
e) Izvršite demodulaciju signala. Grafički prikažite amplitudni spektar modulisanog signala. Reprodukujte dobiveni signal na vašoj zvučnoj kartici
f) Upotrijebite Butterworth-ov filter (ili neki drugi) za poboljšanje kvalitete signala. Više informacija o raspoloživim filterima možete naći u `help`-u.

Laboratorijska vježba br. 6
Uzorkovanje i aliasing

Uvod

Signali koji se javljaju u prirodi su analogni signali, međutim takvi signali nisu podesni za obradu na digitalnim računarima. Stoga se pristupa digitaliziciji signala postupkom AD konverzije. Ovaj postupak podrazumjeva diskretizaciju signala po vremenu i kvantizaciju signala po amplitudi. Pri tome je potrebno osigurati očuvanje poruke sadržane u signalu. Diskretizaciju signala po vremenu vršimo impulsnom modulacijom signala, što podrazumijeva pamćenje vrijednosti signala u trenutku $t = kT$, $k \in \mathbb{Z}$, gdje je T period uzorkovanja.

Da bi kontinualni signal signal $x(t)$ bio jedinstveno određen svojim uzorcima $x(kT)$, $k \in \mathbb{Z}$ i $T = 2\pi/\omega_0$ potrebno je da ima ograničen spektar, tj. $X(j\omega) = 0$, za $|\omega| > \omega_M$ i da vrijedi:

$$\omega_0 > 2\omega_M \tag{2.6.1}$$

Primjer 6.1:

Posmatrajte analogni signal zadan formulom $x(t) = \sin(2\pi f t^2)$. Signal dobiven uzorkovanjem ovog signala je $y(t) = \sin(2\pi f (kT)^2)$. Potrebno je u MATLAB-u kreirati signal $y(t)$ uzorkovan frekvencijom 8000 uzoraka u sekundi, koji traje 10 sekundi i uzima vrijednosti frekvencija od 0 do 12 kHz.

Poslušajte dobiveni signal i objasnite opažene efekte aliasinga.

Rješenje:

Frekvencija signala $x(t)$ je ovisna o vremenu i mijenja se po zakonu: $f_t = 2ft$. Za generisanje traženog signala u radni prostor MATLAB-a unesite sljedeće komande:

```
>> Fs=8000;
>> t=0:1/Fs:10;
>> f=12000/(2*10);
>> y=sin(2*pi*f*t.*t);
>> plot(t,y);
>> soundsc(y,Fs);
```

Primjer 6.2:

Razmotrite signal cvrkutanja moduliran pilastom funkcijom opisan sljedećom relacijom:

$$x(t) = g(t)\sin(2\pi f t^2)\,, \forall t \in (0, T)$$

Gdje je $g(t) = 1 - \left|t - \frac{T}{2}\right| / \frac{T}{2}$. Radi jednostavnijeg i efikasnijeg računanja DFT korištenjem FFT algoritma koji je implementiran u MATLAB-u, obradu signala ćemo vršiti na 8192 uzorka signala, uzorkovanih frekvencijom 8000 uzoraka u sekundi, otprilike 1 sekunda zvučnog signala. Definišite u MATLAB-u ovakav signal koji će sadržavati sve frekvencije od 0 do 2500 Hz. Reprodukujte zvučni signal. Uočavate li efekte aliasinga? Objasnite zašto da/ne.

Korištenjem ‘fft’ komande odredite DFT signala. Grafički prikažite amplituni spektar za frekvencije -4000 Hz do 4000 Hz. Horizontalna osa treba predstavljati frekvenciju u Hz.

