Zavod za elektroničke sustave i obradbu informacija
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu

Upute za laboratorijske vježbe iz signala i sustava

Tomislav Petković, Zvonko Kostanjčar,
Marko Budišić, Branko Jeren

Zagreb, svibanj 2006.

Sadržaj

Upute za laboratorijske vježbe iz signala i sustava

Uvod

Laboratorijske vježbe iz Signala i sustava zamišljene su da približe studenta problematici analize i simulacije sustava. Sve vježbe se izvode na računalu, a koristi se programski sustav MATLAB. Osim što posjeduje mogućnost izvođenja raznih jednostavnih i izuzetno složenih matematičkih operacija, MATLAB ima i modul Simulink koji je zamišljen kao alat za brzo i jednostavno simuliranje raznih sustava.

Prije dolaska na pojedinu vježbu student je obavezan napraviti pripremu, tj. detaljno proučiti sustav koji se simulira na vježbi i riješiti pripremne zadatke. Nakon svake vježbe potrebno je sastaviti izvještaj koji se predaje na sljedećim laboratorijskim vježbama i to ispisan na papiru (izvještaj može biti pisan i rukom). U izvještaju se uz predstavljanje rezultata analize simuliranog sustava obavezno navode i komentari rezultata.

Kako je vrijeme za izradu laboratorijskih vježbi ograničeno a kako uvijek postoji dosta zanimljivih detalja koji i nisu presudni za razumijevanje izloženog dio zadataka i objašnjenja je označen s *Za one koji žele znati više*. Takve dijelove je svakako korisno pročitati, no oni nisu nužno potrebni za dovršetak pojedine laboratorijske vježbe.

Ukoliko je student opravdano spriječen prisustvovati nekoj od vježbi u dogovoru s asistentima bit će organizirane nadoknade u okviru kojih će se moći nadoknaditi propuštene vježbe.

1. Vježba 1. – Pripreme za laboratorijske vježbe

1.1. Uvod

Gotovo svaka odluka koju donesete će imati neke posljedice koje nije moguće odmah uočiti. Stoga se prije donošenja važnih odluka uobičajeno vrše razne analize mogući ishoda. Iako je u većini slučajeva gotovo nemoguće napraviti potpunu analizu temeljem koje se dolazi do pouzdanih zaključaka, u tehničkim disciplinama analiza sustava gotovo uvijek daje neki analitički model koji pruža detaljniji uvid u bitna svojstva nekog sustava.

Tradicionalni pristup analizi sustava se temelji na eksperimentiranju te razvoju i analizi fizičkih modela pa se shodno tome podjednako oslanja na *vještinu i iskustvo* inženjera i na znanstveno razumijevanje problema. No kako se sve više napora ulaže u razvijanje bolje teoretske podloge i kvalitetnijih teoretskih modela mogućnosti kvalitetnog modeliranja i simulacije koje se temelje na znanstvenom razumijevanju problema postaju sve važnije.

Mogućnosti primjene novih modela bile bi nemoguće bez velikih mogućnosti i niskih troškova novih računalnih tehnologija. Modeliranje procesa i simulacija tipično zahtijevaju složena računanja i zahtjevne tehnike vizualizacije što ne bi bilo izvedivo bez digitalnih računala. Prednosti složenih računalnih simulacija su brzo dobivanje rezultata s velikom količinom detalja čime dobivamo na realnosti modela.

1.1.1. Modeliranje

Prvi korak analize nekog sustava je stvaranje modela. Pri tome model može biti bilo fizički bilo matematički. Fizički modeli (npr. makete brodova ili aviona) se obično koriste kada matematičkim modelom ne možemo obuhvatiti cjelokupnu složenost sustava ili kao potvrda ispravnosti matematičkog modela. U okviru ovih laboratorijskih vježbi bavimo se samo matematičkim modelima.

Određivanje apstraktnog modela i realizacija odgovarajućeg modela u nekom realnom sustavu naziva se modeliranjem.

Samo modeliranje je proces pojednostavljivanja stvarnosti kako bi mogli bolje razumjeti kakve učinke imaju pojedine akcije. Analizom modela tada možemo stvoriti sliku o sustavu, identificirati bitne značajke sustava te napraviti što-ako analizu s ciljem predviđanja ponašanja sustava. Sam model je pri tome *samo pomoć* pri donošenju odluka – model *ne donosi odluke* umjesto nas.

1.1.2. Simuliranje

Jednom kada smo odredili model moramo pristupiti njegovoj analizi ako želimo steći uvid u svojstva sustava. Pri tome obično ispitujemo kako će se

ponašati model u raznim situacijama koje pak mogu odgovarati odlukama koje donosimo. Sam postupak se naziva simuliranje.

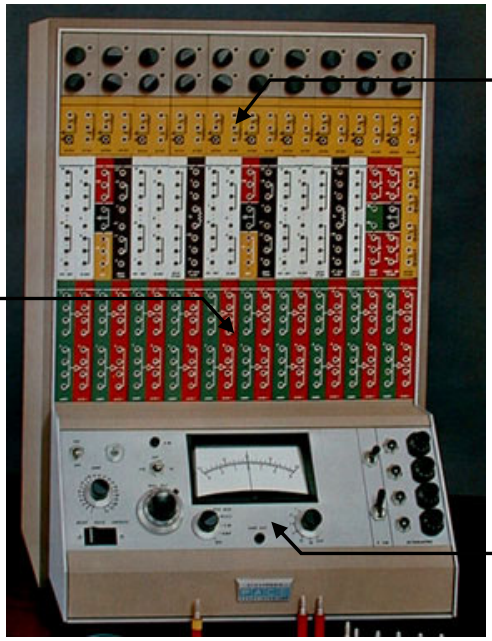
Točno ili približno ponavljanje procesa u sustavu na analognom modelu naziva se simulacijom ili simuliranjem.

1.1.3. Dinamički sustavi i analogna računala

Analizom raznih složenih dinamičkih sustava u većini slučajeva dolazimo do pojednostavljenog modela koji se temelji na diferencijalnim jednadžbama. Tako analizom nekog mehaničkog sustava i neke električne mreže dolazimo do istog matematičkog opisa. Pri tome se naponi i struje u granama mreže vladaju kao pomaci, brzine i akceleracije u mehaničkom sustavu. Za svaka dva sustava koja daju jednaki skup diferencijalnih jednadžbi kažemo da su analogni.

Promatramo li neki skup linearnih diferencijalnih jednadžbi znamo da ga možemo realizirati kao električnu mrežu. Ta električna mreža je jedna realizacija apstraktnog sustava. Kako je električnu mrežu relativno jednostavno ostvariti prvi složeniji modeli sustava su rađeni upravo kao električne mreže.

Štoviše, osim električnih mreža sastavljenih tako da odgovaraju samo jednom sustavu izrađivane su i opće mreže gdje se odgovarajućim spajanjima elemenata sastavljala željena diferencijalna jednadžba. U tom slučaju govorimo o analognom računalu. Analogno računalo se sastojalo od operacijskih pojačala, multiplikatora, potenciometara, generatora funkcija i komparatora. Analogna računala su se koristila za simulaciju dinamičkih sustava 50-tih, 60-tih i 70-tih godina, no pojavom mikroračunala gube na važnosti. Danas se gotovo sve simulacije vrše isključivo na digitalnim računalima.



1.2. Priprema

Na prvoj vježbi ćemo raspravljati o analizi sustava, a posebno o modeliranju i simuliranju. Za pripremu razmislite o metodama opisa bilo kakvih sustava te o razlikama između modeliranja i simuliranja.

1.3. Izvještaj

Za uvodnu vježbu nije potrebno pripremiti izvještaj.

2. Vježba 2. – Upoznavanje s MATLAB-om i Simulink-om

2.1. Priprema

Ako niste upoznati s programskim sustavom MATLAB pročitajte "Kratke upute za korištenje MATLAB-a" koje se mogu pronaći na stranicama predmeta (http://sis.zesoi.fer.hr/vjezbe.html#dodatci).

2.2. Prijava na računalo

Nakon paljenja računala i podizanja operativnog sustava potrebno je upisati korisničko ime i zaporku. Za laboratorijske vježbe iz Signala i sustava koristite korisničko ime `student` i zaporku `student`, a za domenu odaberite `LSS`.

Nakon prve prijave na sustav morate napraviti svoj direktorij u kojeg ćete spremati sve napisane programe i rezultate vježbi. Svoj direktorij kreirajte na mrežnom disku `Z:` u direktoriju `Z:\SIS`, a imenujte ga prema prvom slovu imena te prezimenu. Na primjer, za studenta Signalka Sustavića radni direktorij bi bio `Z:\SIS\ssustavic`.

2.3. Upoznavanje s MATLAB-om

U sklopu vježbi se koristi programski paket MATLAB. Svi podaci u MATLAB-u tretiraju se kao matrice čije dimenzije nije potrebno čuvati kao posebne varijable. Čak se i skalarne veličine pohranjuju kao matrice dimenzije 1×1. Zbog takve transparentne podrške računanju s matricama MATLAB je idealan alat za razvoj raznih algoritama koji se jednostavno prikazuju i opsuju koristeći matrični i/ili vektorski zapis. Na računalima su dostupne dvije inačice MATLAB-a, MATLAB R13 i MATLAB 4.0. Na ovim vježbama koristimo MATLAB R13 kojeg možete pokrenuti izravno s radne površine ili odabirom stavke `START→ Programs→Matlab→Matlab65` iz izbornika.

MATLAB se najčešće koristi kao interaktivni jezik – *interpreter*. Nakon ulaska u program, kao i nakon svake izvedene naredbe, pojavljuje se oznaka za unos oblika » iza koje se nalazi kursor. To označava da MATLAB očekuje unos nove naredbe. Svaka naredba mora završiti tipkom *Enter* – u nastavku teksta oznaka **<ENT>.**

Nakon pokretanja MATLABA pozicionirajte se u svoj radni direktorij.

```
» cd Z:\SIS\ssustavic <ENT>        % naravno, zamijenite ssustavic sa svojim
                                   % direktorijem
» ls <ENT>                         % vaš direktorij bi naravno trebao biti prazan

.   ..
```

Za svaki operator ili funkciju, kao i za čitave programske pakete u MATLAB-u postoje detaljne upute *on line*. Unutar MATLAB ljuske do njih se dolazi korištenjem naredbe `help`.

```
» help                  % daje popis svih programskih paketa
...
» help ops              % daje popis svih operatora
...
» help cos              % ispisuje kratku pomoć za naredbu cos

 COS    Cosine.
    COS(X) is the cosine of the elements of X.

 Overloaded methods
    help sym/cos.m
```

Pomoć unutar ljuske je zamišljena kao kratki podsjetnik što koja naredba radi i koji su joj ulazni parametri. Detaljnu pomoć s primjerima daje *HelpDesk* koji se pokreće zadavanjem naredbe `helpdesk` ili izborom `Help→MATLAB Help` unutar glavnog izbornika. MATLAB *HelpDesk* omogućuje napredna pretraživanja, a osim jednostavne pomoći sadrži i kompliciranije primjere korištenja pojedinih funkcija. Osim MATLAB *HelpDeska* svu MATLAB dokumentacija u elektroničkom obliku (HTML) možete pronaći na zavodskom serveru http://matlab.zesoi.fer.hr/.

2.3.1. Matrice i vektori

Matrice definiramo unošenjem u MATLAB ljusci. S tako definiranim matricama jednostavno računamo na uobičajeni način:

```
» A = [1 2 3; 4 5 6]; <ENT>          % dva retka i tri stupca
» B = [1 2; 3 4; 5 6]; <ENT>         % tri retka i dva stupca
» C = A * B <ENT>                    % za množenje matrice moraju biti ulančane

C =                                  % rezultat ima dva retka i dva stupca
    22    28
    49    64

» D = [4 5 6; 1 2 3]; <ENT>          % dva retka i tri stupca
» S = A + D <ENT>                    % za zbrajanje i oduzimanje matrice moraju
                                     % biti jednakih dimenzija
S =
     5     7     9
     5     7     9

» x = [1 2 3]; <ENT>
» y = [4 5 6]; <ENT>
» z = x .* y <ENT>                   % stavljanje točke prije operacije uzorkuje
                                     % računanje među odgovarajućim članovima
z =

     4    10    18                   % pojedinačno množenje: 1*4 2*5 3*6
```

Za proučavanje sustava važne su nam svojstvene vrijednosti matrice. Svojstvene vrijednosti određujemo funkcijom `eig`:

```
» A = [1 10 2; 35 3 4; 1 4 80]; <ENT>      % matrica A je kvadratna matrica
» eig(A) <ENT>                             % tražimo svojstvene vrijednosti

ans =

  -16.7359
   20.4343
   80.3016

» inv(A) <ENT>                             % tražimo inverz matrice A
                                           % inverz će postojati samo ako je
ans =                                      % matrica A regularna

   -0.0082    0.0288   -0.0012
    0.1018   -0.0028   -0.0024
   -0.0050   -0.0002    0.0126
```

Zadaci

1. Definirajte dvije regularne matrice

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & 1 \\ 0 & 1 & 5 \\ 1 & 10 & 2 \end{bmatrix} \quad \text{i} \quad \mathbf{B} = \begin{bmatrix} 10 & 15 & 1 \\ 25 & 6 & 4 \\ 5 & 3 & 9 \end{bmatrix}$$

   te odredite rezultat matričnih operacija +, -, *, /, \.

2. Za matrice iz prethodnog zadatka odredite također rezultate operacija član-po-član: .*, ./ i .\. U čemu je razlika?
3. Odredite inverze i svojstvene vrijednosti zadanih matrica. Kako možemo koristiti operacije / i \ za određivanje inverza?

2.3.2. Matrice - Za one koji žele znati više

Svojstvene vrijednosti matrice **A** su korijeni karakterističnog polinoma. Karakteristični polinom u MATLAB-u određujemo naredbom `poly`:

```
» A = [1 10 2; 35 3 4; 1 4 80]; <ENT>      % definiramo matricu A
» poly(A) <ENT>                            % tražimo karakteristični polinom

ans =

  1.0e+004 *

    0.0001   -0.0084   -0.0045    2.7462

» roots(ans) <ENT>                         % korijeni karakterističnog polinoma
                                           % su svojstvene vrijednosti
ans =                                      % primijetite da je ans varijabla koja
                                           % sadrži rezultat prethodne naredbe
   80.3016
   20.4343
  -16.7359
```

Prikazani postupak je u potpunosti numerički. MATLAB podržava i simboličku matematiku te bi karakteristični polinom mogli odrediti i na slijedeći način:

```
» syms lambda <ENT>                    % definiramo simboličku varijablu lambda
» det( lambda * eye(3) - A ) <ENT>     % računamo karakteristični polinom

ans =

lambda^3-84*lambda^2-45*lambda+27462

» pretty(solve(ans)) <ENT>             % svojstvene vrijednosti su korijeni polinoma

        [                      1/3    799                          ]
        [                    %1    + ----- + 28                    ]
        [                                1/3                        ]
        [                              %1                           ]
        [                                                           ]
        [        1/3           1                  1/2 /  1/3    799 \]
        [- 1/2 %1    - 799/2 ----- + 28 + 1/2 i 3    |%1    - -----|]
        [                      1/3                    |          1/3|]
        [                    %1                       \        %1   /]
        [                                                           ]
        [        1/3           1                  1/2 /  1/3    799 \]
        [- 1/2 %1    - 799/2 ----- + 28 - 1/2 i 3    |%1    - -----|]
        [                      1/3                    |          1/3|]
        [                    %1                       \        %1   /]

                                                     1/2
                           %1 := 8851 + i 431742198
```

Primijetite da MATLAB nema odgovarajuće sučelje za prikaz simboličke matematike te dobiveni rezultat nije lagano pročitati. Za ozbiljnije simboličke proračune bolje je koristiti programski sustav Mathematica, dok je MATLAB pogodniji za numeričke proračune i simulacije zbog velikog broja gotovih funkcija i modula koji su upravo namijenjeni i prilagođeni inženjerskim primjenama.

2.3.3. Reprezentacija i crtanje signala

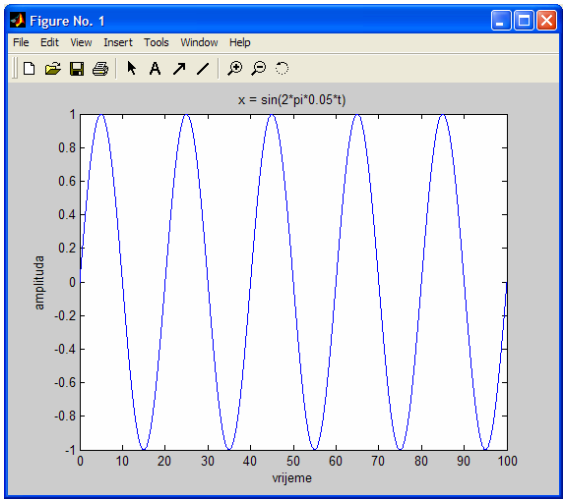
U računalu ne možemo jednostavno simbolički predstaviti neki složeni signal. U praski se najčešće signali otipkavaju u određenim vremenskim trenutcima te se tada računa s takvim otipkanim signalima. Da bi otipkali signal potrebno je definirati vektor vremena s određenim korakom. U MATLAB-u se za to najčešće koristi operator dvotočka:

```
» t = [0:0.01:100]; <ENT>              % definiramo vektor t koji započinje u nuli i
                                       % završava u 100 s korakom 0.01
» plot(t) <ENT>                        % crtamo vektor t prema indeksu
```

Ovime smo definirali vektor od 10001 članova (provjerite naredbom `length`) koji ćemo koristiti za otipkavanje neke funkcije. Otipkajmo i nacrtajmo sinus frekvencije 0.05 Hz:

```
» x = sin(2*pi*0.05*t); <ENT>          % računamo funkciju sin za svaki element
                                       % vektora vremena t
» plot(t,x) <ENT>                      % crtamo x u zavisnosti od t
» title('x = sin(2*pi*0.05*t)') <ENT>  % dajemo naslov slici
» xlabel('vrijeme') <ENT>
» ylabel('amplituda') <ENT>
```

Dobivena slika prikazuje vrijednost vektora `x`, odnosno spaja točke određene parovima koordinata iz vektora `x` i `t`. Za ostale mogućnosti naredbe `plot` pogledajte ugrađenu pomoć.



Zadaci

4. Korištenjem operatora dvotočka definiraj vektor `t` koji započinje u nuli i završava u trenutku $t = 20$ s. Neka je korak $0{,}05$ s.
5. Koristeći vektor `t` iz prethodnog zadatka otipkaj funkciju kosinus frekvencije $f = 0{,}5$ Hz. Nacrtaj dobivenu funkciju.

2.3.4. Crtanje i grafika - Za one koji žele znati više

Grafika u MATLAB-u je objekta, te se sva svojstva prikaza i sam prikaz mogu mijenjati s naredbama `get` i `set`. Naredba `get` dohvaća sva svojstva nekog objekta, dok ih naredba `set` mijenja:

```
» h = figure; <ENT>                    % otvaramo prozor za prikaz, h je broj slike
                                       % naredba plot automatski stvara novu sliku
                                       % ako ista ne postoji
» get(h) <ENT>                         % MATLAB ispisuje sva svojstva slike
        BackingStore = on
        CloseRequestFcn = closereq
        Color = [0.8 0.8 0.8]
        Colormap = [ (64 by 3) double array]
        CurrentAxes = []
        CurrentCharacter =
        ...

» set(h, 'Name', 'Slicica') <ENT>      % mijenjamo ime prozora u 'Slicica'
```

Iako korisne, naredbe `get` i `set` se obično upotrebljavaju unutar programa i skripti, dok se za uobičajen rad koristi manji skup specijaliziranih naredbi kao što su `plot`, `stem`, `xlabel` itd.

Ponekad želimo na sliku dodati posebne oznake koje npr. sadrže grčka slova i neke matematičke simbole. Za tu namjenu koristimo standardne TEX i LATEX kratice:

```
» title('x=sin(2\pift)') <ENT>      % \pi je kratica za grčko slovo π
                                    % kratice se sastoje od znaka \ i engleskog
                                    % naziva simbola, npr. \alpha, \int i \sum
```

Za dodavanje dodatnih oznaka u sliku te crtanje po slici najjednostavnije je koristiti alate iz alatne trake.

2.3.5. Simbolička matematika – Za one koji žele znati više

Jedan od mnogih programskih paketa/modula unutar MATLAB-a podržava simboličku matematiku. Njegove mogućnosti možete pogledati naredbom `help symbolic`, dok naredba `symintro` daje kratki uvod s primjerima. Svaka varijabla koju koristimo je simbolički objekt s dodatnim svojstvima koja se određuju prilikom deklaracije. MATLAB naredbe na simboličkim varijablama izvršavaju se simbolički:

```
» x = sym('x'); <ENT>               % definiramo simboličku varijablu x
» x = sym('x','real'); <ENT>        % realna simbolička varijabla x
» syms a x <ENT>                    % simboličke varijable a i x
» f = sin(a * x) <ENT>              % definiramo simboličku funkciju f(x)

f =

sin(a*x)

» diff(f) <ENT>                     % derivacija funkcije po x

ans =

cos(a*x)*a

» fourier(f) <ENT>                  % Fourierova transformacija funkcije f(x)

ans =

-i*pi*Dirac(-w+a)+i*pi*Dirac(w+a)
```

2.3.6. Funkcije i skripte – Za one koji žele znati više

MATLAB je potpun programski jezik u kojem je moguće napisati vlastite programske odsječke. Pojedine naredbe moguće je izvršiti uvjetno ili ponoviti više puta. Svaka funkcija ili skripta se sprema u običnu tekstualnu datoteku koja mora imati nastavak `.m`.