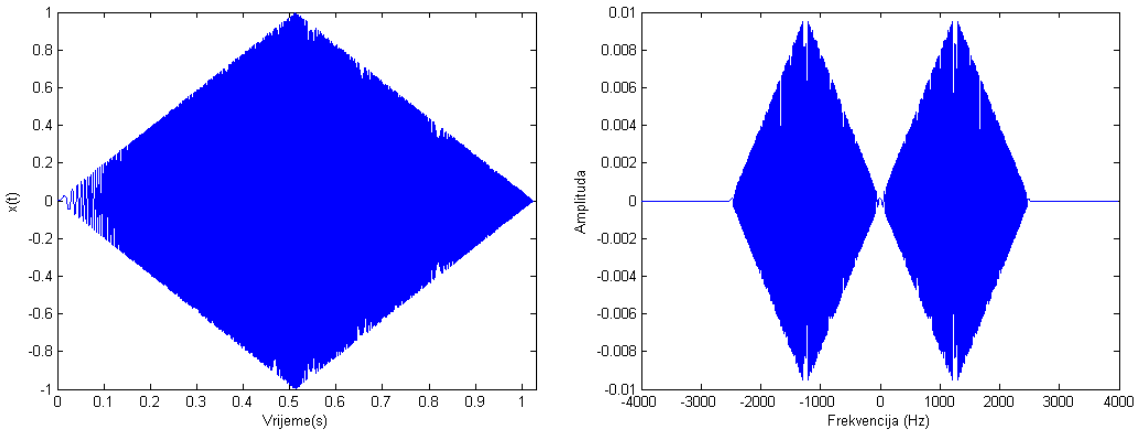
Rješenje:

Odabir frekvencije signala je objašnjen u prethodnom primjeru. Za generisanje traženog signala potrebno je u radni prostor MATLAB-a unijeti sljedeće komande:

```
>> Fs=8000;
>> N=8192;
>> t=0:1/Fs:(N-1)/Fs;
>> f=2500/(2*max(t));
>> g=1-abs(t-N/Fs/2)./(N/Fs/2)
>> x=g.*sin(2*pi*f*t.*t);
>> figure
>> plot(t,x)
>> xlabel('Vrijeme(s)'), ylabel('x(t)');
>> axis([0 1.03 -1 1]);
>> soundsc(x,Fs);
```

Za prikaz amplitudnog spektra date funkcije koristimo kod kao u prethodnoj laboratorijskoj vježbi:

```
>> m =[-N/2:1:N/2-1];
>> w =m*Fs/N;
>> X=1/N*fft(x,N);
>> Y=X(1:N/2-1);
>> Z=X(N/2:N);
>> X=[Z,Y];
>> plot(w,abs(X))
>> xlabel('Frekvencija (Hz)'), ylabel('Amplituda');
```



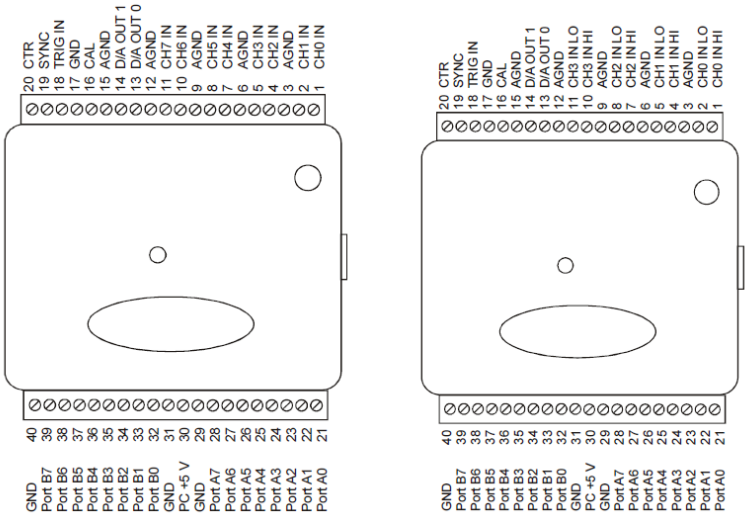
Slika 2.20 Signal x(t) (lijevo), amplitudni spektar signala x(t) (desno)

- Izmjenite signal $x(t)$, tako da sadrži frekvencije u opsegu od 0 do 5000 Hz.
- Reprodukujte signal na zvučnoj kartici.
- Korištenjem ‘fft’ komande odredite DFT signala. Grafički prikažite amplituni spektar za frekvencije -4000 Hz do 4000 Hz. Horizontalna osa treba predstavljati frekvenciju u Hz.
- Objasnite uočene efekte aliasinga.

Ponekad je potrebno izvršiti obradu realnih signala, koje nije moguće matematički egzaktno opisati. U tom cilju je potrebno izvršiti akviziciju takvih signala iz realnog svijeta u MATLAB ili Simulink i potom vršiti potrebnu analizu i obradu. Za izvršavanje ovog zadatka, potreban je specifičan hardver u vidu DAQ kartica (Data Acquisition Card) koji služi za digitalizaciju kontinualnog signala (vrši uzorkovanje i kvantizaciju signala). Razmotriti ćemo upotrebu jedne takve DAQ kartice za akviziciju signala, RedLab-1208FS, prikazane na slici.

Slika 2.21 RedLab-1208FS DAQ kartica i USB konektor za povezivanje sa računarom

Ova kartica posjeduje osam odvojenih ulaznih kanala (4 ukoliko se koriste u diferencijalnom spoju) koji mogu raditi sa signalima u opsezima: $\pm 20\ V$, $\pm 10\ V$, $\pm 5\ V$, $\pm 4\ V$, $\pm 2.5\ V$, $\pm 2.0\ V$, $\pm 1.25\ V$ i $\pm 1.0\ V$ i dva analogna izlaza sa 12-bitnom konverzijom (opseg amplitude napona na izlazu je $0 - 4.096\ V$). Posjeduje i 16 izvoda koji mogu služiti kao digitalni ulazi ili izlazi(Port A i Port B). Maksimalna brzina uzorkovanja signala je 50kHz, s tim da se maksimalna brzina uzorkovanja dijeli sa brojem kanala sa kojih je potrebno očitati vrijednosti. Sadrži i izvod za jedan 32-bitni brojač. Komunikacija sa računarom se ostvarije korištenjem USB kabla.



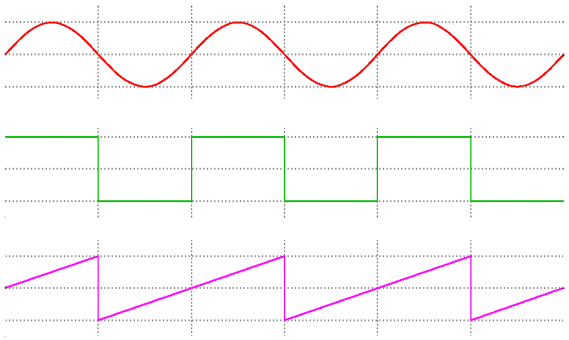
Slika 2.22 Raspored pinova: 8 nezavisnih ulaznih kanala (lijevo), 4 ulazna kanala u diferencijalnom spoju (desno)

Akvizicija signala u Simulink-u je veoma jednostavna. Zahtjeva samo da se unutar Data Acquisiton Toolbox-a odabere blok za prikupljanje podataka (*Analog Input*, *Digital Input*) ili slanje podataka (*Analog Output*, *Digital Output*). Unutar odgovarajućeg bloka je potrebno odabrati frekvenciju uzorkovanja signala, kanal sa kojeg vršimo očitavanje (ili na koji šaljemo signal) i način spajanja. Ostali dio modela za obradu signala gradimo kao da se radi o signalu namjenski kreiranom u Simulinku.

Laboratorijski zadatak

Priprema:

Laboratorijski zadatak je takmičarskog karaktera i radi se u grupama od dva člana. U okviru ove laboratorijske vježbe ćete raditi u parovima. Vaš zadatak je da kreirate funkciju koja se dobije sumiranjem tri funkcije od kojih svaka može biti nekog od navedenih oblika (pri tome oblici signala mogu biti biti i isti , npr. možete imati dvije sinusoide i jednu povorku pravougaonih impulsa, ili tri signala pile) frekvencije do 2000Hz.



Slika 2.23 Dozvoljeni oblici signala

Za dobivenu funkciju grafički prikažite amplitudni spektar. Cilj je napraviti funkciju sa čijeg je amplitudnog spektra što teže odrediti oblik i frekvenciju korištenih funkcija. Amplitudni spektar prikazati u opsegu frekvencija od -250 do 250 Hz.

Potrebno je napraviti i Simu link model koji pomoću tri analogna ulaza DAQ kartice RedLab-1208 FS vrši akviziciju tri signala, sabira ih i crta njihov amplitudni spektar. Signale ćete generisati pomoću tri generatora funkcija i dovesti na analogne ulaze akvizicione kartice.

OBAVEZNO ponijeti slušalice na laboratorijske vježbe!!!