Prilikom rada u MATLAB-u uvijek kada vaš rad zahtijeva slijedno izvršavanje više naredbi u nizu dobra praksa je pisanje skripte pomoću ugrađenog editora koji se poziva naredbom `edit`.

```
» edit <ENT>                        % poziva ugrađeni MATLAB editor
```

Pretpostavimo da se želimo upoznati s raznim mogućnostima naredbe `plot` i da želimo crtati razne signale. U tom slučaju bi u MATLAB editor upisali skriptu slijedećeg sadržaja:

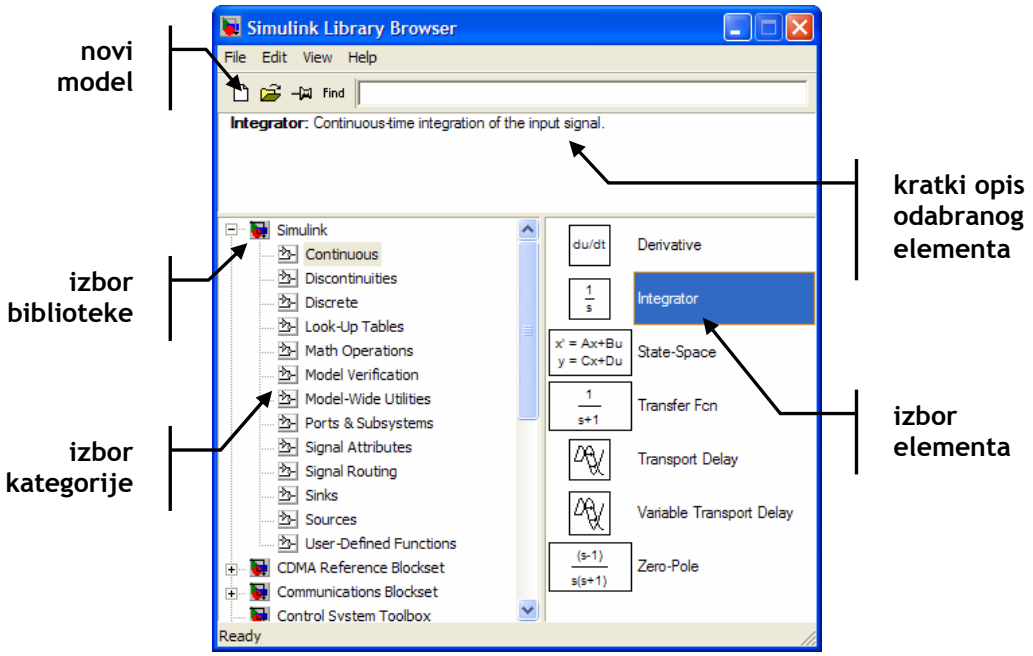
```
% Primjer MATLAB skripte – SIS laboratorijske vježbe
t = [0:0.01:100];
f = 10;
x = 2*sin(2*pi*f*t);
h = figure;
plot(t, x, 'b:');
xlabel('vrijeme u sekundama');
```

Sada svaku promjenu unosimo u editor te skriptu pozivamo iz MATLAB ljuske zadavanjem imena `.m` datoteke u kojoj se ona nalazi ili izravno iz editora odabirom `Degub→Run` stavke (`F5`). Osim toga na raspolaganju imamo i sve uobičajene opcije za privremeno zaustavljanje izvršavanja programa i analizu varijabli što nam olakšava programiranje.

2.4. Upoznavanje sa Simulinkom

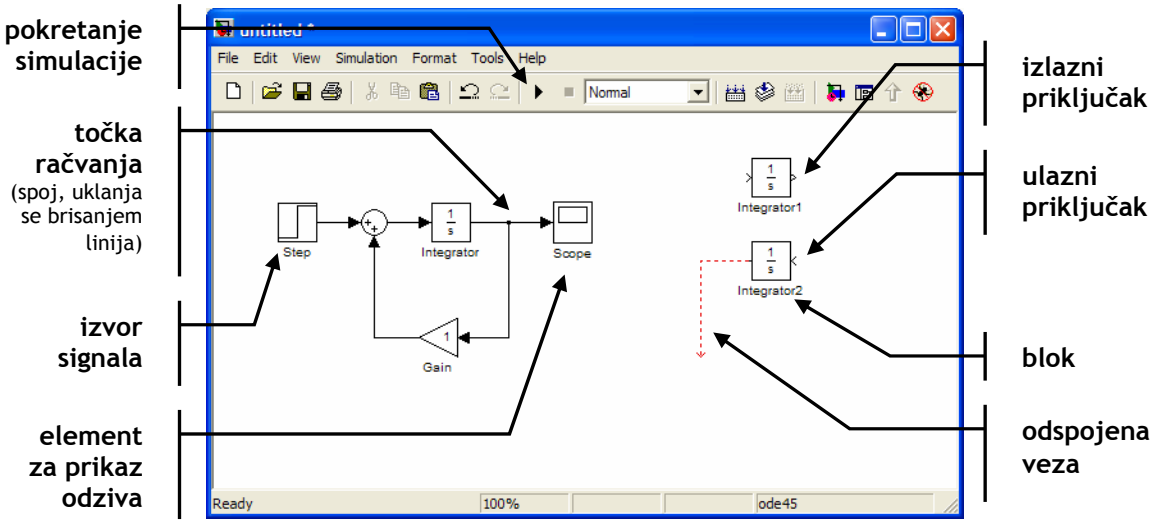
Simulink je dio MATLAB-a namijenjen simuliranju dinamičkih sustava. Za sam unos i opis sustava koji se simulira koristi se jednostavno grafičko sučelje u kojem sastavljamo/crtamo model kombinirajući gotove komponente. Takvim pristupom je simulacija sustava značajno olakšana jer se od korisnika ne zahtijeva unos diferencijalnih ili diferencijskih jednadžbi koje opisuju sustav već je dovoljno poznavanje blok-sheme sustava.

Simulink se pokreće unutar MATLAB-a zadavanjem naredbe `simulink` ili odabirom ikone iz alatne trake. Nakon pokretanja Simulinka otvara se prozor `Simulink Library Browser` prikazan na slici.



Novootvoreni prozor sadrži kolekciju svih blokova koje koristimo pri sastavljanju modela. U donjem dijelu prozora s desne strane nalaze se sve raspoložive kategorije blokova. Odabirom neke kategorije na lijevoj strani Simulink prikazuje sve blokove dostupne unutar odabrane kategorije. No da bi mogli slagati i povezivati blokove te tako definirati sustav kojeg želimo simulirati najprije moramo otvoriti novi model odabirom `File→New Model` ili klikom na ikonu alatne trake. Sada se otvara novi prozor u kojem sastavljamo model.

Na model element dodajemo tako da ga odaberemo, odvučemo i ispustimo unutar prozora novog modela. Svaki element kojeg smo dodali ima ulazne priključke označene s > i izlazne priključke označene s >. Blokove spajamo crtanjem veza između blokova (pokazivač se pretvara u alat za crtanje kada ga postavimo iznad ulaznih ili izlaznih priključaka). Dvostrukim klikom na pojedini blok dobivamo izbornik u kojem postavljamo svojstva danog bloka. Možemo nacrtati samo dozvoljena spajanja, npr. nije moguće spojiti dva izlaza skupa.



2.4.1. Simulacija jednostavnih sustava

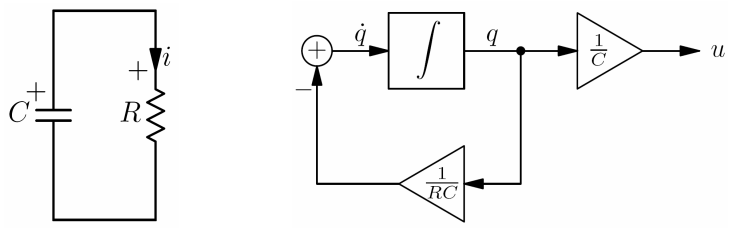
Sastavimo i simulirajmo pomoću Simulinka jednostavni sustav prvog reda koji se sastoji od idealnog kapaciteta C i otpornika R kako je prikazano na slici. Sustav možemo opisati diferencijalnom jednadžbom

$$\frac{q}{RC} + \frac{dq}{dt} = 0$$

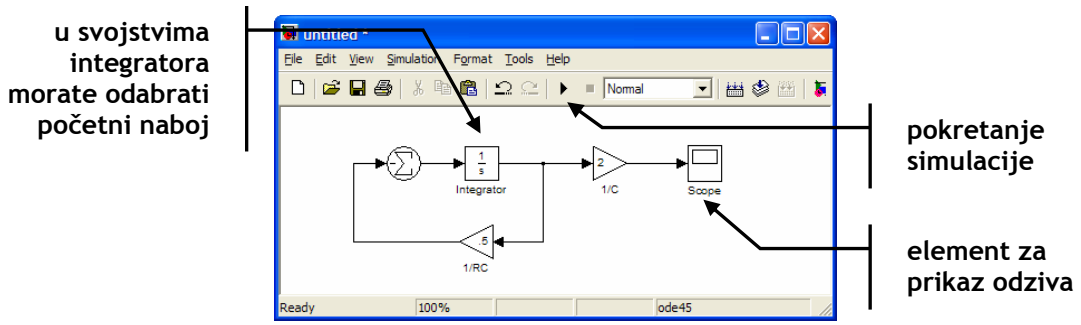
Odaberemo li za izlaz sustava napon na otporniku R kao konačno rješenje dobivamo

$$u(t) = \frac{q_0}{C} e^{-\frac{1}{RC}(t-t_0)} .$$

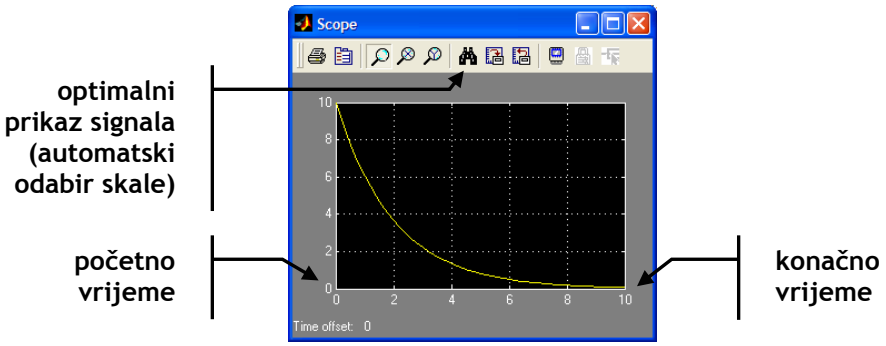
Mi naravno želimo dani sustav simulirati uz pomoć Simulinka. Pretpostavimo da raspolažemo s elementom za integriranje. Promotrimo li diferencijalnu jednadžbu koja opisuje sustav vidimo da je derivacija naboja proporcionalna naboju. Ako bi izlaz iz integratora bio naboj na kapacitetu C, ulaz u integrator tada mora biti derivacija naboja koja je proporcionalna s tim istim nabojem. Vidimo da je potrebno vratiti izlaz iz integratora na njegov ulaz uz odgovarajuće pojačanje. Time smo dobili blok-shemu sustava koju možemo odmah nacrtati u Simulinku.



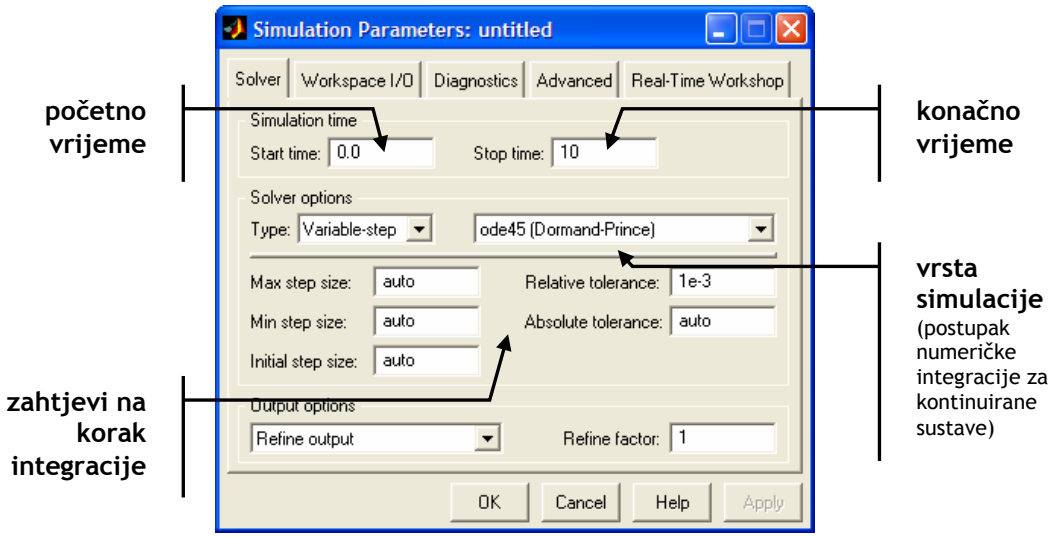
Odaberimo $R = 4$ i $C = 0{,}5$ te početni naboj $q_0 = 5$. Sada možemo nacrtati shemu u Simulinku. Pri tome pripazite na to da je blok za integriranje označen s $1/s$ a ne s integralom zbog veze s Laplaceovom transformacijom. Početno stanje odabirete u svojstvima bloka za integriranje (dvostruki klik na blok).



Da bi mogli vidjeti rezultate simulacije nakon pokretanja morate aktivirati `Scope` blok dvostrukim klikom. Time otvarate prozor prikazan na slici koji se koristi za brzi pregled rezultata. Ako vam treba složenija slika rezultati simulacije se šalju u radni prostor naredbama za crtanje (blok `To Workspace`).



U prozoru vidimo da je odziv sustava eksponencijalna funkcija koja trne kako vrijeme teži k beskonačnosti. Početna vrijednost je određena nabojem na kapacitetu. Simulacijom smo dobili vrijednosti odziva od trenutka $t = 0$ do trenutka $t = 10$. Te trenutke podešavamo u dijalogu parametara simulacije koji je prikazan na slici (odaberite stavku `Simulation→Simulation Parameters` iz izbornika). Važni parametri simulacije koje je gotovo uvijek potrebno provjeriti prije početka simulacije, uz početno i konačno vrijeme, su vrijednosti vremenskog koraka i tip numeričke integracije koji se koristi.



MATLAB i Simulink raspolažu s nekoliko različitih metoda numeričke integracije (*ODE Solvers*) koji rješavaju probleme oblika

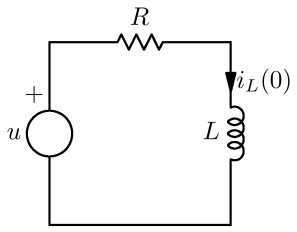
$$y' = F(t, y).$$

Pretpostavljena metoda numeričke integracije je `ode45` i prikladna je za većinu problema. Od ostalih raspoloživih metoda simulacije sustava unutar Simulinka bitno je spomenuti `discrete` postupak koji je primjenjiv za simulaciju diskretnih sustava ili općenito bilo kojih sustava koji nemaju kontinuirane varijable stanja. Pravilnim izborom postupka i parametara simulacije možete značajno skratiti vrijeme potrebno za simulaciju a istodobno povećati točnost.

Zadaci

6. Korištenjem Simulinka odredite odziv RL mreže zadane slikom na pobudu $u = s(t)$[1]. Odziv mreže (veličina koju promatrate) je struja kroz idealni induktivitet L. Neka je $R = 4$ i $L = 2$ te neka je početna struja kroz induktivitet $i_L(0) = 3$.

[1] Funkcija $s(t)$ je Heavisideova funkcija. Koristite `step` funkcijski blok, no ne zaboravite provjeriti da li je skok od nule na jedinicu postavljen točno u trenutku $t = 0$.

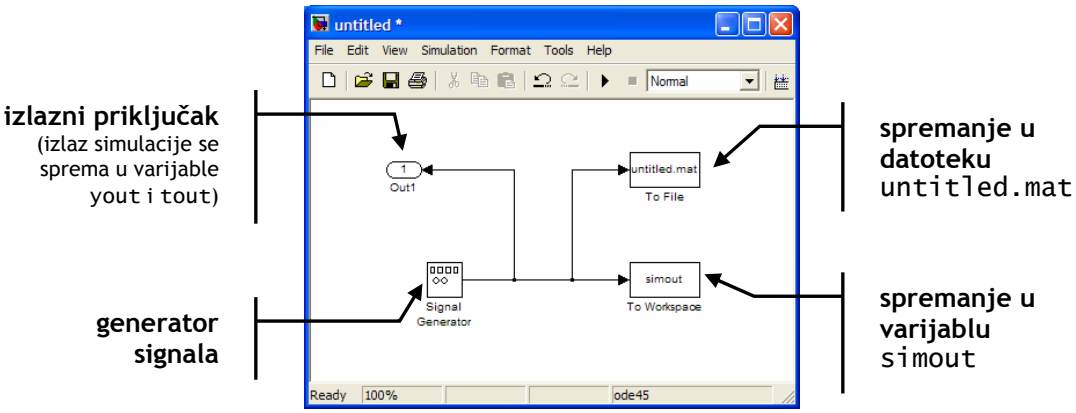


7. Što se događa s odzivom ako za početnu struju odaberemo $i_L(0) = 0{,}25$.

2.4.2. Povezivanje s MATLAB-om – Za one koji žele znati više

Simulink je sastavni dio MATLAB-a te se pri izgradnji modela mogu koristiti sve MATLAB funkcije. Također se modelu mogu proslijediti neki ulazi, a i rezultati simulacije se mogu dohvatiti iz ljuske.

Pogledajmo najprije kako rezultate simulacije spremiti bilo u datoteku bilo u neku varijablu koja je dostupna iz ljuske. Najjednostavniji primjer prikazan je na slici. U prikazanom slučaju izlaz iz generatora signala spremamo na različita mjesta.



Nakon pokretanja simulacije u MATLAB-u možemo pristupiti varijablama `tout` i `yout` (`tout` sadrži vremenske trenutke, a `yout` vrijednosti signala) strukturi `simout` (struktura sadrži i dodatne informacije o signalu) te datoteci `untitled.mat`:

```
» whos <ENT>                          % u ljusci možemo pristupiti rezultatima

  Name          Size                    Bytes  Class

  simout         1x1                     1202  struct array
  tout          51x1                      408  double array
  yout          51x1                      408  double array

Grand total is 181 elements using 2018 bytes

» whos -file untitled <ENT>           % isti rezultati su spremljeni u datoteku

  Name       Size                     Bytes  Class
```

```
  ans        2x51                      816  double array

Grand total is 102 elements using 816 bytes
```

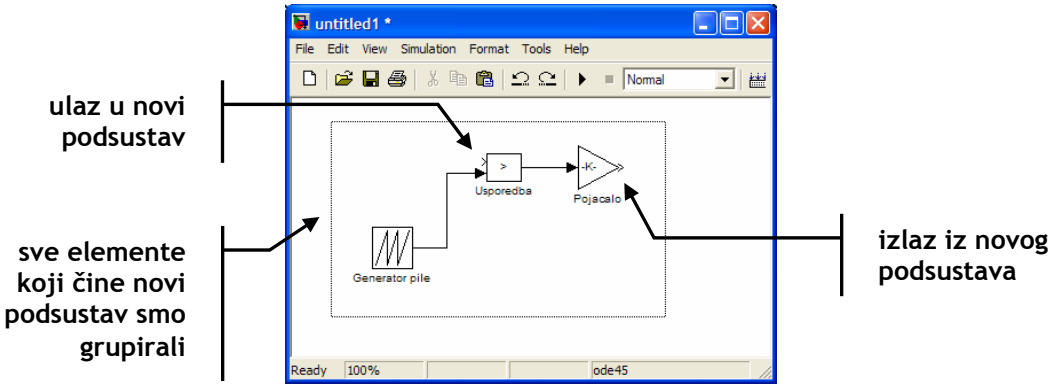
Sada na tako dobivene rezultate simulacije možemo primijeniti bilo koju od raspoloživih funkcija, npr. želimo li dobiti složenije prikaze za koje ne postoji gotov blok u Simulinku snimljeni izlaz prosljeđujmo naredbama za vizualizaciju kao što su `plot`, `stem` ili `surf`.

Ponekad je potrebno izvršiti veliki broj simulacija s različitim parametrima. U tom slučaju uobičajen način koji se sastoji od pokretanja simulacije, spremanja rezultata te naposljetku mijenjanja parametara nije pogodan. MATLAB podržava naredbu `sim` koja se može koristiti za pokretanje simulacije. Osim naredbe `sim` mogu se koristiti naredbe `set_param` i `get_param` za mijenjanje parametara modela. Kombiniranjem navedenih naredbi možemo napisati skriptu koja postavlja parametre simulacije i sprema rezultate. Osim navedenih postoje i mnoge druge naredbe za automatizirani rad sa Simulinkom.

2.4.3. Izrada funkcijskih blokova – Za one koji žele znati više

Pravi funkcijski blok za Simulink definiramo pisanjem S-funkcije. Iako za pisanje S-funkcija možemo koristiti C, C++ i neke druge programske jezike, najjednostavnije ih je napisati koristeći MATLAB-ov M programski jezik. Kako je struktura takvih funkcija nešto kompliciranija od uobičajenih MATLAB funkcija i skripti preporučamo vam da pokretanjem `sfundemos` demonstracije pogledate par ilustrativnih primjera.