Rad u laboratoriji:

a) Pomoću tri generatora signala i kartice za akviziciju RedLab-1208FS ćete generisati i prikupljati signale. Unutar Simulinka ćete sumirati signale i crtati njihov amplitudni spektar. Ostali timovi će pokušati da pogode signale od kojih se sastoji vaš signal.
b) Ukupan broj bodova koji se može dobiti za ovu vježbu pripada timu sa najviše pogođenih signala.

Laboratorijska vježba br. 7
Z-transformacija

Uvod

Z-transformacija predstavlja generalizaciju Fourierove transformacije diskretnog signala. Z-transformacija preslikava sekvencu $f[n]$ u kontinualnu funkciju kompleksne promjenjive z . Jednačina (2.7.1) daje vezu između sekvence i Z-transformacije sekvence. Inverzna Z-transformacija diskretnog signala $f[n]$ je data jednačinom (2.7.2).

$$F(z) = \sum_{n=-\infty}^{\infty} f(n) z^{-n} \tag{2.7.1}$$

$$f[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(e^{j\omega}) e^{j\omega n} d\omega \tag{2.7.2}$$

Primjer 7.1:

Odrediti Z-transformaciju sekvence:

$$f_n = \begin{cases} nT_s & n \geq 0 \\ 0 & n < 0 \end{cases}$$

Rješenje:

Primjetite da je zadana sekvenca kauzalna. Za računanje Z-transformacije diskretnog signala, možemo upotrijebiti komandu ‘`ztrans`’. Unesite sljedeće naredbe u radni prostor MATLAB-a:

```
>> syms n Ts
>> f=n*Ts*heaviside(n);
>> Z=ztrans(f)

Z =

(Ts*z)/(z^2 - 2*z + 1)
>> pretty(simplify(Z))

    Ts z
  --------
          2
  (z - 1)
```

Mogli smo računati Z-transformaciju po definiciji, odnosno računati direktno sumu iz jednačine (2.7.1) koristeći naredbu ‘`symsum`’. U tu svrhu unesite sljedeće naredbe u radni prostor MATLAB-a:

```
>> syms n Ts z
>> Z=symsum(n*Ts*z^(-n),n,0,inf);
```

Rezultat koji daje MATLAB možemo predstaviti u sljedećoj formi:

```
>> pretty(simplify(Z))
```

```
          /   Ts z                          \
piecewise| -------- if 0 < |z| (|z| - 1)  |
          |        2                        |
          \ (z - 1)                         /
```

Primjer 7.2:

Odredite sekvencu čija je inverzna Z-transformacija data relacijom:

$$X(z) = \frac{z}{(z-1)^2(z-0.5)}$$

Grafički prikazati dobivenu sekvencu.

Rješenje:

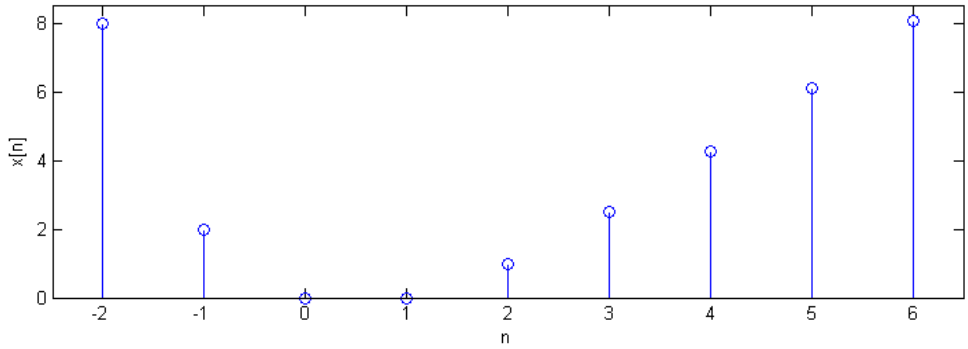
Računanje inverzne Z-transformacije omogućava naredba ‘`iztrans`’. Unesite u radno okruženje sljedeće komande:

```
>> syms z
>> X = z/((z-1)^2*(z-0.5));
>> x=iztrans(X);
>> pretty (x)

          / 1 \n
  2 n + 4 | - |   - 4
          \ 2 /
```