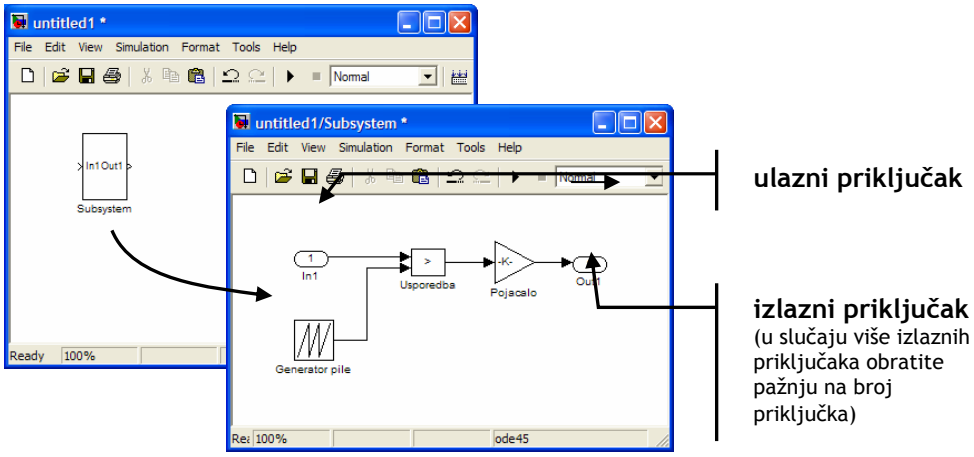
Umjesto pisanja S-funkcija ponekad je dovoljno iskoristiti gotove funkcijske blokove za slaganje novog bloka. Postupak se sastoji od sastavljanja gotovih blokova te grupiranja istih.

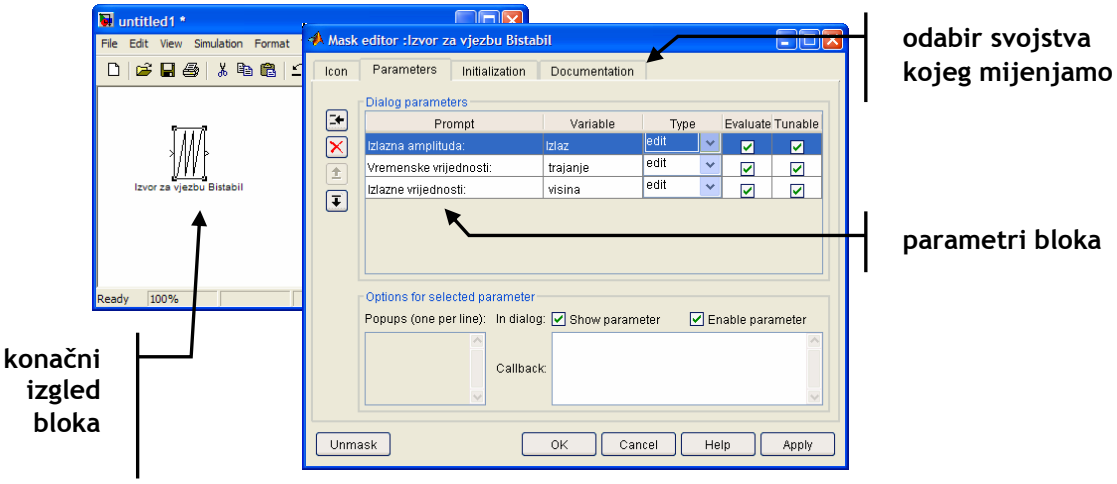


Najprije crtamo željeni sustav koji će nam predstavljati jedan blok. Pri tome određene ulaze i izlaze elemenata ne spajamo jer će oni postati novi ulazi i izlazi iz zamišljenog funkcijskog bloka. Sada grupiramo elemente koji čine naš podsustav te odabiremo `Edit→Create Subsystem`. Dobivamo novi blok koji ima onoliko ulaza i izlaza koliko smo ih ostavili odspojenima. Označimo li blok te

odaberemo `Edit→Look Under Mask` možemo pogledati kako je složen podsustav.

Sva svojstva takvog sustava, izgled maske i kratki opis za korisnika možemo mijenjati odaberemo li `Edit→Edit Mask` opciju. U novom dijalogu sada postavljamo sve parametre vezane za podsustav.





2.4.4. Runge-Kutta metoda – Za one koji žele znati više

MATLAB i Simulink imaju ugrađene različite metode za numeričko rješavanje diferencijalnih jednadžbi koji rješavaju probleme oblika

$$y' = F(t, y).$$

Ugrađene metode možemo podijeliti na metode koje koriste stalni korak te na metode koje koriste promjenjivi korak. Jedna od najčešće korištenih metoda sa stalnim korakom je Runge-Kutta metoda 4. reda (ode4 u MATLAB-u).

Pretpostavimo da je zadana jednadžba

$$y' = F(t, y)\,,\;\; y(t_0) = y_0\,.$$

Rješenje u diskretnoj domeni je dano s

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)\,,$$

gdje je k_1 nagib na početku intervala, k_2 nagib u sredini intervala (koristi se k_1 te se nagib računa u trenutku $t_n + h/2$ koristeći Eulerovu formulu[2]), k_3 je opet nagib u sredini intervala, no sada se y računa preko k_2. k_4 je nagib na kraju intervala i za njega smo vrijednost y odredili pomoću nagiba k_3. Vrijedi

$$k_1 = F\left(t_n, y_n\right),$$

$$k_2 = F\left(t_n + h/2, y_n + k_1\, h/2\right),$$

$$k_3 = F\left(t_n + h/2, y_n + k_2\, h/2\right) \text{ i}$$

$$k_4 = F\left(t_n + h, y_n + k_3 h\right).$$

[2] Najjednostavnija metoda prelaska s kontinuiranih na diskretne sustave je Eulerova metoda koja će biti razmatrana na auditornim vježbama. Eulerova metoda odgovara ode1 metodi u MATLAB-u.

3. Vježba 3. – Konačni automat

3.1. Priprema

Ako niste upoznati s programskim sustavom MATLAB pročitajte "Kratke upute za korištenje MATLAB-a" koje se mogu pronaći na stranicama predmeta (http://sis.zesoi.fer.hr/vjezbe.html#dodatci). Također detaljnije proučite dio o programiranju u MATLAB.

Konačni automati su sekvencijalni u smislu da reagiraju na niz ulaznih simbola i pri tome mijenjaju stanje. Zbog sekvencijalnosti implementacija konačnih automata na računalu je izrazito jednostavna. Prisjetimo se da je konačan automat u potpunosti određen uređenom petorkom $(Stanja, Ulazi, Izlazi, FunkcijaPrijelaza, PočetnoStanje)$. Za realizaciju automata unutar MATLAB-a biti će potrebno definirati skupove $Stanja$, $Ulazi$ i $Izlazi$ te napisati funkciju prijelaza.

3.1.1. Nizovi znakova

Prisjetimo se da se u MATLAB-u svi numerički podaci tretiraju kao matrice. Osim numeričkih podataka MATLAB može rukovati i s tekstualnim podacima koji se tretiraju kao nizovi slova:

```
» poruka = 'Ovo je jedna recenica' <ENT>    % definiramo tekst

poruka =

Ovo je jedna recenica

» whos <ENT>                                % MATLAB tekst opet tretira kao
                                            % matricu, tj. vektor redak
  Name          Size          Bytes  Class

  poruka        1x21             42  char array

Grand total is 21 elements using 42 bytes

» poruka = [poruka '!'] <ENT>               % rečenice možemo jednostavno
                                            % spajati
poruka =

Ovo je jedna recenica!
```

Za rukovanje tekstom postoje razne funkcije čiji kraći opis možete dobiti zadavanjem naredbe `help strings`. Primijetite da MATLAB rukuje sa tekstom kao i s matricama. Probajmo stoga zadati drugi red teksta:

```
» poruka = [poruka; 'Ovo je drugi red!'] <ENT>       % dodajemo red
???  All rows in the bracketed expression must have the same
number of columns.

» poruka = [poruka; 'Ovo je drugi red!     '] <ENT>  % retci matrice moraju
                                                     % biti jednake duljine
poruka =                                             % uočite razmake
```

```
Ovo je jedna recenica!
Ovo je drugi red!
```

3.1.2. Skupovi

Pri definiranju konačnog automata svaki simbol ćemo uglavnom opisati jednim tekstualnim nizom. Sada takve simbole moramo skupiti u skupove. Na taj način možemo definirati skupove ulaznih i izlaznih simbole te stanja. Skup se u MATLAB-u naziva `cell array` i može sadržavati bilo kakve varijable:

```
» Ulazi = {'50 lipa', '1 kuna', '2 kune', 'otkucaj', 'odsutan'} <ENT>

Ulazi =

    '50 lipa'    '1 kuna'    '2 kune'    'otkucaj'    'odsutan'

» whos <ENT>                % varijabla Ulazi je tipa cell array i sadrži pet simbola

  Name        Size          Bytes  Class

  Ulazi       1x5             526  cell array

Grand total is 38 elements using 526 bytes

» Ulazi(2) <ENT>            % elemente skupa možemo dohvatiti na uobičajen način

ans =

    '1 kuna'
```

Kada elemente skupa dohvaćamo na uobičajen način korištenjem oblih zagrada rezultat je opet tipa `cell array` što u nekim slučajevima nije poželjno. Pogledajmo stoga kako dohvatiti npr. drugo slovo iz drugog elementa skupa `Ulazi`:

```
» X = Ulazi(2); <ENT>               % u X spremamo drugi element
» X(2) <ENT>                        % pokušavamo ispisati treće slovo
???  Index exceeds matrix dimensions.

» whos X <ENT>                      % X je nažalost tipa cell array i ima samo
                                    % jedan element
  Name      Size          Bytes  Class

  X         1x1             104  cell array

Grand total is 7 elements using 104 bytes

» X = Ulazi{2}; <ENT>               % koristimo li vitičaste zagrade možemo
» X(2) <ENT>                        % dohvatiti drugi znak niza

ans =

                                    % znak ne vidite jer je riječ o razmaku :)

» whos X <ENT>                      % X je sada tipa char array

  Name      Size          Bytes  Class
```

```
  X          1x6              12  char array

Grand total is 6 elements using 12 bytes

» Ulazi{2}(3) <ENT>                    % dohvaćamo izravno treći znak

ans =

k
```

3.1.3. Funkcije

Funkcije unutar MATLAB-a spremamo u obične tekstualne datoteke s nastavkom `.m`. Za pisanje funkcija najbolje je koristiti ugrađeni editor. Pogledajmo primjer jednostavne funkcije koja zbraja i oduzima dva broja:

```
8.    function [zbroj, razlika] = primjer(x, y)
9.    % PRIMJER - Zbrajanje i oduzimanje dva broja.
10.   %   PRIMJER(X, Y) zbraja X i Y.
11.   %
12.   %   [Z, R] = PRIMJER(X, Y) vraca zbroj X + Y u varijabli
13.   %   Z i razliku brojeva X-Y u varijabli R.
14.
15.   % Oba ulazna argumenta moraju biti zadana.
16.   error(nargchk(2,2,nargin));
17.
18.   % Izracunaj zbroj i razliku.
19.   zbroj = x + y;
20.   razlika = x - y;
```

Danu funkciju možemo sada prevesti u pseudokod[3] te izvršiti:

```
» what <ENT>                           % koje funkcije su u trentunom direktoriju

M-files in the current directory z:\SIS\ssustavic

primjer

» pcode primjer <ENT>                  % generiramo pseudokod
» primjer(2,3) <ENT>                   % pozivamo funkciju

ans =

     5                                 % primijetite da MATLAB ispisuje samo prvi
                                       % rezultat
» [a,b] = primjer(2,3) <ENT>           % samo ako eksplicitno navedemo drugu
                                       % varijablu MATLAB je vraća
a =

     5


b =
```

[3] Generiranje pseudokoda nije nužno potrebno jer će MATLAB prilikom rada prevoditi funkcije po potrebi. Kako se `.p` datoteka uvijek izvršava ako postoje i `.m` i `.p` datoteke ne zaboravite generirati novi pseudokod ako promijenite kod odgovarajuće `.m` datoteke.

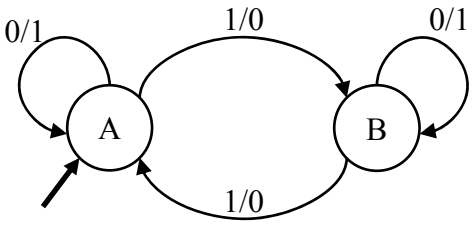
```
-1
```

Pripremni zadatak

1. Zadan je konačni automat s dva stanja $\{A, B\}$ i funkcijom prijelaza zadanom slikom. Skupovi ulaznih i izlaznih simbola su:

$$Ulazi = \{0, 1, odsutan\}$$

$$Izlazi = \{0, 1, odsutan\}$$



Napišite MATLAB funkciju `prijelaz` koja sadrži definiciju zadane funkcije prijelaza. Prisjetite se da je funkcija prijelaza definirana kao preslikavanje

$$Stanja \times Ulazi \mapsto Stanja \times Izlazi$$

te bi funkcija trebala kao ulaze primati stanje i ulaz, a kao izlaze vratiti novo stanje i izlaz[4]. Nemojte zaboraviti da je simbol *odsutan* također dozvoljen ulaz i izlaz. Prilikom usporedbe dva tekstualna niza koristite funkciju `strcmp` umjesto operatora == koji uspoređuje nizove znak po znak. Za nulu i jedinicu iz skupa simbola također pretpostavite da su tekstualni nizovi.

3.1.4. Interakcija s korisnikom

Za interakciju s korisnikom možete koristiti naredbu `input`. Uz korištenje while petlje možemo napisati jednostavnu MATLAB skriptu koja pita korisnika za unos nekog teksta sve dok ne upiše dosta:

```
% Skripta koja trazi unos od korisnika sve dok se ne unese dosta.

% Definiraj string u koji spremamo rezultat i tekst pitanja.
unos = 'start';
pitanje = 'Upiši neku riječ! Sigurno ćeš pogriješiti! >';

% Ponavljaj sve dok korisnik ne unese dosta.
while ~strcmp(unos, 'dosta')
   unos = input(pitanje, 's');
end
```

[4] Odnosno, deklaracija funkcije u MATLAB-u bi bila function `[novostanje, izlaz] = prijelaz(starostanje, ulaz)`.

Pripremni zadatak

2. Koristeći danu skriptu kao predložak napišite program koji stalno pita korisnika za unos nekog simbola. Ako se uneseni simbol nalazi u skupu ulaznih simbola zadanog automata iz prethodnog zadatka pozovite funkciju `prijelaz` te ispišite izlaz automata, a ako se simbol ne nalazi u skupu ulaznih simbola funkciji prijelaz pošaljite simbol `odsutan`. Stanje koje vam vrati funkcija prijelaza je novo stanje automata.
3. Za konačni automat opisan u zadatku 1. dijela 3.2. nacrtajte dijagram prijelaza. Nacrtani dijagram koristiti ćete na vježbi kao predložak za implementaciju automata.

3.2. Konačni automat

Zadatak

1. Koristeći funkcije koje ste napisali za pripremu kao predložak konstruirajte konačni automat (napišite funkciju prijelaza i funkciju za interakciju s korisnikom koja pamti stanje automata) koji modelira slijedeću situaciju:

   > Student i studentica su zaljubljeni i studentica je *sretna*. Kada joj student kupi *cvijeće* počne *skakati* od sreće. *Poljubi* li je student kada je sretna *udari ga*. Ako *vrijeme prolazi* a student joj ne kupi cvijeće postaje *nesretna* i počne *gnjaviti* studenta. Poljubi li je student kada je nesretna *zagrli ga* i postaje sretna. Prođe li vrijeme dok je studentica nesretna *ostaviti* će studenta[5].

   Istaknuti izrazi u opisu moraju biti bilo stanja bilo elementi skupa ulaznih i izlaznih simbola. U izvještaju obavezno navedite uređenu petorku koja definira vaš konačni diskretni automat.

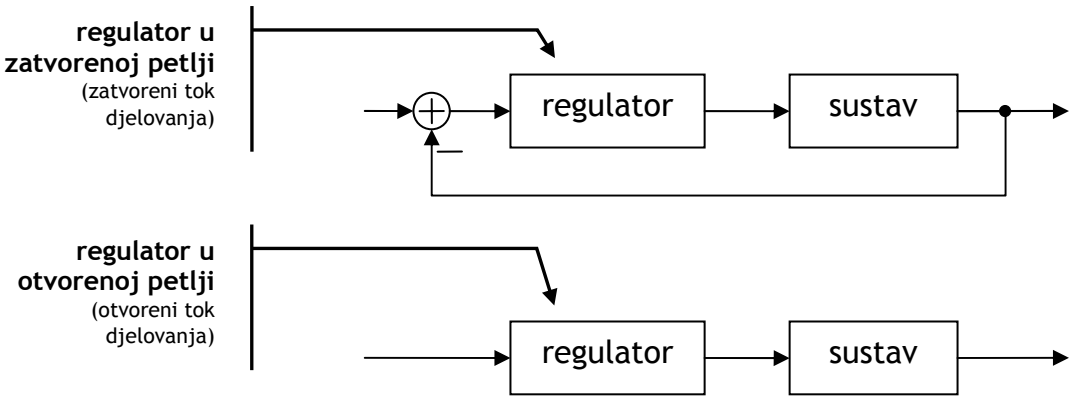
   Napomena: Korisne funkcije za rješavanje zadataka su `ismemeber`, `char`, `input`, `strcmp`, `switch`, `nargin` i `disp`. Prvo definirajte sve skupove koji su vam potrebni. Prilikom pisanja funkcije indeksirajte elemente definiranih skupova, npr. nemojte definirati novo stanje kao `stanje = 'nesretna'`, već kao `stanje = Stanja{2}`. Time smanjujete mogućnost pogrešaka koje se javljaju zbog (pre)brzog tipkanja. Funkcija prijelaza bi trebala provjeravati pripada li ulazni simbol skupu ulaznih simbola (naredba `ismemeber`). Ove napomene se odnose i na drugi zadatak.

3.3. Automatsko upravljanje sustavom

Za većinu stvarnih sustava upravljanje se obično izvodi korištenjem povratnih veza tako da na temelju izmjerenih izlaza sustava određujemo kako pobuditi sustav. Izravno upravljanje bez mjerenja izlaza sustava uglavnom nije moguće.

---

[5] Studentice mogu zamijeniti uloge u opisu.



Zadatak

2. Konstruirajte automat koji će služiti kao regulator u otvorenoj petlji. Automat mora davati ulazne simbole automatu kojeg ste konstruirali u prethodnom zadatku s ciljem da studentica nikad ne ostavi studenta. Drugim riječima, automat mora generirati simbole *poljubac* i *vrijeme prolazi* tako da studentica nikad ne dosegne stanje *ostavi studenta*.

   Primijetite da ovakav automat nema neki posebni skup ulaznih simbola već je dovoljan skup

   $$Ulazi = \{1, odsutan\},$$

   gdje jedinica uzrokuje pojavu odgovarajućeg simbola na izlazu.

   Napišite MATLAB program koji spaja dva automata u kaskadu te eksperimentalno (simulacijom) potvrdite da studentica nikada neće ostaviti studenta.

4. Vježba 4. – Linearni diskretni sustav drugog reda

4.1. Priprema

Pročitajte poglavlje 10.5. “Sustavi drugog reda” i 10.6. “Linearni sustav drugog reda” iz skripte “Signali i sustavi” koju možete pronaći na WWW stranicama predmeta (http://sis.zesoi.fer.hr/).

Pripremni zadatak

1. Analitički odredite impulsni odziv diskretnog sustava određenog jednadžbom diferencija

$$y[n] - 0{,}98\, y[n-1] + 0{,}81\, y[n-2] = u[n].$$

4.1.1. Direktna realizacija sustava

Neka je diskretni sustav opisan ulazno-izlaznom diferencijskom jednadžbom oblika

$$a_0\, y[n] + a_1\, y[n-1] + \ldots + a_k\, y[n-k] = b_0\, u[n] + b_1\, u[n-1] + \ldots + b_l\, u[n-l]$$

Ne smanjujući općenitost pretpostavimo da je prvi koeficijent $a_0 = 1$ i ograničimo se na sustav trećeg reda. Tada diferencijska jednadžba postaje

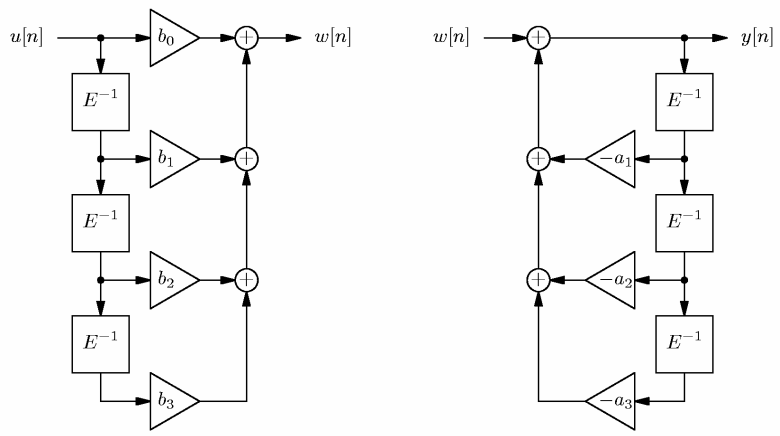
$$y[n] + a_1\, y[n-1] + a_2\, y[n-2] + a_3\, y[n-3] = b_0\, u[n] + b_1\, u[n-1] + b_2\, u[n-2] + b_3\, u[n-3].$$

Uvedemo li novi signal $w[n]$ možemo jednostavno realizirati sustave

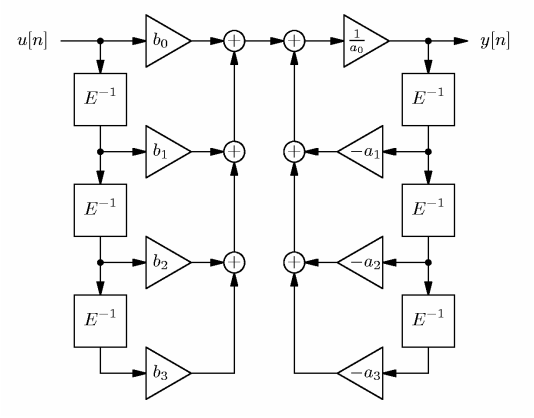
$$y[n] + a_1\, y[n-1] + a_2\, y[n-2] + a_3\, y[n-3] = w[n]$$

i

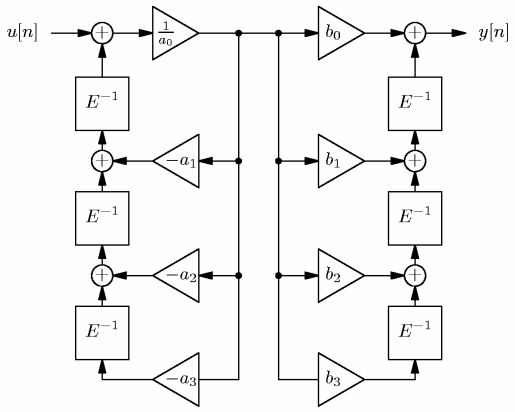
$$w[n] = b_0\, u[n] + b_1\, u[n-1] + b_2\, u[n-2] + b_3\, u[n-3].$$



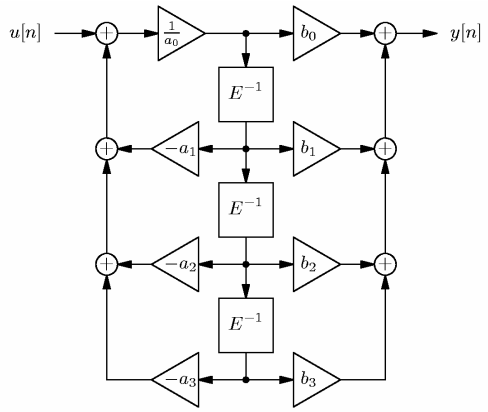
Spajanjem dobivenih realizacija u kaskadu dobivamo razne oblike direktne realizacije sustava. Sve realizacije u kojima se javljaju upravo koeficijenti ulazno-izlazne jednadžbe (odnosno prijenosne funkcije sustava) spadaju u direktne realizacije.



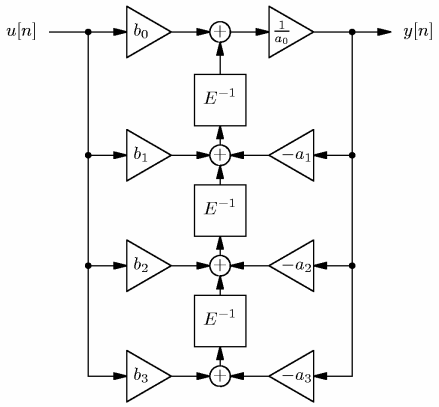
direktna forma I



direktna forma I_t



direktna forma II



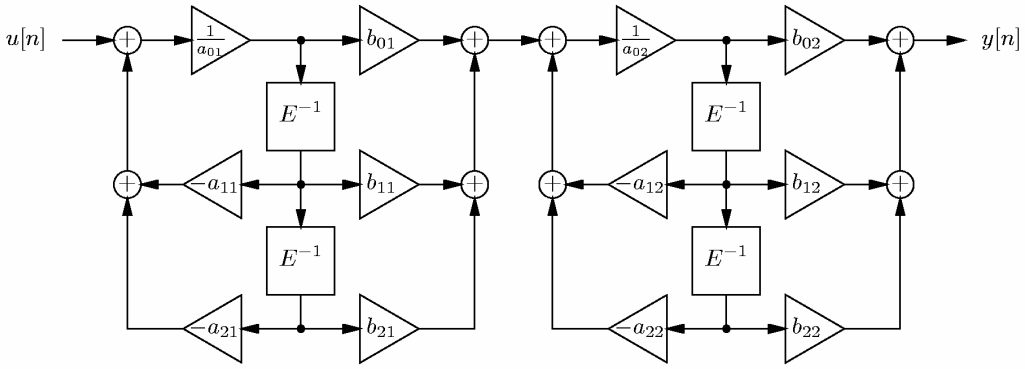
direktna forma II_t

4.1.2. Kaskadna realizacija – Za one koji žele znati više

Prikazane direktne forme su izravno povezane s prijenosnom funkcijom sustava, no nisu povoljne za stvarnu izvedbu na računalu ili DSP procesorima, pogotovo kada se uzmu u obzir efekti koji se javljaju pri kvantizaciji koeficijenata. Najčešće se stoga koristi kaskadna realizacija sustava gdje se prijenosna funkcija rastavlja u produkt racionalnih funkcija drugog reda[6].

Rastavljanjem prijenosne funkcije u kaskadu sekcija drugog reda pojednostavnili smo postupak analize sustava. Ostaje još pitanje koje parove polova i nula grupirati zajedno – za rješavanje tog optimizacijskog problema se obično koristi računalo (pogledajte naredbu `sos` u MATLAB-u).

[6] Ne koriste se racionalne funkcije prvog reda jer je potrebno izbjeći kompleksne vrijednosti pojačanja te se konjugirano-kompleksni parovi polova i nula grupiraju zajedno.

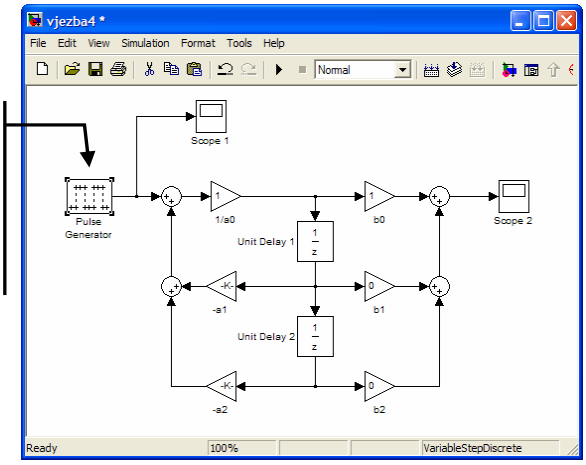


4.2. Odziv sustava

4.2.1. Impulsni odziv

Impulsni odziv diskretnog sustava je odziv na Kroneckerovu funkciju $\delta[n]$ uz početne uvjete jednake nuli. Odziv određujemo simulacijom sustava pomoću Simulinka koristeći neku od navedenih direktnih realizacija.

za dobivanje $\delta[n]$ možete koristiti *Pulse Generator* (pri tome je potrebno postaviti period dulji od vremena simulacije)



Zadaci

1. Koristeći Simulink odredite impulsni odziv diskretnog sustava određenog jednadžbom diferencija

$$y[n] - 0{,}98\, y[n-1] + 0{,}81\, y[n-2] = u[n].$$

   Usporedi impulsni odziv s odzivom iz pripremnog zadatka.

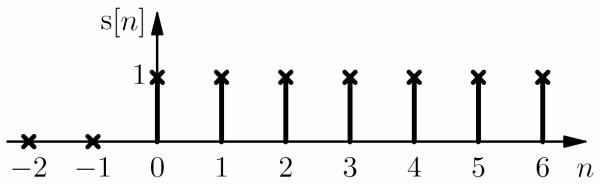
2. Koristeći Simulink odredite impulsni odziv diskretnog sustava određenog jednadžbom diferencija

$$y[n] - 1{,}38\, y[n-1] + 1{,}42\, y[n-2] = u[n].$$

3. Smanjite period generatora impulsa tako da se unutar odabranog vremena simulacije pojavljuju dva jedinična impulsa. Kakve sada odzive dobivamo za sustave iz prva dva zadatka?

4.2.2. Odziv na skok

Odziv na skok diskretnog sustava je odziv na funkciju $\mathrm{s}[n]$. Odziv određujemo simulacijom sustava pomoću Simulinka koristeći neku od navedenih direktnih realizacija.



Zadaci

4. Koristeći Simulink odredite odziv na $\mathrm{s}[n]$ diskretnog sustava određenog jednadžbom diferencija

$$y[n] - 0{,}98\, y[n-1] + 0{,}81\, y[n-2] = u[n].$$

   Neka su početni uvjeti jednaki nuli za prvu simulaciju. Za drugu simulaciju odaberite početne uvjete različite od nule.

5. Koristeći Simulink odredite odziv na $\mathrm{s}[n]$ diskretnog sustava određenog jednadžbom diferencija

$$y[n] - 1{,}38\, y[n-1] + 1{,}42\, y[n-2] = u[n].$$

   Neka su početni uvjeti jednaki nuli.

4.3. Stabilnost sustava

Rastu li neke varijable u sustavu neograničeno kako vrijeme teži k beskonačnosti kažemo da je sustav nestabilan. Kako je navedeni opis dosta nedorečen često se koristi BIBO definicija (*bounded-input bounded-output*):

Sustav je stabilan ako daje ograničeni odziv za svaku ograničenu pobudu.

Zadatak

6. Promatranjem dobivenih odziva na $\delta[n]$ i $\mathrm{s}[n]$ za prethodno zadane diferencijske jednadžbe odredite koji sustav nije BIBO stabilan.

4.3.1. Položaj polova i nula

Za linearne diskretne sustave dovoljno je promatrati vlastite frekvencije sustava, odnosno položaj polova u z ravnini. Nalaze li se svi polovi unutar jedinične kružnice sustav je stabilan, a u suprotnome je nestabilan. Jedinična kružnica nam predstavlja granicu stabilnosti.

Promotrimo sustav opisan jednadžbom

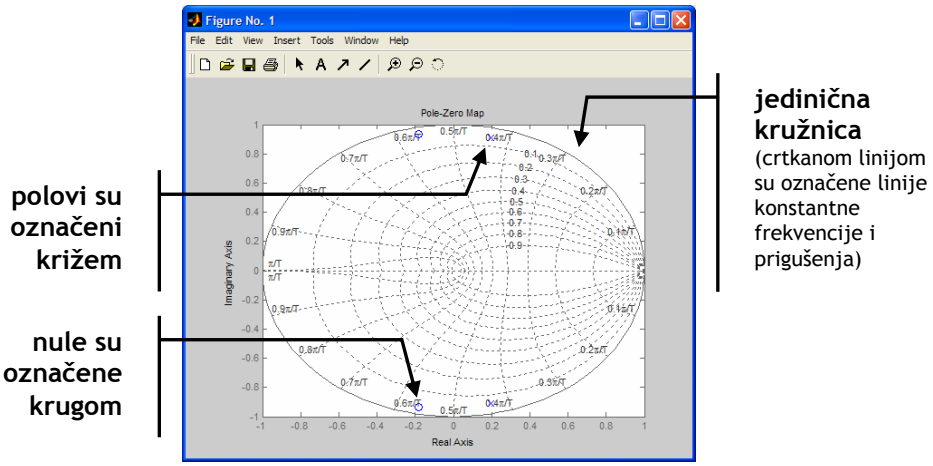
$$y[n] + a_1\, y[n-1] + a_2\, y[n-2] = b_0\, u[n] + b_1\, u[n-1] + b_2\, u[n-2].$$

Prijenosna funkcija tog sustava u z-domeni je

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} .$$

Uobičajeno se prijenosna funkcija izražava po z^{-1}, a ne po z. Polovi sustava su korijeni nazivnika, dok su nule sustava korijeni brojnika.

```
» A=[90 -35 78]; <ENT>                % definiramo koeficijente nazivnika
» B=[11 4 10]; <ENT>                  % definiramo koeficijente brojnika
» roots(A) <ENT>                      % računamo korijene nazivnika

ans =

   0.1944 + 0.9104i
   0.1944 - 0.9104i

» abs(ans) <ENT>                      % kako je apsolutna vrijednost korijena manja
                                      % od jedinice sustav je stabilan
ans =

    0.9309
    0.9309

» sys=tf(B,A,1) <ENT>                 % kreiramo LTI objekt koji predstavlja sustav
                                      % primijetite da MATLAB prijenosnu funkciju
Transfer function:                    % ispisuje po potencijama od z
11 z^2 + 4 z + 10
-----------------
90 z^2 - 35 z + 78

Sampling time: 1
» pzmap(sys) <ENT>                    % crtamo polove i nule sustava
» zgrid <ENT>                         % crtamo koordinatnu mrežu
```



Zadatak

7. Odredite položaj polova i nula sustava

$$y[n] - 0{,}98\, y[n-1] + 0{,}81\, y[n-2] = u[n]$$

i

$$y[n] - 1{,}38\, y[n-1] + 1{,}42\, y[n-2] = u[n].$$

Na temelju položaja polova odredi koji sustav je stabilan, a koji nestabilan.

4.3.2. Sustav na granici stabilnosti – Za one koji žele znati više

Za diskretne sustave jedinična kružnica predstavlja granicu između područja stabilnosti i nestabilnosti s obzirom na položaj polova. Za sustav kojemu se neki jednostruki polovi nalaze točno na jediničnoj kružnici dok su svi ostali polovi unutar jedinične kružnice kažemo da je na granici stabilnosti. Svaki jednostruki pol koji se nalazi na jediničnoj kružnici daje titranje stalne amplitude[7].

Nalazi li se na jediničnoj kružnici višestruki pol sustav postaje nestabilan jer varijabla stanja povezana s tim polom neograničeno raste (za slučaj k-strukog pola p dominantni član je $n^k p^n$).

Zadaci – Za one koji žele znati više

8. Snimite odziv na $\delta[n]$ i $s[n]$ diskretnog sustava drugog reda koji ima jedan jednostruki pol na jediničnoj kružnici dok se drugi pol nalazi unutar jedinične kružnice (npr. $2\, y[n] - 3\, y[n-1] + y[n-2] = u[n]$).
9. Snimite odziv na $\delta[n]$ i $s[n]$ diskretnog sustava drugog reda koji ima dvostruki pol na jediničnoj kružnici (npr. $y[n] - 2\, y[n-1] + y[n-2] = u[n]$).

4.4. Prikaz u prostoru stanja – Za one koji žele znati više

Promatramo li sustav kroz jednadžbu stanja i izlaznu jednadžbu vidimo da svakoj varijabli stanja odgovara jedan memorijski element. Koristimo li direktnu realizaciju sustava putanju u prostoru stanja možemo snimiti odnosno nacrtati zapamtimo li signale koji su se pojavljivali na izlazima elemenata za kašnjenje.

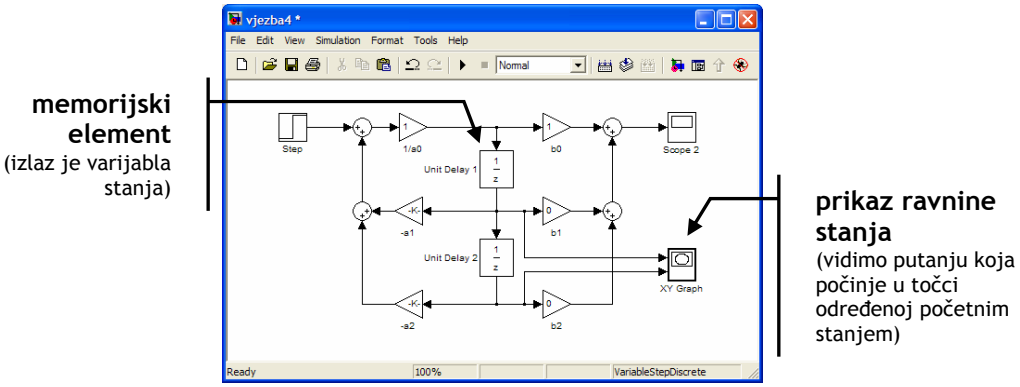
Za slučaj sustava drugog reda prostor stanja je ravnina. Putanja uvijek započinje u početnom stanju sustava. Promatramo li stabilan sustav putanja se uvijek približava točki koja određuje stacionarno stanje.

Zadatak – Za one koji žele znati više

10. Snimite putanje u prostoru stanja za oba diskretna sustava koja smo promatrali.

---

[7] Primijetite da je konstanta kao vlastita frekvencija u odzivu posljedica pola koji se nalazi u jedinici jer je $1^n = 1$.



4.5. Frekvencijska karakteristika

4.5.1. Računanje frekvencijske karakteristike

Linearni diskretni sustav možemo karakterizirati prijenosnom funkcijom. Izvršimo li zamjenu $z \mapsto e^{j\omega}$ dobivamo frekvencijsku karakteristiku sustava. Dobivena frekvencijska karakteristika je periodična te se uobičajeno računa i crta samo jedan period. Unutar MATLAB-a koristi se naredba `freqz`:

```
» B=[11 4 10]; <ENT>          % brojnik prijenosne funkcije
» A=[90 -35 78]; <ENT>        % nazivnik prijenosne funkcije
» freqz(B,A) <ENT>            % računamo i crtamo frekvencijsku karakteristiku
```

Zadatak

11. Korištenjem naredbe `freqz` odredite amplitudnu i faznu frekvencijsku karakteristiku za oba diskretna sustava koja smo promatrali.

4.5.2. Snimanje frekvencijske karakteristike

Korištenjem frekvencijske karakteristike koja nam određuje *stacionarno stanje* možemo odmah odrediti izlaznu amplitudu za harmonijsku pobudu oblika $u[n] = Ae^{j\omega_0 n}$. Iz frekvencijske karakteristike očitavamo pojačanje za danu frekvenciju pobude te tada računamo izlazni signal prema

$$y[n] = H(e^{j\omega_0})Ae^{j\omega_0 n}.$$

No isto tako simulacijom odziva sustava na harmonijsku pobudu te očitavanjem promjene amplitude i faze možemo snimiti frekvencijsku karakteristiku točku po točku.

Primijetite da stacionarno stanje postoji samo za stabilne sustave tako da frekvencijska karakteristika nema smisla promatramo li nestabilne sustave.

Zadatak

12. Na vaš simulacijski dijagram postavite harmonijsku pobudu (sinus ili kosinus funkcija) te snimite par točaka amplitudne i fazne frekvencijske

karakteristike za oba diskretna sustava koja smo promatrali. Poklapaju li se rezultati s frekvencijskom karakteristikom određenom pomoću naredbe `freqz`? Možete li simulacijom odrediti frekvencijsku karakteristiku nestabilnog sustava?

5. Vježba 5. – Nelinearni diskretni sustav - infekcijska jednadžba

5.1. Priprema

Prisjetite se svega što ste do sada naučili o diskretnim sustavima.

5.1.1. Kermack-McKendrickov epidemijski model

W. O. Kermack i A. G. McKendrick su modelirali širenje zaraznih bolesti unutar neke populacije. U njihovom modelu cijela populacija je podijeljena u tri grupe.

a) Osobe podložne zarazi (*susceptible*) su pojedinci koji se mogu zaraziti i postati nosioci zaraze.

b) Zaraženi (*infective*) pojedinci su oni koji šire zarazu unutar populacije.

c) Uklonjeni (*recovered*, *removed*) pojedinci su oni koji su se zarazili pa umrli ili pak oni koji su se oporavili i postali imuni te se više ne mogu zaraziti. U ovu skupinu spadaju i izolirani pojedinci (zbog bilo kojeg razloga, npr. karantena) koji više ne mogu doći u kontakt s ostalom populacijom. Ukratko, uklonjeni pojedinci više ne doprinose širenju zaraze[8].

Sam proces zaraze i uklanjanja pojedinaca se odvija prema slijedećim pravilima:

a) Brzina promjene broja podložnih osoba je proporcionalna broju kontakata između podložnih i zaraženih pojedinaca. Broj kontakata procjenjujemo prema ukupnom broju podložnih zarazi i zaraženih te uzimamo da je proporcionalan produktu SI gdje je S broj podložnih pojedinaca, a I broj zaraženih pojedinaca. Ovakav model zanemaruje vrijeme inkubacije.

b) Zaraženi pojedinci se uklanjaju iz ukupne populacije proporcionalno njihovom broju I.

c) Ukupni broj pojedinaca je stalan i vrijednost $S+I+R$ se ne mijenja. Model dakle zanemaruje sve promjene ukupnog broja pojedinaca koje se javljaju zbog rođenja, smrti, migracije i ostalih uzroka.

Predstavimo li sva navedena pravila u matematičkom obliku dobivamo sustav diferencijalnih jednadžbi:

---

[8] Ovakvi modeli se nazivaju SIR modeli zbog ciklusa *susceptible*→*infected*→*recovered*. Takvi modeli nisu pogodni za modeliranje epidemije bolesti kao što su prehlada ili gripa jer se kod tih bolesti ne može razviti trajni imunitet. Modeli koji modeliraju bolesti kod kojih se ne razvija trajni imunitet se nazivaju SIS modeli zbog ciklusa *susceptible*→*infected*→*susceptible*.

$$\dot{S} = -iSI$$
$$\dot{I} = iSI - rI$$
$$\dot{R} = rI$$

gdje su i i r pozitivne konstante koje određuju stope zaraze i smrtnosti (uklanjanja pojedinaca).

Iz prve jednadžbe vidimo da $S(t)$ ne može biti rastuća funkcija, dok druga jednadžba implicira povećanje broja zaraženih $I(t)$ s vremenom t ako je $S(t) > r/i$, a smanjenje u protivnom. Shodno tome, ako je u početnom trenutku $t = 0$ broj podložnih pojedinaca S_0 manji od r/i epidemija umire (nije održiva), no ako je S_0 veći od kritične vrijednosti r/i epidemija počinje. U tom slučaju se broj zaraženih pojedinaca povećava sve dok je $S(t)$ veće od kritične vrijednosti r/i, a nakon toga počinje padati.

Iz prethodnog razmatranja je vidljivo da se epidemija javlja ako i samo ako je početni broj podložnih pojedinaca veći od praga S_P koji u ovom modelu iznosi r/i. Na primjer, opasna epidemija visoko smrtonosne bolesti se rijetko javlja jer je stopa smrtnosti visoka.

5.2. Diskretni SIR model

U pripremi smo razmotrili kontinuirani Kermack-McKendrickov epidemijski model. Prema prethodno navedenim pravilima možemo definirati diskretan model širenja epidemije.