Funkcija ‘`ezplot`’ bi iscrtala kontinualnu funkciju $x(t)$. Predstavljanje sekvence $x[n]$ ostvarujemo pomoću sljedećeg koda:

```
>> t=-2:1:6;
>> y=subs(x,t);
>> stem(t,y)
>> axis([-2.5 6.5 0  8.5]),xlabel('n'),ylabel('x[n]')
```



Slika 2.24 Dobivena sekvenca x[n]

Mnogi realni sistemi koje izučavamo, npr. filteri, imaju dinamiku koja se može opisati diferencijalnom jednačinom sa konstantnim koeficijentima. Međutim takve jednačine nisu podesne za obradu na digitalnom računaru, jer zahtijevaju rad sa kontinualnim signalima, a MATLAB je alat specijaliziran za rad sa digitalnim signalima. Zbog toga se pri određivanju odziva sistema često pristupa numeričkom rješavanju diferencijalne jednačine koja ga opisuje.

Posmatrajmo sistem opisan diferencijalnom jednačinom:

$$\frac{d}{dt}y(t) = ay(t) + x(t), \qquad y(0) = \alpha \tag{2.7.3}$$

Ukoliko uzorkujemo ulazni signal sa dovoljno malim periodom T_s, $x_k = x(kT_s)$, $k \geq 0$, tada će sekvenca $\{y_k\}$, definirana izrazom:

$$\overbrace{\frac{y_{k+1}-y_k}{T_s}}^{\frac{dy}{dt}} = ay_k + x_k, \qquad y(0) = \alpha, \qquad k \geq 0 \tag{2.7.4}$$

težiti sekvenci uzorkovanih vrijednosti funkcije $y(t)$, tj. za T_s dovoljno malo će vrijediti: $y_k \approx \{y(kT_s)\}$.

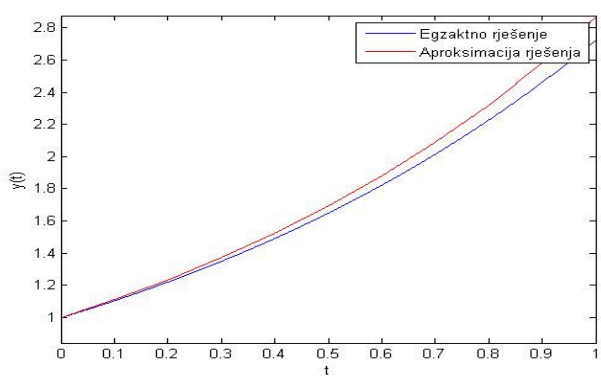
Primjer 7.3:

Koristeći analitički i numerički metod rješavnja odredite rješenje diferencijalne jednačine: $\dot{y} = y, \quad y(0) = 1$ i grafički ih uporedite.

Rješenje:

Koristeći simbolički račun pomoću naredbe 'dsolve' možemo odrediti analitičko rješenje jednačine:

```
>> ya=dsolve('Dya=ya','ya(0)=1');
>> ezplot(ya,[0 1]),xlabel('t'),ylabel('y(t)');
>> Ts=0.1;
>> yd(1)=1;
>> for i=2:11
yd(i)=yd(i-1)/(1-Ts);
end
>> hold on, plot(0:Ts:1,yd,'r');
>> legend('Egzaktno rješenje','Aproksimacija rješenja');
```



Slika 2.25 Poređenje analitički i numeričkim postupkom dobivenog rješenja diferencijalne jednačine y’=y

Primjer 7.4:

Koristeći analitički i numerički metod rješavnja odredite i grafički uporedite dobivena rješenja diferencijalne jednačine drugog reda:

$$\ddot{y} + \dot{y} + y = 0, \qquad y(0) = 1\,, \dot{y}(0) = 0$$

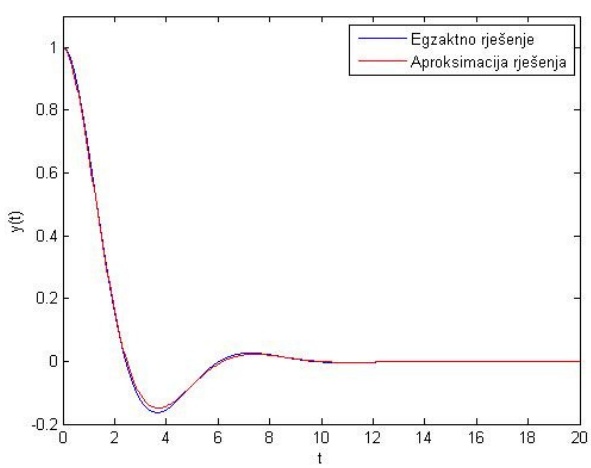
Rješenje:

Definišimo vektor stanja $x(t)$ kao: $\vec{x}(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}$. Vektor izvoda stanja je sada: $\dot{\vec{x}}(t) = A\vec{x} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\vec{x}$, pri čemu vrijedi $\vec{x}(0) = \vec{\alpha}$. Sada je postavljenu jednačinu moguće zapisati kao:

$$\frac{\vec{x}_{k+1} - \vec{x}_k}{T_s} = A\,\vec{x}_k$$

Odakle slijedi: $\vec{x}_{k+1} = (I - T_s A)^{-1}\,\vec{x}_k$, uz uslov: $\vec{x}(0) = \vec{\alpha}$. Da bi sproveli opisani postupak u MATLAB-u potrebno je u radno okruženje unijeti sljedeći kod:

```
>> A=[ 0  1; -1 -1];
>> I=[ 1  0;  0 1];
>> x0=[1; 0];
>> Ts=0.05;
>> N=20/Ts;
>> x=x0;
>> ya(1)=x(1);
>> for i=1:N
    x=inv(I-A*Ts)*x;
    ya(i+1)=x(1);
end;
>> yd=dsolve('D2yd + Dyd + yd = 0','yd(0)=1','Dyd(0)=0');
>> ezplot(yd,[0 20]), hold on, plot([0:N]*Ts,ya,'r'), axis([0 20 -0.2 1.1]);
>> xlabel('t'),ylabel('y(t)')
>> legend('Egzaktno rješenje','Aproksimacija rješenja');
```



Slika 2.26 Poređenje aproksimacijom dobivenog rješenja diferencijalne jednačine i rješenja dobivenog korištenjem naredbe 'dsolve'

Laboratorijski zadatak

Rad u laboratoriji:

a) Korištenjem simboličkog računa u MATLABU izračunajte Z-transformaciju sljedećih diskretnih signala:

$$f_n = \begin{cases} n^2 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

$$g_n = a^{-n}$$

b) Odrediti i grafički predstaviti signal čija je Z-transformacija jednaka:

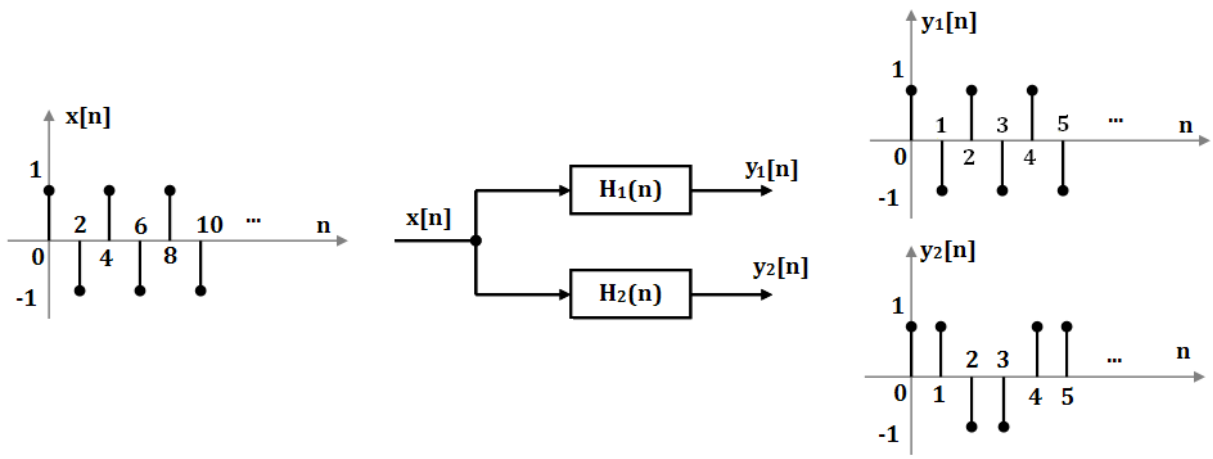
$$X(z) = \frac{z^2(z-1.2)(z+1)}{(z-0.5+j0.7)(z-0.5-j0.7)(z-0.8)}$$