Neka su $S[n]$, $I[n]$ i $R[n]$ ukupni brojevi pojedinaca koji pripadaju grupama podložnih, zaraženih i uklonjenih. Uz pretpostavku prostorno neograničenih interakcija između pojedinaca (svaki podložni pojedinac može biti jednako-vjerojatno zaražen od strane svakog zaraženog pojedinca) model možemo predstaviti sa sljedećim skupom diferencijskih jednadžbi

$$I[n+1] = I[n] + S[n]\left(1 - e^{-iI[n]}\right) - rI[n]$$
$$R[n+1] = R[n] + rI[n]$$
$$S[n+1] = N - I[n+1] - R[n+1]$$

gdje je i koeficijent proporcionalan vjerojatnosti zaraze podložnog pojedinca i r vjerojatnost uklanjanja zaraženog pojedinca. N je ukupni broj pojedinaca u populaciji.

Zadaci

1. Pokažite da kako korak n teži k beskonačnosti vrijednosti S_∞, I_∞ i R_∞ postoje. Koja je vrijednost od I_∞?
2. Koristeći razmatranje za kontinuirani sustav pokažite da za ovaj model epidemija postoji ako i samo ako je početni broj podložnih pojedinaca S_0 veći od kritične vrijednosti. Neka su početni uvjeti takvi da je $R_0 = 0$ i $I_0 \ll S_0$.

5.2.1. Modeliranje marketinških aktivnosti

Zadatak

3. Koristeći diskretni SIR model modelirajte sljedeću situaciju:

   Vi ste vlasnik nove tvrtke "*Varaj kao profesionalac*" koja prodaje najbolja pomagala za varanje na ispitu koja su čak ispitana u praksi, no usprkos tome još se niste probili na tržište. Zbog toga ste se odlučili na marketinšku kampanju usmjerenu na vaše potencijalne kupce kako bi ih informirali o proizvodu. Marketinški pristup kojeg koristite je izravan u smislu da planirate zaposliti određen broj učenika ili studenata koji bi među ciljanom populacijom proširili vijest o proizvodima. Pri tome, naravno, želite postići maksimalan efekt uz minimalne troškove te ste se odlučili simulirati vašu kampanju još u fazi planiranja.

   Epidemija je u ovom slučaju širenje vaše reklamne poruke. Sami procijenite stope širenja i uspješnosti reklame (širenje odgovara stopi zaraze pojedinaca a uspješnost stopi uklanjanja[9]), ukupnu populaciju te broj podložnih, zaraženih i uklonjenih pojedinaca. Početni broj zaraženih pojedinaca neka odgovara broju zaposlenih učenika/studenata koji šire informaciju. Pokažite simulacijom da je potreban kritičan broj početnih pojedinaca koji šire informaciju među populacijom da bi kampanja bila uspješna.

   Snimite promjene varijabli $S[n]$, $I[n]$ i $R[n]$ u ovisnosti o koraku n. Ispitajte ponašanje vašeg modela za različite veličine populacije te za različite vrijednosti stopa širenja i uspješnosti.

5.2.2. Modeliranje širenja podataka Internetom

Zadatak

4. Koristeći diskretni SIR model modelirajte sljedeću situaciju:

   Kontroverzni *dokumentarni film*[10] se pojavio na internetu i trenutno ga dijeli mali broj korisnika. No taj film je pobudio veliki interes te se strahovitom brzinom širi među korisnicima interneta. Zanima nas koje je vrijeme potrebno za zarazu cijele populacije te da li je zarazu moguće zaustaviti.

   Za ovaj slučaj pretpostavite manji broj početnih korisnika koji su *zaraženi* (dijele film), no također pretpostavite visoku stopu širenja i te malu stopu uklanjanja pojedinaca r.

---

[9] Pojedinca uklanjamo kada je ili kupio proizvod ili u potpunosti odustao od kupnje.

[10] Zadatak je inspiriran stvarnim slučajem kada su se privatne snimke našle dostupne putem Interneta.

Snimite promjene varijabli $S[n]$, $I[n]$ i $R[n]$ u ovisnosti o koraku n. Ispitajte ponašanje vašeg modela za različite vrijednosti uklonjenih pojedinaca (ne žele dijeliti film) te za različite vrijednosti stope uklanjanja. Da li je moguće zaustaviti širenje? Da li je moguće zaustaviti širenje jednom kada započne? Obrazložite odgovor.

5.2.3. Dodatna objašnjenja vježbe

U ovoj vježbi vam je ostavljena velika sloboda prilikom izrade vježbe. Slične probleme koji nisu u potpunosti definirani ćete vjerojatno sve češće susretati kako se bližite kraju studija – od vas se neće zahtijevati samo da ponovite napisano već da pokažete vlastitu kreativnost. Kako nekim studentima to predstavlja problem ovdje navodimo dodatne upute za vježbu.

U vježbi je potrebno napraviti simulacijski model diskretnog SIR sustava unutar Simulinka kao što ste radili na prošloj laboratorijskoj vježbi. Svi potrebni signali $S[n]$, $I[n]$ i $R[n]$ se mogu predstaviti izlazima iz blokova za jedinično kašnjenje (`Unit Delay`) na način da je ulazni signal u blok vrijednost npr. $S[n+1]$, dok je izlaz tada $S[n]$. U odgovarajuće blokove se zatim postavljaju početni uvjeti za simulaciju. Kada ste postavili tri potrebna `Unit Delay` bloka potrebno je dodati spojeve tako da struktura odgovara skupu diferencijskih jednadžbi

$$
\begin{aligned}
I[n+1] &= I[n] + S[n]\left(1 - e^{-iI[n]}\right) - rI[n] \\
R[n+1] &= R[n] + rI[n] \\
S[n+1] &= N - I[n+1] - R[n+1]
\end{aligned}
$$

Funkcija e^x se može dobiti kao `Math Operations`→`Math Function`, pri čemu se odabire eksponencijalna funkcija. Na ulaz tog bloka se dovodi izlaz iz bloka `Unit Delay` koji odgovara signalu $I[n]$, no provučen kroz pojačalo pojačanja $-i$. U izrazu vam je još potreban N koji je stalan tijekom simulacije te odgovara zbroju početnih vrijednosti $S[0] + I[0] + R[0]$, odnosno morate ga prilagoditi svaki put kada promijenite početne uvjete.

Kako bi izbjegli potrebu za ponovnim računanjem parametra N preporučamo vam da sve početne uvjete definirate kao varijable koje možete mijenjati iz radnog prostora, npr. u blokove `Unit Delay` bi redom upisali `s0`, `r0` i `i0`. Sada vrijednost $N = S[0] + I[0] + R[0]$ dobivate korištenjem `Sources`→`Constant` bloka u kojeg upišete `s0+r0+i0`. Mijenjanje parametara se u ovom slučaju radi u radnom prostoru.

U vaš model još je samo potrebno dodati blokove za crtanje signala te možete pokrenuti simulaciju.

Interpretacija značenja parametara modela i i r puno je jednostavnija ako promatrate kontinuirani model. Parametri i i r su u rasponu od nula do jedan s time da parametar i određuje koliki je dio populacije koju zarazi *jedan* nosilac u

jednom koraku ako je cijela populacija podložna zarazi. Parametar r je dio *zaražene* populacije koja *prestaje biti zaražena* u jednom koraku.

Važno je uočiti da se sustav uvijek stabilizira ako je ispravno sastavljen, s time da epidemija počinje samo ako je početna populacija podložnih pojedinaca veća od kritične vrijednosti S_P koja u ovom modelu iznosi r/i.

Vrijednosti koje možete koristiti za isprobavanje su npr. $r = 0{,}4$, $i = 10^{-4}$, $S[0] = 6000$, $I[0] = 1$, $R[0] = 0$ te $r = 0{,}5$, $i = 10^{-3}$, $S[0] = 10500$, $I[0] = 100$, $R[0] = 0$. Za navedene parametre bi trebali dobiti odzive iz kojih se vidi početak epidemije koji se zatim lagano stišava. **VAŽNO: izvještaji s laboratorijskih vježbi koji budu imali jednake parametre neće biti prihvaćeni!**

6. Vježba 6. – Linearni kontinuirani sustav drugog reda

6.1. Priprema

Pročitajte poglavlje 5. "Sustavi drugog reda" iz skripte "Signali i sustavi" koju možete pronaći na WWW stranicama predmeta (http://sis.zesoi.fer.hr/).

Pripremni zadatak

1. Analitički odredite impulsni odziv kontinuiranog sustava određenog diferencijalnom jednadžbom

$$y''(t) + 2\,y'(t) + 26\,y(t) = u(t).$$

6.1.1. Direktna realizacija sustava

Neka je linearni kontinuirani sustav opisan ulazno-izlaznom diferencijalnom jednadžbom oblika

$$a_k\,y^{(k)}(t) + a_{k-1}\,y^{(k-1)}(t) + \ldots + a_1\,y^{(1)}(t) + a_0\,y(t) = \\ = b_l\,u^{(l)}(t) + b_1\,u^{(l-1)}(t) + \ldots + b_1\,u^{(1)}(t) + b_0\,u(t)$$

Ne smanjujući općenitost pretpostavimo da je prvi koeficijent $a_k = 1$ i ograničimo se na sustav trećeg reda. Tada diferencijalna jednadžba postaje

$$y'''(t) + a_2\,y''(t) + a_1\,y'(t) + a_0\,y(t) = b_3\,u'''(t) + b_2\,u''(t) + b_1\,u'(t) + b_0\,u(t).$$

Prijenosna funkcija sustava opisanog dobivenom diferencijalnom jednadžbom je

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0}\,.$$

Realiziranjem kontinuiranog sustava preko pojačala i integratora tako da pojačanja odgovaraju koeficijentima prijenosne funkcije sustava $H(s)$ dobivamo direktnu realizaciju[11].

Dobiveni sustav možemo opisati i u prostoru stanja jednadžbom stanja

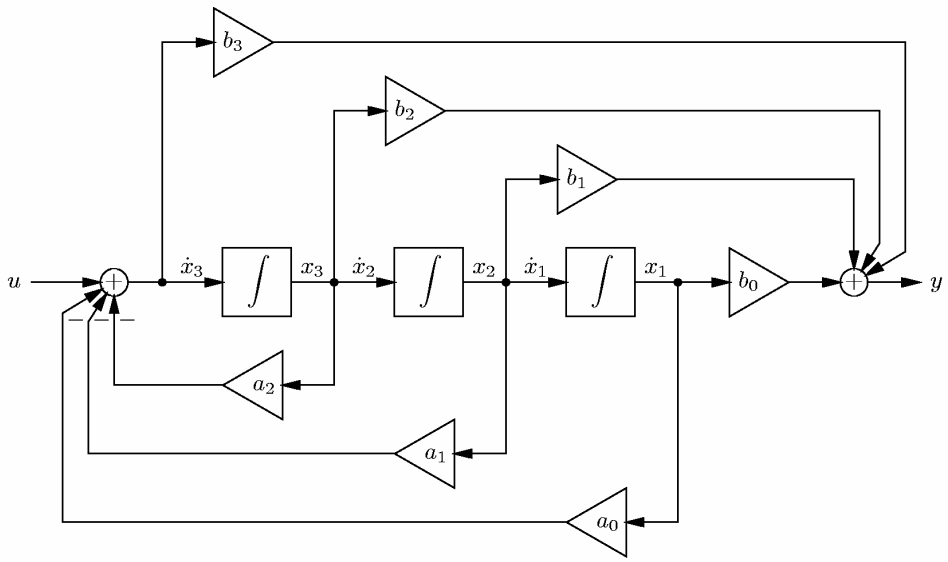
$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

i izlaznom jednadžbom

[11] Usporedite dobivenu realizaciju s direktnim realizacijama diskretnog sustava te uočite sličnosti i razlike. Pri tome pripazite na činjenicu da se prijenosne funkcije diskretnog sustava uobičajeno pišu po z^{-1} a za kontinuirane sustave po s.

$$y(t) = \begin{bmatrix} b_0 - a_0 b_3 & b_1 - a_1 b_3 & b_2 - a_2 b_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_3 \end{bmatrix} u(t),$$

pri čemu varijable stanja x_1, x_2 i x_3 odgovaraju upravo izlazima iz integratora korištenih za realizaciju sustava.



6.2. Odziv sustava

6.2.1. Impulsni odziv

Impulsni odziv sustava je odziv mrtvog sustava na Diracovu funkciju $\delta(t)$. Simulink nema izvora koji bi davao pobudu $\delta(t)$ funkcijom[12] te se impulsni odziv određuje izravno u MATLAB-u.

```
» A=[90 35 78]; <ENT>                % nazivnik prijenosne funkcije
» B=[11 4 10]; <ENT>                 % brojnik prijenosne funkcije
» impulse(B,A) <ENT>                 % crtamo impulsni odziv koristeći staru
                                     % sintaksu naredbe impulse

» sys=tf(B,A) <ENT>                  % kreiramo LTI objekt koji opisuje
                                     % kontinuirani sustav
Transfer function:
11 s^2 + 4 s + 10
-----------------
90 s^2 + 35 s + 78
```

[12] Primijetite da je ipak moguće pobuditi sustav nekim signalom koji je blizak Diracovom impulsu te bi dobiveni odziv u tom slučaju bio blizak impulsnom odzivu (signal mora imati što kraće trajanje te jediničnu površinu).

```
» [h,t]=impulse(sys); <ENT>          % računamo impulsni odziv h
» impulse(sys, 100); <ENT>           % crtamo impulsni odziv do trenutka 100
```

Zadaci

1. Odredite impulsni odziv kontinuiranog sustava određenog diferencijalnom jednadžbom
$$y''(t) + 2\,y'(t) + 26\,y(t) = u(t).$$
Usporedite dobiveni odziv s odzivom iz pripremnog zadatka.
2. Odredite impulsni odziv kontinuiranog sustava određenog diferencijalnom jednadžbom
$$y''(t) - 2\,y'(t) + 26\,y(t) = u(t).$$
Da li je sustav stabilan?

6.2.2. Odziv na skok

Odziv na skok kontinuiranog sustava je odziv mrtvog sustava na Heavisideovu funkciju $s(t)$. Odziv možemo odrediti simulacijom u Simulinku ili korištenjem naredbe `step`.

Zadatak

3. Odredite odziv na $s(t)$ kontinuiranih sustava iz prethodna dva zadatka.

6.2.3. Odziv na harmonijsku pobudu – Za one koji žele znati više

Harmonijska pobuda je bitna u analizi linearnih vremenski nepromjenjivih kontinuiranih sustava jer je odziv također harmonijska funkcija. Nadalje, kako većinu signala možemo predstaviti kao linearnu kombinaciju raznih harmonijskih funkcija određivanje odziva možemo svesti na određivanje odziva na svaku od harmonijskih komponenti koje zajedno čine pobudni signal.

Zadatak – Za one koji žele znati više

4. Za kontinuirani sustav opisan diferencijalnom jednadžbom
$$y''(t) + 0{,}2\,y'(t) + 0{,}16\,y(t) = u(t)$$
napravite model za simulaciju. Odredite odziv na pravokutni signal frekvencije $f = 0{,}1\ \mathrm{Hz}$ i amplitude $0{,}32$. Odredite također pojedinačne odzive na prve tri harmonijske komponente koje čine zadani pravokutni signal. Usporedite odziv sustava na pravokutni signal s zbrojem odziva sustava na prve tri harmonijske komponente.

6.2.4. Odziv na slučajni signal – Za one koji žele znati više

Promatramo li odziv na neki slučajni signal znamo da ga određujemo na jednaki način kao i za deterministički signal pomoću npr. konvolucijskog

integrala. Odziv određen na takav način nije pretjerano koristan jer je ispravan samo za točno određeni slučajni signal kojim smo pobudili sustav.

Uobičajeno se pri analizi odziva sustava na neki slučajni signal sam slučajni signal promatra kao jedna realizacija slučajnog procesa (stohastičkog procesa). Tada se uz određena ograničenja na svojstva slučajnog procesa[13] osim odziva sustava na pojedinu realizaciju slučajnog procesa može odrediti i izlazni slučajni proces.

Same vrijednosti amplitude kao takve nemaju posebnog značaja kad promatramo slučajne procese već se najčešće promatraju gustoće spektra snage – pojedine realizacije procesa će imati različite amplitude, no energije i snage će biti slične među raznim realizacijama. Za takav slučaj kontinuirani sustav više ne opisujemo prijenosnom funkcijom $H(\omega)$, već prijenosnom funkcijom snage $|H(\omega)|^2 = H^*(\omega)H(\omega)$.

6.2.5. Simulacija impulsne pobude u Simulinku – Za one koji žele znati više

Impulsni odziv jednostavno dobivamo unutar MATLAB-a korištenjem naredbe `impulse`, no pregledamo li cijelu biblioteku Simulinkovih blokova možemo uočiti da ne postoji blok koji na izlazu daje Diracov $\delta(t)$. Kako možemo simulirati impulsnu pobudu unutar Simulinka?

Jedinični pobudni impuls je Diracova δ-distribucija. Važna karakteristika δ-distribucije jest da je jednaka nul-funkciji na svim intervalima koji ne sadrže ishodište, te da nema određenu vrijednost u ishodištu. Osim toga je

$$\int f(t)\delta^{(n)}(t-t_0)\,dt = (-1)^n f^{(n)}(t_0)\,,$$

no uobičajeno se kaže samo da δ-distribucija vadi vrijednost podintegralne funkcije. Pobudimo li vremenski sustav s δ-distribucijom dobivamo

$$h(t) * \delta(t) = \int h(\tau)\delta(t-\tau)\,d\tau = h(t)\,.$$

Moguća interpretacija jest da pobuda tipa Diracove δ-distribucije sustavu predaje jediničnu energiju u neizmjerno kratkom intervalu.

Kako simulirati predaju jedinične energije sustavu? Svi sustavi unutar Simulinka se rješavaju metodama numeričke integracije i pri tome moramo pažljivo odabrati parametre simulacije[14]. Obično odabrani parametri ovise o sustavu kojeg simuliramo – prema tome i simuliranu δ-distribuciju moramo odabrati u ovisnosti u sustavu kojeg simuliramo.

Pretpostavimo da simuliramo jednostavan linearan vremenski nepromjenjiv sustav. Takav sustav možemo opisati preko polova i nula, te dodatno znamo da

---

[13] Obično se zahtijeva stacionarnost drugog reda u širem smislu.

[14] Za najjednostavniji primjer pogledajte zahtjeve na period otipkavanja pri korištenju Eulerove transformacije.

svaki pol uzrokuje pojavu kompleksne eksponencijale u odzivu. Za svaki pol možemo točno odrediti kompleksnu eksponencijalu i njenu pripadnu vremensku konstantu. Zanima nas pol koji je najudaljeniji od ishodišta u s-kompleksnoj ravnini jer on određuje najkraću vremensku konstantu. Predaju jedinične energije sustavu možemo simulirati kao impulsnu pobudu trajanja značajno kraćeg od vremenske konstante najudaljenijeg pola jer time sustavu predamo traženu energiju, a zbog tromosti sustav ne može reagirati u tako kratkom intervalu[15]. Shodno tome za simulaciju impulsne pobude u Simulinku možemo koristiti bilo koji blok koji nam omogućuje kreiranje takve pobude. Kako sam oblik impulsa nije pretjerano bitan najjednostavnije je za simulaciju impulsne pobude odabrati pravokutni impuls trajanja t_i i amplitude $\frac{1}{t_i}$ (time smo osigurali jediničnu energiju).

6.3. Prikaz u prostoru stanja

Svaki memorijski element u sustavu je povezan s jednom varijablom stanja. Za direktnu realizaciju najjednostavnijeg kontinuiranog sustava drugog reda opisanog jednadžbom

$$y''(t) + 2\delta\, y'(t) + \omega_0^2\, y(t) = u(t)$$

tipično za varijable stanja biramo izlaz sustava $x_1 = y(t)$ i derivaciju izlaza $x_2 = y'(t)$.

Zadatak

5. Koristeći Simulink snimite odzive $y(t)$ te putanje u prostoru stanja $x_1 O x_2$ kontinuiranog sustava opisanog diferencijalnom jednadžbom

$$y''(t) + 2\delta\, y'(t) + \omega_0^2\, y(t) = 0$$

za $\omega_0 = 0{,}4$ i za vrijednosti parametra δ od -0.05, 0, $0{,}1$, $0{,}4$ i 1.

6.4. Frekvencijska karakteristika

6.4.1. Računanje frekvencijske karakteristike

Linearni kontinuirani sustav možemo karakterizirati prijenosnom funkcijom. Izvršimo li zamjenu $s \mapsto j\omega$ dobivamo frekvencijsku karakteristiku sustava. Dobivena frekvencijska karakteristika za sustave čija prijenosna funkcija ima realne koeficijente je konjugirano simetrična te se uobičajeno crta samo za pozitivne vrijednosti frekvencija[16]. Unutar MATLAB-a koristi se naredba `freqs`:

```
» B=[11 4 10]; <ENT>          % brojnik prijenosne funkcije
» A=[90 35 78]; <ENT>         % nazivnik prijenosne funkcije
» freqs(B,A) <ENT>            % računamo i crtamo frekvencijsku karakteristiku
```

[15] Opis nije u potpunost korektan jer egzaktna matematička formulacija navedenog značajno prelazi okvire kolegija.
[16] Amplitudna karakteristika je parna dok je fazna karakteristika neparna.

Zadatak

6. Odredite amplitudnu i faznu frekvencijsku kontinuiranog sustava opisanog diferencijalnom jednadžbom

$$y''(t) + 2\delta\, y'(t) + \omega_0^2\, y(t) = u(t)$$

za $\omega_0 = 0{,}4$ i za vrijednosti parametra δ od -0.05 i $0{,}4$.

6.4.2. Snimanje frekvencijske karakteristike

Korištenjem frekvencijske karakteristike koja nam određuje stacionarno stanje možemo odmah odrediti izlaznu amplitudu za harmonijsku pobudu oblika $u(t) = A\cos(\omega_0 t + \varphi)$. Iz frekvencijske karakteristike očitavamo pojačanje za danu frekvenciju pobude te tada računamo izlazni signal prema

$$y(t) = \left|H(\omega_0)\right| A\cos\left(\omega_0 t + \varphi + \arg\left(H(\omega_0)\right)\right).$$

No isto tako simulacijom odziva sustava na harmonijsku pobudu te očitavanjem promjene amplitude i faze možemo snimiti frekvencijsku karakteristiku točku po točku.

Primijetite da stacionarno stanje postoji samo za stabilne sustave tako da frekvencijska karakteristika nema smisla promatramo li nestabilne sustave.

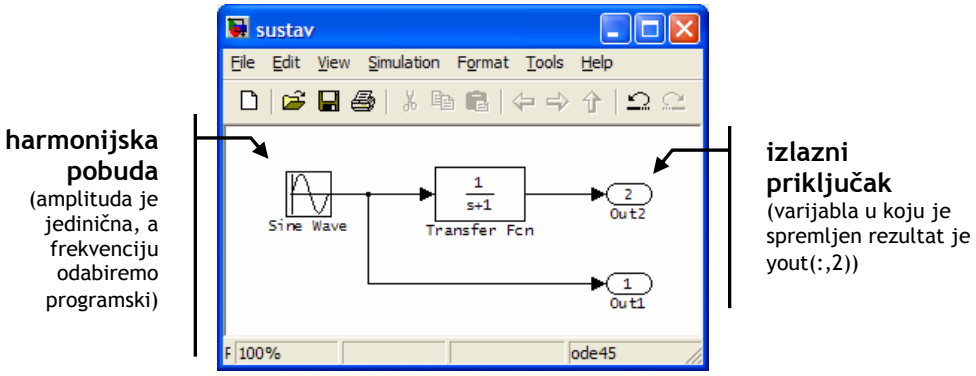
Zadatak

7. Na simulacijski dijagram postavite harmonijsku pobudu (sinus ili kosinus funkcija) te snimite par točaka amplitudne i fazne karakteristike za slučajeve kada parametar δ poprima vrijednosti -0.05 i $0{,}4$. Poklapaju li se rezultati s frekvencijskim karakteristikama određenim pomoću naredbe freqs? Možete li simulacijom odrediti frekvencijsku karakteristiku nestabilnog sustava?

6.4.3. Automatsko snimanje amplitudne i fazne frekvencijske karakteristike – Za one koji žele znati više

Amplitudna frekvencijska karakteristika je prikaz pojačanja sustava u ovisnosti o frekvenciji harmonijske pobude. Fazna amplitudna karakteristika je pak povezana s vremenom koje je potrebno za prolaz harmonijske pobude kroz sustav. Postupak simuliranog mjerenja amplitudne i fazne frekvencijske karakteristike moguće je automatizirati preko jednostavne m skripte.

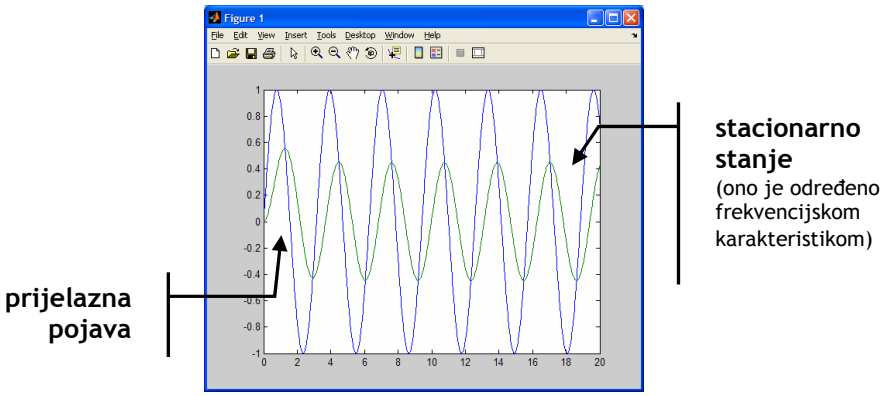
Prije pisanja skripte potrebno je sastaviti odgovarajući Simulink model linearnog vremenski nepromjenjivog sustava. Na ulaz sustava se tada spaja blok koji daje harmonijsku pobudu (npr. `Sources→Sine Wave`), dok i ulaz i izlaz moramo vratiti u radni prostor kako bi mogli odrediti pojačanje i fazni pomak. Najjednostavniji način dobivanja tih signala u radnom prostoru je dodavanje izlaznih priključaka (`Sinks→Out1`). Sastavljeni model je prikazan na slici.



Postavimo li frekvenciju u bloku `Sine Wave` na neku neodređenu vrijednost `w` koju definiramo u radnom prostoru korištenjem naredbe `sim` možemo pozivati simulaciju te zatim analizirati dobivene rezultate:

```
» w = 2; <ENT>                          % definiramo frekvenciju pobude
» sim('sustav') <ENT>                   % pokrećemo simulaciju modela 'sustav'
» whos <ENT>                            % rezultati simulacije su u varijabli yout
  Name        Size                        Bytes  Class

  tout     1000x1                          8000  double array
  w           1x1                             8  double array
  yout     1000x2                         16000  double array

Grand total is 3001 elements using 24008 bytes

» plot(tout,yout) <ENT>                 % crtamo ih
```

Nakon simuliranja modela u Simulinku dobivamo dvije nove varijable u radnom prostoru - `tout` i `yout`. U varijabli `tout` su spremljeni vremenski trenutci, dok se u varijabli `yout` nalaze vrijednosti izlaznih signala. Pri tom je `yout(:,1)` ulazni signal, a `yout(:,2)` izlazni signal jer smo tako postavili priključnice. Nacrtajmo dobivene signale za sustav $H(s)=\frac{1}{s+1}$ pri frekvenciji $\omega=2$. Iz slike jasno vidimo prijelaznu pojavu te stacionarno stanje. Određivanjem odnosa amplituda te vremenskog razmaka između prolazaka kroz nulu možemo odrediti pojačanje i fazni pomak.



Kako će računalo izmjeriti amplitudnu i faznu karakteristiku? Kako znamo kolika je pobudna frekvencija (varijabla w) na temelju vremenskih trenutaka spremljenih u varijablu `tout` možemo odabrati samo jedan period signala, i to onaj s kraja snimljenog simuliranog signala koji odgovara upravo stacionarnom stanju. Amplitudnu karakteristiku određujemo kao omjer maksimalnih vrijednosti ulaznog i izlaznog signala[17]. Faznu karakteristiku određujemo iz udaljenosti prolazaka kroz nulu (ili udaljenosti maksimuma, ili pak udaljenosti minimuma). Prisjetite se da vrijeme jednog perioda signala odgovara kutu od 2π, a vremenski razmak između istih točaka na ulaznom i izlaznom signalu faznom pomaku. Množenjem očitanog vremensko razmaka s varijablom w odmah dobivamo fazni pomak.

Sada je samo potrebno napisati MATLAB skriptu koja mijenja frekvencije te računa pojačanje i fazni pomak. Pseudokod potrebne m skripte bi bio:

```
ws = linspace(w1, w2);            % možete korisiti i logaritamsku skalu
for i = 1 : length(ws)
   w = ws(i);                     % postavljamo frekvenciju
   sim('model');                  % pokrećemo simulaciju
   n = length(tout);
   T = 2*pi/w;
   j = find(tout>tout(n)-T);      % odabiremo zadnji period
   A(i) = ...                     % računanje amplitude iz yout(j,2)
   phi(i) = ...                   % računanje faznog pomaka iz yout i tout
end
plot(ws, A);                      % crtamo amplitudnu karakteristiku
```

Da bi rezultati bili ispravni morate još podesiti parametre simulacije tako da simulacija traje značajno dulje i od perioda najsporije pobude i od vremenske konstante pola najbližeg ishodištu kompleksne ravnine.

[17] Primijetite da ako odaberemo jediničnu amplitudu omjer nije potrebno računati, već je dovoljno samo očitati amplitudu izlaznog signala.

7. Vježba 7. – Nelinearni kontinuirani sustav (bistabil)

7.1. Priprema

Pročitajte poglavlje 4.5. "Nelinearan sustav" iz skripte "Signali i sustavi" koju možete pronaći na WWW stranicama predmeta (http://sis.zesoi.fer.hr/). Iz "Zbirke riješenih zadataka iz signala i sustava" pročitajte poglavlja 3.3. "Aproksimacija U/I karakteristike linearnim segmentima" i 7. "Nelinearni sustavi" (http://sis.zesoi.fer.hr/vjezbe/pdf/sis_zbirka_20040914.pdf).

7.2. Blokovi potrebni za simulaciju

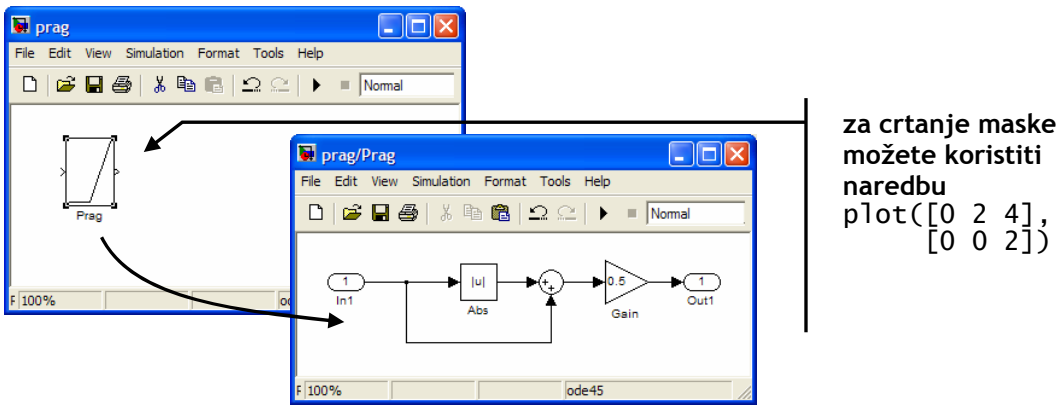
7.2.1. Funkcijski blok prag

Funkcijski blok prag propušta signal samo kada je on veći od nule i određen je izrazom

$$y(t) = \begin{cases} x(t), & x(t) > 0 \\ 0, & \text{inače} \end{cases}.$$

Možemo ga jednostavno konstruirati koristeći apsolutnu vrijednost realizacijom izraza

$$y(t) = \frac{1}{2}\left(x(t) + |x(t)|\right).$$



Zadatak

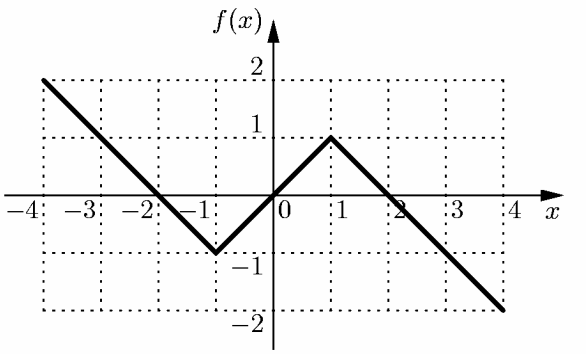
1. Realizirajte funkcijski blok prag u Simulinku.

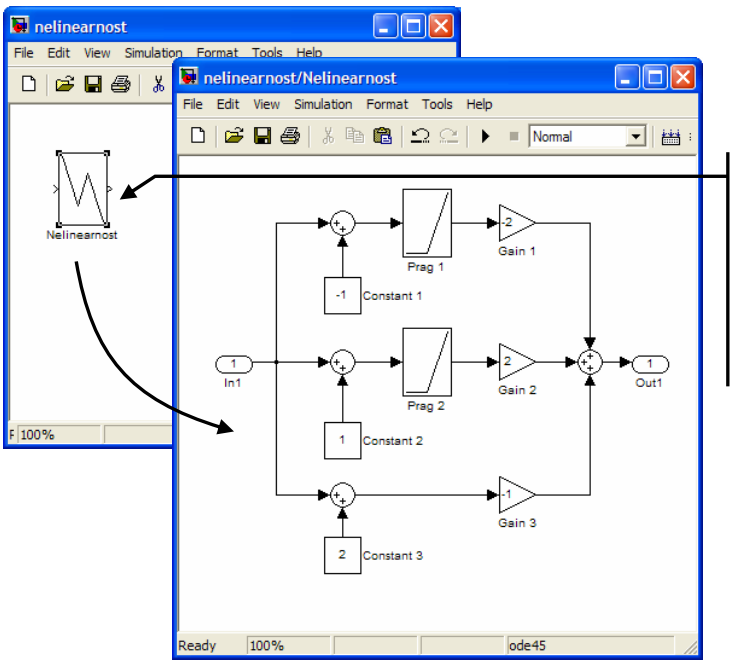
7.2.2. Nelinearnost za povratnu granu

Nelinearnost za povratnu granu je određena izrazom

$$f(x)=\begin{cases}-x+2, & 1\le x\\ x, & -1<x<1\\ -x-2, & x\le 1\end{cases}.$$

i prikazana je na slici. Zadanu nelinearnost realiziramo korištenjem prethodno realiziranog bloka prag kako je prikazano na slici.





za crtanje maske možete koristiti naredbu

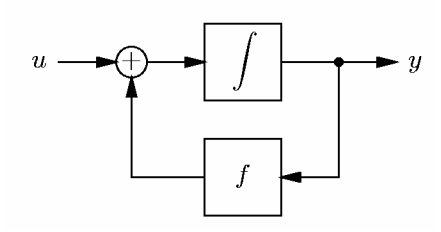
```
plot([0 2 4 6],
     [2 -1 1 -2])
```

Zadatak

2. Realizirajte zadanu karakteristiku u Simulinku.

7.3. Bistabil

Nelinearni sustav prvog reda kojemu je povratna grana nelinearna karakteristika realiziranu u zadatku dva ima dva stabilna stanja (bistabil).



Zadaci

3. Realizirajte zadani nelinearni sustav u Simulinku. Pobuda sustava će biti impulsi stalne amplitude i promjenjivog trajanja koje možete definirati unutar bloka *Signal Builder*.
4. Snimite impuls na ulazu i odziv sustava te trajektorije $y' = f(y)$ i $y = f(y')$. Neka je amplituda ulaznog impulsa $U = 1.5$ V a trajanje T neka poprima vrijednosti $0{,}8$, $2{,}16$, $2{,}24$, $4{,}8$ i 20 (dakle simulirate pet različitih situacija gdje je trajanje pojedinog pobudnog impulsa iz zadanog skupa). Početno stanje integratora postavite u -2. Prilikom simulacije postavite bolju relativnu toleranciju da bi dobili ispravne rezultate (za ode45 tolerancija najviše 10^{-6}).
5. Opišite ponašanje modeliranog sustava ako držimo trajanje pobudnog impulsa stalnim, a mijenjamo mu amplitudu!

8. Vježba 8. – Frekvencijska analiza vremenski kontinuiranih signala

8.1. Priprema

Pročitajte bilješke s predavanja "Frekvencijska analiza vremenski kontinuiranih signala" koje možete pronaći na WWW stranicama predmeta (http://sis.zesoi.fer.hr/).

Pročitajte dio o simboličkoj matematici u MATLAB-u iz "Kratkih uputa za korištenje MATLAB-a" koje se mogu pronaći na WWW stranicama predmeta (http://sis.zesoi.fer.hr/vjezbe.html#dodatci).

Pripremni zadatak

1. Odredite rastav u Fourierov red signala

$$x_1(t) = 220 \cdot \cos(100\pi t)$$

i signala

$$x_2(t) = 220 \cdot \cos(100\pi t) + 50 \cdot \cos(300\pi t + \pi/3).$$

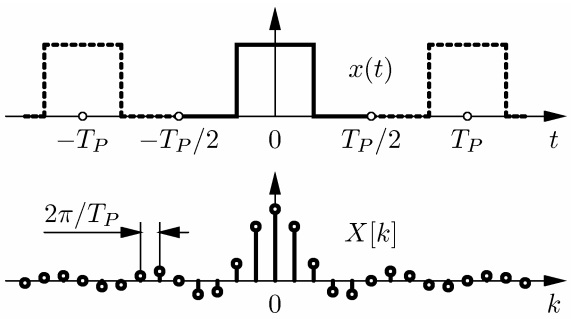
8.2. Fourierov red

Fourierov red se koristi za prikaz periodičkih kontinuiranih signala i određen je izrazima

$$X[k] = \frac{1}{T_P} \int_{T_P} x(t) e^{-j\omega_P k t}\, dt$$

$$x(t) = \sum_{k=-\infty}^{+\infty} X[k] e^{j\omega_P k t}$$

Pri tome je T_P period signala, dok je ω_P kružna frekvencija povezana s periodom preko izraza $2\pi/T_P = \omega_P$. Primjer rastava periodičkog signala u Fourierov red dan je na slici.



Za određivanje Fourierovog reda u MATLAB-u koristimo Symbolic Toolbox.

```
» syms t Tp k <ENT>                                  % potrebne varijable
» int(1*exp(-2*pi*j/Tp*t*k),-Tp/4,Tp/4)/Tp <ENT>    % integracija

ans =

1/2*i*(exp(-1/2*i*pi*k)-exp(1/2*i*pi*k))/pi/k

» pretty(simplify(ans)) <ENT>                        % uljepšavanje rezultata

                                  sin(1/2 pi k)
                                  -------------
                                      pi k
```

Zadatak

1. Koristeći MATLAB odredite rastav u Fourierov red signala

$$x_1(t) = 220 \cdot \cos(100\pi t)$$

i signala

$$x_2(t) = 220 \cdot \cos(100\pi t) + 50 \cdot \cos(300\pi t + \pi/3).$$

Interpretirajte dobivene spektre.

8.2.1. Gustoća spektra snage

Periodički kontinuirani signali imaju beskonačnu energiju i konačnu srednju snagu. Štoviše, kako su signali periodički unutar jednog perioda srednja snaga je uvijek jednaka. Želimo li izračunati snagu iz spektra koristimo Parsevalovu relaciju

$$P = \frac{1}{T_P} \int_{T_P} |x(t)|^2 \, dt = \sum_{k=-\infty}^{+\infty} |X[k]|^2 \, .$$

Pri tome $|X[k]|^2$ predstavlja srednju snagu k-te harmoničke komponente signala. Prikažemo li $|X[k]|^2$ kao funkciju frekvencije $k\omega_P$ dobivamo gustoću spektra snage.

Zadatak

2. Koristeći MATLAB odredite rastav u Fourierov red signala niza pravokutnih impulsa jedinične amplitude trajanja T koji se ponavljaju svakih T_P pri čemu je $T_P > T$. Za taj signal odredite spektar, gustoću spektra snage i srednju snagu. Nacrtajte dobivene funkcije za $T = 1$ i $T_P = 20$ te $T = 1$ i $T_P = 2$.

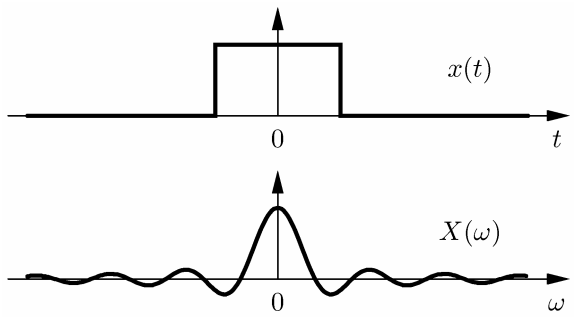
8.3. Fourierov integral

Fourierov integral (Fourierova transformacija) se koristi za prikaz bilo kakvih kontinuiranih signala i određen je izrazima

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t}\, dt$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega) e^{j\omega t}\, d\omega$$

Primjer rastava jednog kontinuiranog signala dan je na slici.



Za računanje Fourierove transformacije u MATLAB-u možete koristiti funkciju `fourier` ili alternativno računati izravno definicijske izraze:

```
» fourier(sym('Heaviside(t)')) <ENT>        % računamo Fourierov transformaciju
                                            % step funkcije
ans =

pi*Dirac(w)-i/w

» fourier(sym('Heaviside(t-1)-Heaviside(t+1)')) <ENT>

ans =

exp(-i*w)*(pi*Dirac(w)-i/w)-exp(i*w)*(pi*Dirac(w)-i/w)

» simplify(ans) <ENT>

ans =

-2*sin(w)/w
```

Zadaci

3. Koristeći MATLAB odredite Fourierovu transformaciju signala

$$x_1(t) = 220 \cdot \cos(100\pi t)$$

i signala

$$x_2(t) = 220 \cdot \cos(100\pi t) + 50 \cdot \cos(300\pi t + \pi/3).$$

Interpretirajte dobivene spektre. Usporedite rezultat s rezultatom 1. zadatka[18].

[18] Primijetite da bi očekivali spektar u kojem se javljaju linije amplitude koja odgovara upravo polovini amplitude zadanog kosinusa jer rastavljamo signal u kombinaciju eksponencijalnih

4. Koristeći MATLAB odredite Fourierovu transformaciju pravokutnog impulsa jedinične amplitude trajanja T. Usporedi dobivenu transformaciju s rastavom periodičkog niza impulsa u Fourierov red iz drugog zadatka.

8.3.1. Gustoća spektra energije

Za periodičke signale smo računali srednju snagu unutar perioda. Za općeniti kontinuirani signal snagu unutar nekog intervala možemo odrediti kao

$$P = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} |x(t)|^2 \, dt \, .$$

Želimo li pomoću danog izraza odrediti srednju snagu cijelog signala za većinu signala dobili bi nulu. Stoga se kontinuirani aperiodski signali i signali konačnog trajanja obično opisuju preko energije. Za određivanje energije signala koristimo Parsevalovu relaciju

$$E = \int_{-\infty}^{+\infty} |x(t)|^2 \, dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |X(\omega)|^2 \, d\omega \, .$$

Veličina $X(\omega)X^*(\omega) = |X(\omega)|^2$ nam predstavlja gustoću spektra energije.