c) Odredite odziv na jediničnu step pobudu sistema čija je prenosna funkcija:

$$G(s) = \frac{1}{s^2+s+1}$$

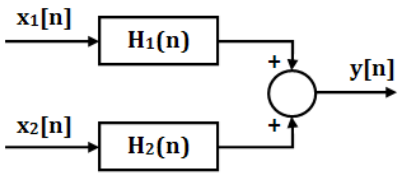
- Pomoću naredbe ‘dsolve’
- Korištenjem aproksimacije derivacije.

d) Diskretni sistem na slici pobuđen signalom na ulazu, odaziva se svojim izlazima kao na slici:



Slika 2.27 Ulaz i izlazi diskretnog sistema

Odrediti i grafički prikazati odziv sistema na slici $y[n]$ ako se na ulaz dovodi impulsna funkcija $x_1[n]$ i jedinična step funkcija $x_2\ [n]$:



Slika 2.28 Realizirani sistem

e) Relizirati sitem u Simulinku, te posmatrati odziv na rampu, sinusnu i eksponencijalnu pobudu. Da li izlaz sistema prati dinamiku promjene ulaza?

ZAKLJUČAK

Praktikum za laboratorijske vježbe iz “Signala i sistema” se bavi osnovnim metodama akvizicije i obrade signala, kao i nekim u praksi često susretanim problemima u ovom području. Za ovu namjenu je korišten MATLAB softverski paket koji uključuje i razvojno okruženje Simulink-a.

Upoznavanje sa radnim okruženjem, kao i pregled osnovnih komandi i mogućnosti obrade signala koje MATLAB i Simulink nude su izloženi u prvom poglavlju. Unutar ovog poglavlja su kroz jednostavne primjere demonstrirani neki od mogućih načina akvizicije signala, vršenja proračuna i vizualnog prikaza dobivenih rezultata raspoloživih unutar MATLAB-a. Potom su izloženi načini predstavljanja sistema preko njegove blokovske strukture i vršenja simulacija u kontinualnom i diskretnom vremenu u Simulink-u.

U sklopu drugog poglavlja se nalazi sedam laboratorijskih vježbi koje se bave obradom signala u vremenskom i frekventnom domenu. Svaka laboratorijska vježba se sastoji iz uvoda, u kojem su izloženi osnovni teorijski koncepti i komande u MATLAB-u potrebne za uspješno obavljanje laboratorijske vježbe i laboratorijskog zadatka. Laboratorijski zadatak obuhvata i pripremu koju je student obavezan ponijeti sa sobom na laboratorijsku vježbu i zadatak koji će biti obrađen u toku rada u laboratoriji. Samostalan rad, u vidu pripreme za laboratorijsku vježbu osigurava adekvatnu pripremljenost studenta za rad u laboratoriji.

LITERATURA

[1] **Melita Ahić-Đokić**, *"Signali i sistemi"*, Elektrotehnički fakultet Sarajevo, 2010.

[2] **Alan V. Oppenheim, Alan S. Willsky**, *“Signals and systems”*, Prentice Hall, 1997.

[3] **Alan V. Oppenheim, Roland W. Schafer**, *“Discrete-time Signal and Processing”*, Prentice Hall, 1999.

[4] **The MathWorks, Inc**., “*Getting Started with MATLAB*”, http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf , 11.12.2009. godine

[5] **Nasser Kehtarnavaz, Philipos Loizou, Mohammad Rahman**, *"An Interactive Approach to Signals and Systems Laboratory,"* Connexions, http://cnx.org/content/col10667/1.10/, 12. 12. 2010. godine.

[6] **Adnan Tahirović, Mujo Hebibović,** *“MATLAB u teoriji automatskog upravljanja – praktikum za laboratorijske vježbe ”,* Elektrotehnički fakultet u Sarajevu, 2002.