Zadatak

5. Koristeći MATLAB odredite gustoću spektra energije i ukupnu energiju pravokutnog impulsa jedinične amplitude trajanja T.

---

funkcija (funkcije kosinus ili sinus se mogu zapisati kao zbroj dvije eksponencijale). Umjesto toga dobili smo spektralne linije amplitude koja odgovara amplitudi kosinusa pomnoženoj s π. Razlog je u tome što se konstanta $1/(2\pi)$ nalazi uz integral inverzne transformacije kao što je uobičajeno u elektrotehnici, a ne uz integral za računanje transformacije.

9. Vježba 9. – Frekvencijska analiza vremenski diskretnih signala

9.1. Priprema

Pročitajte bilješke s predavanja "Frekvencijska analiza vremenski diskretnih signala" koje možete pronaći na WWW stranicama predmeta (http://sis.zesoi.fer.hr/).

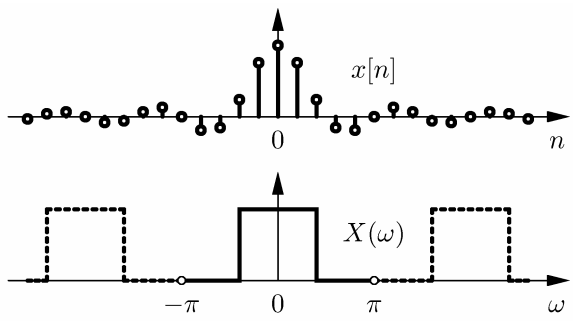
9.2. Vremenski diskretna Fourierova transformacija

Vremenski diskretna Fourierova transformacija (DTFT – *Discrete-Time Fourier Transform*) se koristi za prikaz nizova. Određena je izrazima:

$$X(\omega) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n}$$

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{+\pi} X(\omega)e^{j\omega n}\,d\omega \,.$$

Primjer rastava jednog niza dan je na slici.



Prilikom obrade i analize signala na računalu najčešće radimo sa signalima konačnog trajanja te računanje sume eksponencijalnih funkcija ne predstavlja osobit problem. Za crtanje spektra u MATLAB-u možemo koristiti funkciju `freqz`:

```
» n=[1 2 3 4 3 2 1] <ENT>          % definiramo konačni niz brojeva (signal)

n =

     1     2     3     4     3     2     1

» freqz(n,1) <ENT>                 % crtamo frekvencijsku karakteristiku
                                   % primijetite da je naredba predviđena za
                                   % računanje frekvencijske karakteristike te
                                   % kao nazivnik moramo zadati jedinicu
```

Korištenjem naredbe `freqz` na jednostavan način možemo odrediti samo spektar niza kojemu prvi član uvijek započinje u nuli. Želimo li odrediti spektar

konačnog niza kojemu prvi član nije u koraku $n=0$ moramo se poslužiti teoremom o pomaku.

$$x[n-n_0] \circ\!\!-\!\!\bullet\, X(\omega)e^{-j\omega n_0}.$$

Unutar MATLAB-a koristimo funkciju `freqz` za numeričko računanje karakteristike koju tada pomnožimo s $e^{-j\omega n_0}$:

```
» n = [1 1 1 1 1 1 1]; <ENT>          % definiramo konačni niz brojeva (signal)
» [H, w] = freqz(n,1); <ENT>          % računamo DTFT
» H = H .* exp(-j*w*5); <ENT>         % primijenjujemo teorem o pomaku
» plot(w, abs(H)) <ENT>               % crtamo karakteristiku
```

Zadatak

1. Koristeći MATLAB odredite vremenski diskretnu Fourierovu transformaciju pravokutnog impulsa jedinične amplitude koji ima $N=10$ uzoraka[19]. Kako se spektar mijenja ako mijenjamo trajanje N i korak u kojem počinje impuls?
   Naredbi `freqz` možete zadati niz frekvencija na kojima se računa vrijednost karakteristike. Sve frekvencijske karakteristike odredite na intervalu $[-5\pi, 5\pi]$.

9.2.1. Gustoća spektra energije

Za određivanje energije signala koristimo Parsevalovu relaciju

$$E=\sum_{n=-\infty}^{+\infty}|x[n]|^2=\frac{1}{2\pi}\int_{-\pi}^{+\pi}|X(\omega)|^2\,d\omega.$$

Veličina $X(\omega)X^*(\omega)=|X(\omega)|^2$ nam predstavlja gustoću spektra energije.

Zadatak

2. Odredite gustoću spektra energije i ukupnu energiju signala iz prošlog zadatka.

9.2.2. Veza s Z transformacijom – Za one koji žele znati više

Z transformaciju možemo promatrati kao poopćenje vremenski diskretne Fourierove transformacije. Određena je izrazima:

$$X(z)=\sum_{n=-\infty}^{+\infty}x[n]z^{-n}$$
$$x[n]=\frac{1}{2\pi j}\oint_C X(z)z^n\frac{dz}{z}.$$

Usporedimo li izraze za Z transformaciju s izrazima za DTFT vidimo da DTFT odgovara Z transformaciji ako se ograničimo samo na jediničnu kružnicu u kompleksnoj z-ravnini, odnosno ako uzmemo

[19] Signal se sastoji od N jedinica u nizu.

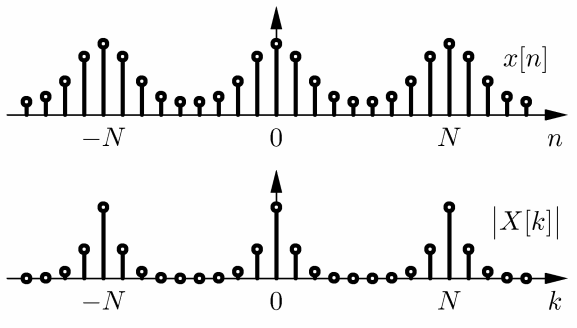
$$z = e^{j\omega}.$$

9.3. Diskretna Fourierova transformacija

Periodički niz možemo prikazati koristeći diskretni Fourierov red (DFS – *Discrete Fourier Series*):

$$\begin{aligned} X[k] &= \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-2\pi jnk/N} \\ x[n] &= \sum_{k=0}^{N-1} X[k]e^{2\pi jnk/N} \end{aligned}.$$

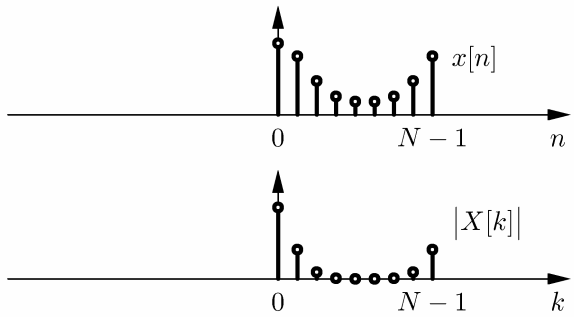
Jedan periodički niz i pripadni amplitudni spektar prikazani su na slici.



Kako u izrazima za DFS utjecaj na transformaciju imaju samo uzorci unutar jednog perioda (ostale uzorke znamo zbog periodičnosti) sama transformacija je pogodna za računalne primjene. Za obradu signala na računalu raspolažemo tipično s N snimljenih uzoraka otipkanih s nekom frekvencijom otipkavanja f_s. Pretpostavimo li da je tih N snimljenih uzoraka upravo jedan period periodičkog signala možemo odrediti DFS. Ako se striktno ograničimo na jedan period transformacija se obično naziva diskretna Fourierova transformacija i određena je s parom

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk}, \quad 0 \le k \le N-1 \\ x[n] &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-nk}, \quad 0 \le n \le N-1 \end{aligned}.$$

Pri tome se kompleksna eksponencijala uobičajeno označava s $W_N^{nk} = e^{-2\pi jnk/N}$. Primjer DFT transformacije dan je na slici.

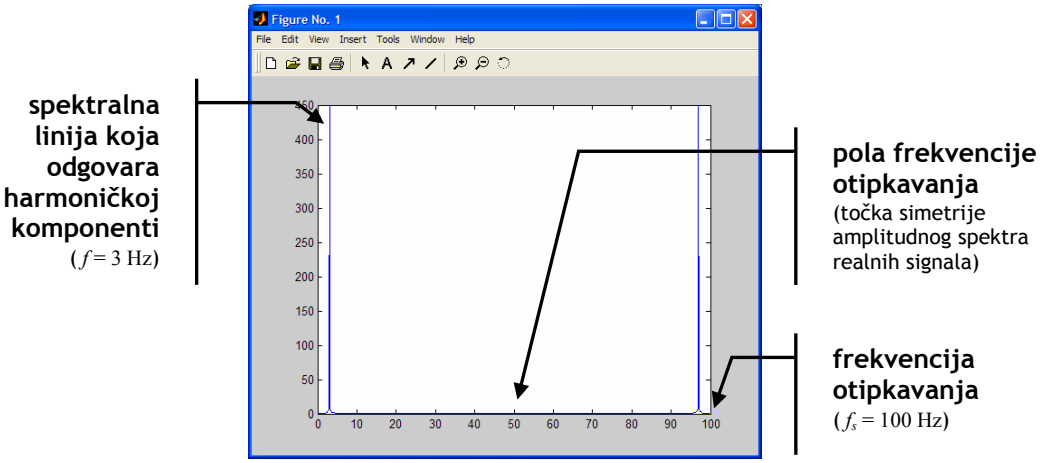


Koeficijenti transformacije $X[k]$ su funkcija broja k. Zbog analize poželjno je uspostaviti vezu između indeksa k i stvarne frekvencije u Hz. Promatrani signal je snimljen u N točaka frekvencijom otipkavanja f_s. Uzorak spektra s indeksom k odgovara stvarnoj frekvenciji od

$$k\frac{f_s}{N}\,\mathrm{Hz}\,.$$

Za računanje diskretne Fourierove transformacije koristimo funkciju `fft`:

```
» t = [1:0.01:10]; <ENT>          % definiramo vektor vremenski trenutaka
» x = sin(2*pi*3*t); <ENT>        % otipkavamo sinus frekvencije 3 frekvencijom 100
» X = fft(x); <ENT>               % računamo DFT
» A = abs(X); <ENT>               % računamo amplitudni spektar
» N = length(X); <ENT>            % broj uzoraka je N
» w = ([1:N]-1)/N*100; <ENT>      % računamo vektor stvarnih frekvencija
» plot(w, A) <ENT>                % crtamo spektar
```



Zadaci

6. Koristeći MATLAB otipkajte funkciju

$$x_1(t) = \sin(2\pi 800t)$$

uz frekvenciju otipkavanja $f_s = 8\ \mathrm{kHz}$ u $N = 8000$ uzoraka. Slušajte otipkani signal[20]. Nacrtajte amplitudni i fazni spektar dobivenog signala.

7. Otipkajte sada funkciju

$$x_2(t) = \sin(2\pi 805 t)$$

uz frekvenciju otipkavanja $f_s = 8\ \mathrm{kHz}$ u $N = 8000$ uzoraka. Preslušajte otipkani signal. Nacrtajte amplitudni i fazni spektar dobivenog signala.

8. Razmotrite signal koji odgovara superpoziciji (zbroju) signala $x_1(t)$ i $x_2(t)$. Preslušajte ga. Nacrtajte signal u vremenskoj domeni. Kako se izgled signala poklapa s onim što čujete? Nacrtajte amplitudni i fazni spektar. U kakvom su odnosu dobiveni spektri prema spektrima iz prva dva zadatka.
9. Otipkajte sada funkciju

$$x_3(t) = \sin(2\pi 800 t^2)$$

uz frekvenciju otipkavanja $f_s = 8\ \mathrm{kHz}$ u $N = 8000$ uzoraka. Preslušajte otipkani signal. Što čujete? Nacrtajte signal u vremenskoj domeni. Kako se izgled signala poklapa s onim što čujete? Nacrtajte amplitudni i fazni spektar dobivenog signala.

9.3.1. FFT algoritam – Za one koji žele znati više

Diskretnu Fourierovu transformaciju računamo kao

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad 0 \le k \le N-1 .$$

Kako je W_N^{nk} kompleksni broj za računanje jednog uzorka potrebno je N kompleksnih množena i N kompleksnih zbrajanja. Kako spektar ima ukupno N uzoraka broj potrebnih operacija je $2N^2$ što odgovara složenosti $O(N^2)$. FFT algoritam (FFT – *Fast Fourier Transform*) je postupak za efikasno računanje diskretne Fourierove transformacije koji se temelji grupiranju uzoraka te iskorištavanju svojstava periodičnosti i simetrije kompleksne eksponencijale[21]. Ti postupci su primjenjivi samo ako se duljina ulaznog niza može rastaviti u produkt prostih faktora. Za efikasno izvršenje FFT algoritam u MATLAB-u zahtijeva da duljina ulaznog niza bude potencija broja dva, te je u tom slučaju složenost algoritma $O(N\log(N))$.

9.4. Spektrogram

Signal $x_3(t) = \sin(2\pi 800 t^2)$ ima trenutnu kružnu frekvenciju

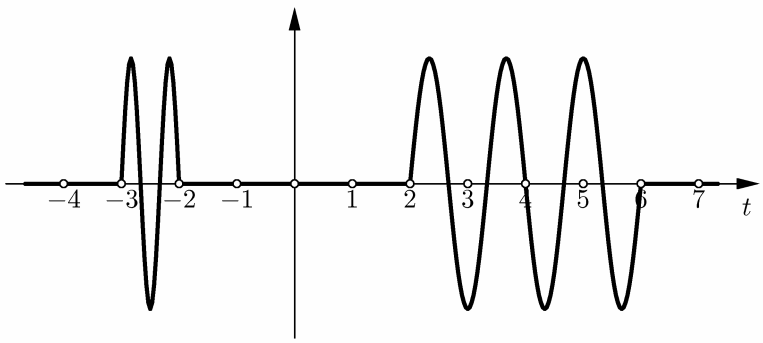
$$\omega(t) = \frac{d}{dt}\left(2\pi 800 t^2\right) = 4\pi 800 t$$

[20] Signal možete preslušati korištenjem naredbe `soundsc`. Ako imate uzorke spremljene u vektoru `x` otipkane frekvencijom `fs` naredba je `soundsc(x, fs)`.

[21] Za kompleksnu eksponencijalu vrijedi $W_N^{2r} = W_{N/2}^{r}$ i $W_N^0 = 1$, $W_N^{N/2} = -1$, $W_N^{N/4} = -j$ i $W_N^{3N/4} = j$.

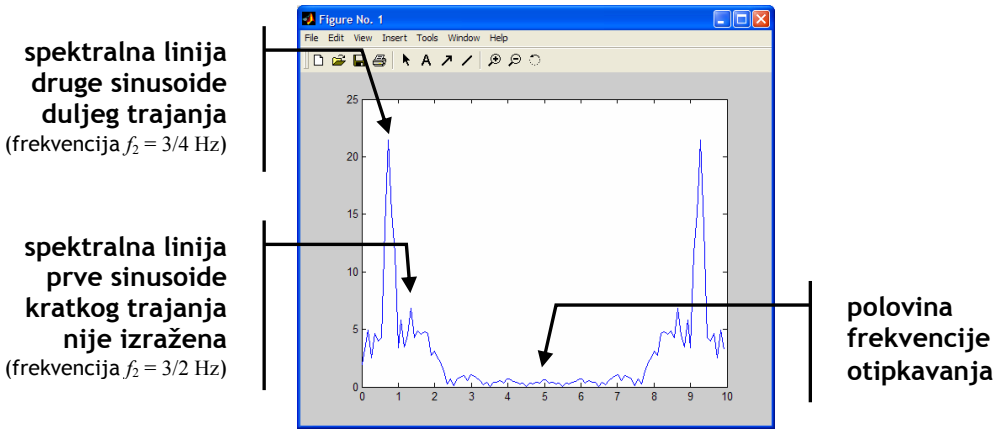
koja raste s vremenom. Kada smo odredili spektar tog signala u prošlom zadatku vidjeli smo da se javljaju sve frekvencije od nule do $1600t$ Hz. Primijetite da iz dobivenog spektra možemo odrediti koje su frekvencije prisutne u signalu no da ih ne možemo lokalizirati u vremenu, odnosno ne možemo odrediti kada se koja frekvencija javlja.

Razmotrimo signal koji se sastoji od dvije lokalizirane sinusoide jedinične amplitude kako je prikazano slikom.



Prva sinusoida ima frekvenciju $f_1 = 3/2$ Hz dok druga sinusoida ima frekvenciju $f_2 = 3/4$ Hz. Otipkajmo signal te odredimo spektar u MATLAB-u:

```
» t = [-4:0.1:7]; <ENT>                    % definiramo trenutke uzorkovanja
» t1 = [-3:0.1:-2]; <ENT>                  % vektor vremena za prve sinusoidu
» x1 = sin((t1-3)*3*pi); <ENT>             % otipkavamo prvu sinusoidu
» t2 = [2:0.1:6]; <ENT>
» x2 = sin((t2-2)*3*pi/2); <ENT>           % otipkavamo drugu sinusoidu
» x = 0*t; <ENT>                           % signal je svugdje jednak nuli
» x(1/0.1:2/0.1) = x1; <ENT>               % dodajemo u signal prvu sinusoidu
» x(6/0.1:10/0.1) = x2; <ENT>              % dodajemo u signal drugu sinusoidu
» plot(t,x) <ENT>                          % crtamo signal
» S = fft(x); <ENT>                        % računamo DFT
» N = length(x);
» w = ([1:N]-1)/N*10;
» plot(w, abs(S)) <ENT>                    % crtamo amplitudni spektar
```



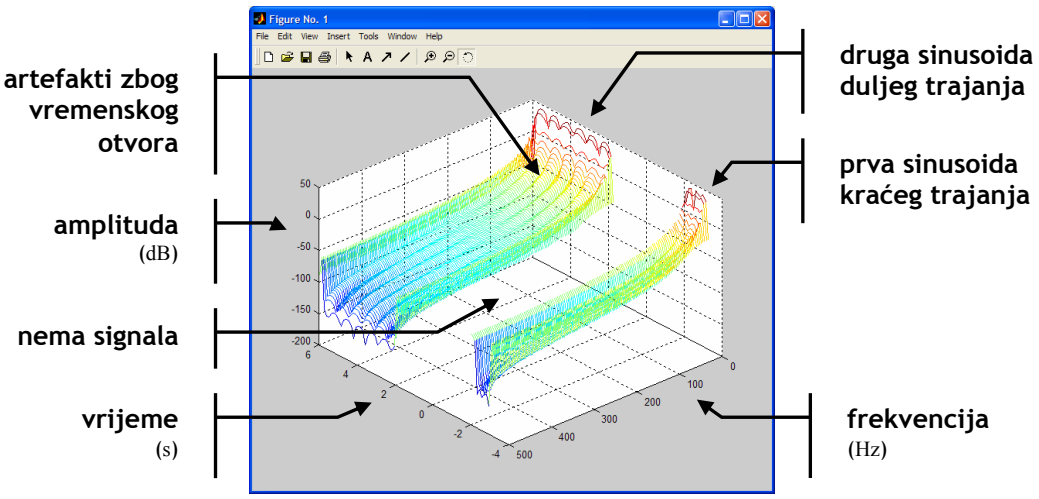
Dobiveni spektar nije pogodan za analizu promatranog signala. Iz njega možemo očitati dominantnu komponentu frekvencije $f_1 = 3/4\ \mathrm{Hz}$, no pitanje je gdje se izgubila druga komponenta? Naime, promatramo li spektar signala koji je linearna kombinacija dvije čiste nelokalizirane sinusoide jasno vidjeli bi dvije spektralne linije.

Ograničavanje sinusoida na konačni vremenski interval možemo modelirati množenjem s vremenskim otvorom. U ovom slučaju pomnožili smo ih s funkcijom rect,

$$x(t) = x_1(t)\,\mathrm{rect}\left(t + \frac{2}{5}\right) + x_2(t)\,\mathrm{rect}\left(\frac{t-4}{4}\right).$$

Kako množenje u vremenskoj domeni odgovara konvoluciji u frekvencijskoj domeni, spektar svake sinusoide je postao razmazan sinc funkcijom. Nadalje, kako su dva pravokutna otvora različitog trajanja amplitude sinc funkcija su različite. U našem slučaju razlika je tolika da jednu frekvencijsku komponentu više ne možemo uočiti.

Za rješavanje ovih problema prije računanja spektra možemo umjetno lokalizirati signal u vremenskoj domeni množenjem s vremenskim otvorom[22]. Odredimo li spektar takvog signala znamo da se prisutne frekvencije javljaju samo unutar intervala određenog vremenskim otvorom. Vremenski otvor sada pomičemo preko signala i za svaki položaj otvora računamo spektar. Nacrtamo li tako dobivene spektre zajedno dobivamo spektrogram. Spektrogram za dvije lokalizirane sinusoide prikazan je na slici.



Unutar MATLAB-a već postoji funkcija `specgram` za računanje i crtanje spektrograma, no za potrebe vježbi napisane su funkcije `spektrogram` i

[22] Formalnom razradom ove ideje dolazimo do vremenski kratkotrajne Fourierove transformacije (STFT – *Short-Time Fourier Transform*).

`spektrogram3d` kao jednostavan primjer kako se računa spektrogram. Kod tih funkcija možete pogledati u dodatku. Funkcije se koriste kako slijedi:

```
» t=[0:1/8000:1]; <ENT>                          % trenutci uzorkovanja
» x=cos(2*pi*800*t)+cos(2*pi*805*t); <ENT>       % signal
» spektrogram(x, 8000) <ENT>                     % crtamo spektrogram u 2D
» spektrogram3d(x, 8000) <ENT>                   % crtamo spektrogram u 3D
» specgram(x, 128, 8000) <ENT>                   % crtamo spektrogram pomoću
                                                 % ugrađene MATLAB funkcije
```

Zadaci

10. Odredite i prikažite spektrogram signal iz zadatka 9.,

$$x_3(t) = \sin(2\pi 800 t^2).$$

    Kako se izgled spektrograma poklapa s onim što čujete? Možete li iz spektrograma odrediti trenutnu frekvenciju signala i njenu promjenu?

11. Odredite i prikažite spektrogram signal iz zadatka 8. (superpozicija signala $x_1(t)$ i $x_2(t)$). Kako se izgled spektrograma poklapa s onim što čujete?

    Superpozicijom dvije bliske sinusoide dobili smo udare koje jasno čujemo. Iskoristimo li izraz za rastav zbroja

$$\sin(\omega_1 t) + \sin(\omega_2 t) = 2\sin\left(\frac{\omega_1 + \omega_2}{2} t\right)\cos\left(\frac{\omega_1 - \omega_2}{2} t\right) = 2\sin(\omega_C t)\cos(\omega_\Delta t)$$

    možemo odrediti frekvenciju udara ω_Δ. Uočavate li udare na spektrogramu? Možete li iz spektrograma očitati frekvenciju udara ω_Δ?

10. Vježba 10. – Uzorkovanje i preklapanje spektra

10.1. Priprema

Pročitajte bilješke s predavanja "Digitalna obradba kontinuiranih signala" koje možete pronaći na WWW stranicama predmeta (http://sis.zesoi.fer.hr/).

Pripremni zadatak

1. Uredno rukom na čistom papiru napišite Nyquistov kriterij te teorem otipkavanja[23]. Promatramo li čisti harmonijski signal frekvencije $8\,\mathrm{kHz}$ koje su dozvoljene vrijednosti perioda otipkavanja, a koje frekvencije otipkavanja da ne dođe do preklapanja spektra?

10.2. Uzorkovanje i preklapanje spektra

Signal $x(t) = \sin(2\pi f t^2)$ ima trenutnu kružnu frekvenciju koja linearno ovisi o vremenu t prema izrazu

$$\omega(t) = \frac{d}{dt}\left(2\pi f t^2\right) = 4\pi f t\,.$$

Dobivena kružna frekvencija je naravno u radijanima po sekundi. Otipkamo li taj signal dobivamo signal

$$x[n] = \sin(2\pi f n^2 T^2)\,,$$

gdje je T period otipkavanja.

Osim prikazanog signala koristiti ćemo isti signal s trokutastom ovojnicom definiran s

$$x(t) = w(t)\sin(2\pi f t^2), \quad 0 \le t \le T_{\max}\,,$$

gdje je

$$w(t) = \begin{cases} 1 - \dfrac{2}{T_{\max}}\dfrac{\left|2t - T_{\max}\right|}{2}, & 0 \le t \le T_{\max} \\ 0, & \text{inače} \end{cases}$$

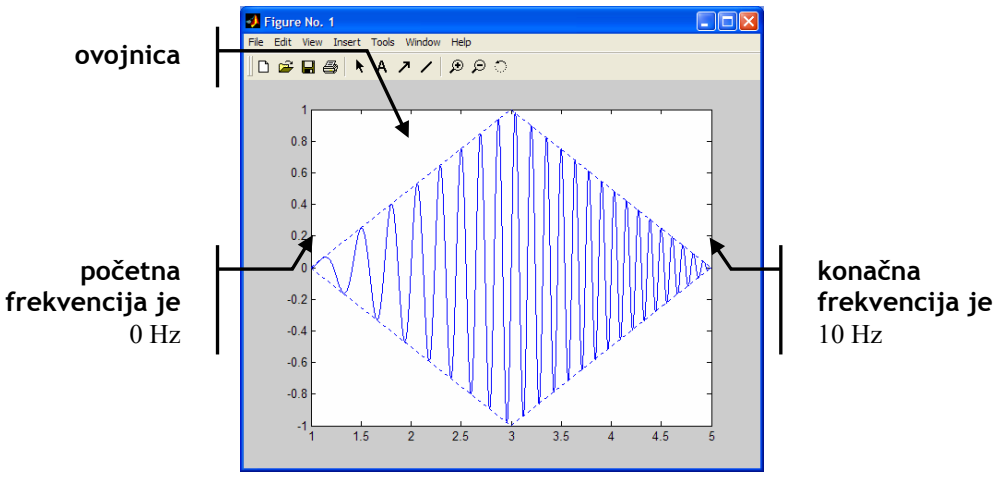
trokutasti vremenski otvor. U MATLAB-u takav signal možemo kreirati s naredbama:

```
» t = [1:1/8000:5]'; <ENT>          % definiramo vektor vremena - kako koristimo
                                    % funkciju triang za defiranje trokutnog
                                    % signala vektor mora biti vektor-stupac
```

[23] To su stvari koje svakako trebate znati. Nakon odrađenih laboratorijskih vježbi možete pripremu zalijepiti na vaš kuhinjski hladnjak ili zid u sobi, te, svaki put kada vas netko pita «Što je to?», objasnite mu ☺. To je najbrži način da naučite Nyquistov kriterij.

```
» x = sin(2*pi*t.*t); <ENT>          % definiramo signal
» x = x.*triang(length(x)); <ENT>    % množimo ga s otvorom
» plot(t,x) <ENT>                    % crtamo signal
» hold on <ENT>                      % ucrtavamo ovojnicu
» plot(t,triang(length(x)),':'); <ENT>
» plot(t,-triang(length(x)),':'); <ENT>
```

Dobiveni signal je prikazan na slici. Vidimo trokutastu ovojnicu i signal promjenjive frekvencije kojemu se frekvencija mijenja od $0\ \mathrm{Hz}$ do $10\ \mathrm{Hz}$.



Zadaci

1. Otipkajte kontinuirani signal trajanja deset sekundi kojemu se frekvencija linearno mijenja od $0\ \mathrm{Hz}$ do $12\ \mathrm{kHz}$ frekvencijom otipkavanja od $8\ \mathrm{kHz}$. Preslušajte tako dobiveni signal i objasnite artefakte koje čujete.
2. Otipkajte kontinuirani signal kojemu se frekvencija linearno mijenja od $0\ \mathrm{Hz}$ do $2500\ \mathrm{Hz}$ u točno $2^{14} = 8192$ točaka frekvencijom otipkavanja od $8\ \mathrm{kHz}$ (potencija broja dva je odabrana zbog FFT algoritma). Tako otipkani signal pomnožite s trokutnim vremenskim otvorom koristeći funkciju `triang`. Preslušajte tako dobiveni signal. Javljaju li se sada artefakti?
   Signal s trokutastom ovojnicom ćete koristiti kroz sve ostale zadatke.
3. Koristeći diskretnu Fourierovu transformaciju izračunajte i nacrtajte amplitudni i fazni spektar signala iz prethodnog zadatka. Spektre crtajte u intervalu $[-4\ \mathrm{kHz}, 4\ \mathrm{kHz}]$.
4. Promijenite signal tako da se frekvencija linearno mijenja od $0\ \mathrm{Hz}$ do $5000\ \mathrm{Hz}$. Preslušajte takav signal. Čujete li artefakte? Odredite i nacrtajte DFT amplitudni i fazni spektar na intervalu $[-4\ \mathrm{kHz}, 4\ \mathrm{kHz}]$. Možete li u spektru uočiti artefakte?

10.3. Pretipkavanje i podtipkavanje

Pretpostavimo da smo otipkali neki signal s poznatom frekvencijom otipkavanja. Postavlja se pitanje možemo li promijeniti frekvenciju otipkavanja ako raspolažemo samo sa otipkanim signalom.

Uzmemo li iz otipkanog niza svaki i-ti uzorak promijenili smo frekvenciju otipkavanja, tj. smanjili smo je. Postupak možemo opisati s izrazom

$$y[n] = x[in].$$

Opisani postupak je najjednostavniji slučaj podtipkavanja, odnosno smanjivanja frekvencije otipkavanja. Dodamo li još u postupak i filtraciju s ciljem izbjegavanja preklapanja spektra dobili smo sustav za smanjivanje frekvencije otipkavanja ili decimator. Postupak se zove decimacija.

Dodamo li između svaka dva susjedna uzorka ulaznog signala jednak broj nula opet smo promijenili frekvenciju otipkavanja, tj. povećali smo je. Postupak možemo opisati izrazom

$$y[n] = \begin{cases} x[n/i], & n/i \in \mathbb{N} \\ 0, & \text{inače} \end{cases}.$$

Takav postupak je najjednostavniji slučaj pretipkavanja. Dodamo li nakon pretipkavanja filtar koji rekonstruira vrijednosti uzoraka postavljenih u nulu koji se nalaze između dva uzorka s poznatim vrijednostima dobivamo interpolator ili sustav za povećavanje frekvencije otipkavanja. Postupak se zove interpolacija.

Zadaci

5. Promotrimo opet signal iz 2. zadatka. Podtipkavanjem kreirajte od tog signala otipkanog s $8\,\text{kHz}$ novi signal koji je otipkan s $4\,\text{kHz}$ (*downsampling*). Odredite i nacrtajte amplitudni i fazni spektar novog signala. Imajte u vidu da je frekvencija otipkavanja promijenjena te sada spektar crtate na intervalu $[-2\,\text{kHz}, 2\,\text{kHz}]$.
   Da li je došlo do preklapanja spektra? Zašto?
6. Jednostavnim pretipkavanjem dodavanjem nula od signala iz 2. zadatka kreirajte novi signal koji je otipkan s frekvencijom otipkavanja od $16\,\text{kHz}$ (*upsampling*). Odredite i nacrtajte amplitudni i fazni spektar novog signala. Imajte u vidu da je frekvencija otipkavanja promijenjena te sada spektar crtate na intervalu $[-8\,\text{kHz}, 8\,\text{kHz}]$. Objasnite dobiveni spektar. Preslušajte dobiveni signal[24]. Kako se zvuk koje čujete odnosi prema spektru kojeg ste nacrtali?

[24] Kako je signal sada otipkan s drugom frekvencijom otipkavanja ne zaboravite zadati ispravnu frekvenciju naredbi `soundsc`, u ovom slučaju `soundsc(x, 16000)`.

Zadatak – Za one koji žele znati više

7. Koristeći naredbu `butter` u MATLAB-u dizajnirajte Butterworthov NP filtar[25] kojim ćete iz pretipkanog signala frekvencije otipkavanja $16\ \mathrm{kHz}$ dobiti visokokvalitetnu reprezentaciju signal iz drugog zadatka, no sada s frekvencijom otipkavanja $16\ \mathrm{kHz}$ (ovime postupkom dobivate jednostavan interpolator). Preslušajte tako dobiveni signal. Za filtriranje možete koristiti naredbu `filter`.
   Opisani postupak (ili njegove varijante) se koristi u većini CD uređaja. Signal zapisan na CD-u je otipkan frekvencijom od $44{,}1\ \mathrm{kHz}$. Takav signal se prvo pretipkava s faktorom 4 ili 8 te se na taj način dobiva signal otipkan s $176{,}4\ \mathrm{kHz}$ ili $352{,}8\ \mathrm{kHz}$. Primjenom odgovarajućeg digitalnog filtra se tada interpolirani signal prilagodi tako da dobijemo kvalitetnu reprezentaciju izvornog signala, ali ovog puta s većom frekvencijom otipkavanja. Sada se takav signal prosljeđuje na D/A pretvornik koji radi na odabranoj većoj frekvenciji. Ovim postupkom smo problem visokovjerne reprodukcije prebacili u diskretnu domenu te se cijeli postupak može izvesti na računalu bez upotrebe složenih analognih sklopova.

[25] Vaš filtar mora spektar izmijeniti na takav način da odgovara spektru signala iz drugog zadatka.

11. Dodatak – popis korisnih MATLAB naredbi za pojedine vježbe

U ovom dodatku je dan popis korisnih naredbi za neke vježbe. Detaljnije upute o svakoj naredbi potražite u MATALB-ovom sustavu pomoći (naredba `helpdesk` ili `doc <ime_naredbe>`). Na kraju svakog opisa naredbe unutar MATLAB-ovog sustava pomoći se nalazi popis srodnih naredbi koji će vam svakako biti koristan jer su ovdje navedene samo osnovne naredbe.

Vježba 2. – Upoznavanje s MATLAB-om i Simulink-om

help	kratka pomoć za pojedinu naredbu
helpdesk	pokreće ugrađeni sustav pomoći s indeksom i mogućnošću pretraživanja
doc	detalja pomoć za pojedinu naredbu
lookfor	ispis svih naredbi čija kratka pomoć sadrži zadanu riječ
edit	pokretanje ugrađenog uređivača teksta

Vježba 3. – Konačni automat

ismember	provjera pripadnosti člana skupu
char	kreiranje znakovnog niza
input	postavljanje upita korisniku (tekstualni odgovor)
switch	višestruko grananje toka izvođenja programa
strcmp	usporedba dva tekstualna niza
nargchk	provjera broja ulaznih argumenata
disp	tekstualni ispis rezultata

Vježba 4. – Linearni diskretni sustav drugog reda

tf	definiranje sustava preko prijenosne funkcije (morate zadati period otipkavanja)
bode	crtanje Bodeovog dijagrama za LTI sustave
impulse	određivanje impulsnog odziva
freqz	određivanje frekvencijske karakteristike diskretnog LTI sustava
sos	rastav sustava na kaskadu sustava drugog reda
pzmap	određivanje položaja polova i nula u z-ravnini
zgrid	crtanje linija stalnog faktora prigušenja i frekvencija u z-ravnini
sim	pokretanje simulacije nekog Simulink modela

Vježba 6. – Linearni kontinuirani sustav drugog reda

tf	definiranje sustava preko prijenosne funkcije (ne zadajete period otipkavanja)
freqs	određivanje frekvencijske karakteristike kontinuiranog LTI sustava
impulse	određivanje impulsnog odziva
bode	crtanje Bodeovog dijagrama za LTI sustave
sgrid	crtanje linija stalnog faktora prigušenja i frekvencija u s-ravnini

Vježba 8. – Frekvencijska analiza vremenski kontinuiranih signala

mhelp	MAPLE help (simbolička matematika koristi MAPLE, tako da dostupne naredbe nisu pokrivene običnom help naredbom)
syms	definiranje simboličkih varijabli
int	integriranje
pretty	lijepi ispis simboličkog izraza
simplify	pojednostavljivanje složenih simboličkih izraza
fourier	Fourierova integralna transformacija
ezplot	crtanje simboličkih funkcija
real	realni dio kompleksnog broja
imag	imaginarni dio kompleksnog broja
abs	apsolutna vrijednost kompleksnog broja
angle	kut kompleksnog broja

Vježba 9. – Frekvencijska analiza vremenski diskretnih signala

freqz	računanje frekvencijske karakteristike diskretnih LTI sustava, primijetite da se funkcija može koristiti i za računanje DTFT-a (vremenski diskretna Fourierova transformacija)
fft	brz i efikasan algoritam za računanje DFT-a (diskretna Fourierova transformacija)
specgram	računanje spektrograma
length	duljina vektora
numel	broj elemenata u matrici
wavrecord	snimanje audio zapisa (samo za Windows OS)
wavplay	reprodukcija audio zapisa (samo za Windows OS)
sound	reprodukcija audio zapisa
soundsc	reprodukcija audio zapisa s automatskim podešavanjem glasnoće

Vježba 10. – Uzorkovanje i preklapanje spektra

upsample	pretipkavanje signala
downsample	podtipkavanje signala
interp	interpolacija signala
decimate	decimacija signala
triang	trokutni vremenski otvor (Bartletov otvor)
filter	filtriranje diskretnog signala
butter	dizajn Butterworthovog filtra
linspace	vektor s linearno razmaknutim uzorcima
[a:b:c]	vetkor čiji elementi idu od a do b s korakom c

12. Dodatak – kôd korištenih MATLAB funkcija

U ovom dodatku je dan kod nekih MATLAB funkcija koje se koriste na laboratorijskim vježbama ili su rješenja zadataka za pripremu. Sam kod je dan kao primjer uobičajenog koda MATLAB funkcija ili kao pomoć pri pisanju pripreme.

12.1. Popis funkcija

prijelaz	funkcija prijelaza automata iz pripreme za vježbu 3.
unos	skripta za interakciju s korisnikom iz pripreme za vježbu 3.
spektrogram	računanje i 2D prikaz spektrograma
spektrogram3d	računanje i 3D prikaz spektrograma

12.2. Funkcija prijelaz

Funkcija `prijelaz` je rješenje pripremnog zadatka 1. za vježbu 3. Prije korištenja danog koda svakako najprije sami pokušajte riješiti pripremni zadatak.

```
function [novostanje, izlaz] = prijelaz(starostanje, ulaz)
% PRIJELAZ - Funkcija prijelaza konacnog automata za vjezbu 3.
%   [N, I] = PRIJELAZ(S, U) vraca novo stanje automata N i
%   vrijednost izlaza I. S je staro stanje, a ulaz U je
%   ulazni simbol.

% SIS - Laboratorijske vjezbe - 12. 10. 2004.

% $Revizija: 1.0 $  $Datum: 2004/10/12 $
% $Autor(i): Tomislav Petkovic $

% Oba ulazna argumenta moraju biti zadana.
error(nargchk(2,2,nargin));

% Definiraj skupove ulaznih i izlaznih simbola te stanja.
Ulazi = {'odsutan', '0', '1'};
Izlazi = {'odsutan', '0', '1'};
Stanja = {'A', 'B'};

% Ovisno o ulaznom simbolu te stanju vrati novo
% stanje i izlazni simbol. Koristimo naredbu switch.
% Jednako tako smo mogli korisiti npr. if-then-else
% naredbe.
switch starostanje
case Stanja(1)
   switch ulaz
   case Ulazi(1)
      % Na ulazu smo dobili simbol odsutan te bi automat
      % trebao ostati u nepromijenjenom stanju i vratiti na
      % izlaz simbol odsutan (stuttering reaction).
      novostanje = starostanje;
      izlaz = Izlazi{1};

   case Ulazi(2)
      % Nalazimo se u stanju A i ulaz je 0. Postavljamo
      % novo stanje prema zadanoj funkciji prijelaza te
```

```
37.         % vracamo ispravan izlazni simbol. Jednako radimo
38.         % za sve ostale slucajeve.
39.         novostanje = Stanja{1};
40.         izlaz = Izlazi{3};
41.
42.      case Ulazi(3)
43.         novostanje = Stanja{2};
44.         izlaz = Izlazi{2};
45.
46.      otherwise
47.         % Primili smo ulazni simbol koji nije u skupu definiranih
48.         % simbola. U ovom slucaju bi trebali ispisati poruku o
49.         % gresci, no mozemo i sve nepoznate simbole proglasiti jednakima
50.         % simbolu odsutan.
51.         novostanje = starostanje;
52.         izlaz = Izlazi{1};
53.
54.      end % switch ulaz
55.
56.   case Stanja(2)
57.      switch ulaz
58.      case Ulazi(1)
59.         novostanje = starostanje;
60.         izlaz = Izlazi{1};
61.
62.      case Ulazi(2)
63.         novostanje = Stanja{2};
64.         izlaz = Izlazi{3};
65.
66.      case Ulazi(3)
67.         novostanje = Stanja{1};
68.         izlaz = Izlazi{2};
69.
70.      otherwise
71.         novostanje = starostanje;
72.         izlaz = Izlazi{1};
73.
74.      end % switch ulaz
75.
76.   otherwise
77.      % Zadano je nepostojece stanje za koje funkcija prijelaza
78.      % nije definirana.
79.      error('Zadano je nedozvoljeno stanje.');
80.
81.   end % switch starostanje
```

12.3. Skripta unos.m

Skripta `unos.m` je rješenje pripremnog zadatka 2. za vježbu 3. Prije korištenja danog koda svakako najprije sami pokušajte riješiti pripremni zadatak.

```
1.    % Skripta za laboratorijsku vjezbu 3.
2.
3.    % SIS - Laboratorijske vjezbe - 12. 10. 2004.
4.
5.    % $Revizija: 1.0 $  $Datum: 2004/10/12 $
6.    % $Autor(i): Tomislav Petkovic $
7.
```

```
8.    % Definiraj string u koji spremamo rezultat i tekst pitanja.
9.    ulaz = 'start';
10.   pitanje = 'Upisi ulazni simbol (dosta za kraj) >';
11.
12.   % Definiraj skup dozvoljenih ulaznih simbola i stanja.
13.   Ulazi = {'odsutan', '0', '1'};
14.   Stanja = {'A', 'B'};
15.
16.   % Pocetno stanje je A.
17.   stanje = Stanja{1};
18.   disp(['Pocetno stanje je ' stanje '.']);
19.
20.   % Ponavljaj sve dok korisnik ne unese dosta.
21.   while ~strcmp(ulaz, 'dosta')
22.      ulaz = input(pitanje, 's');
23.
24.      % Ispitaj da li je unos ispravan ulaz za nas automat.
25.      % Ako nije proslijedi funkciji prijelaz simbol odsutan.
26.      Simbol = {ulaz};
27.      switch Simbol{1}
28.      case Ulazi
29.         % Za varijable tipa cell array naredba switch izvrsava
30.         % case dio ako ulazni element odgovara bilo kojem elementu
31.         % iz skupa Ulazi.
32.         [stanje, izlaz] = prijelaz(stanje, ulaz);
33.      otherwise
34.         % Funkciji prijelaza proslijedujemo simbol odsutan.
35.         [stanje, izlaz] = prijelaz(stanje, Ulazi{1});
36.      end % switch {unos}
37.      disp(['Novo stanje je ' stanje '. Izlazni simbol je ' izlaz '.']);
38.
39.   end
```

12.4. Funkcija spektrogram

Funkcija `spektrogram` računa i crta spektrogram signala. Koristite li MATLAB R13 možete upotrebljavati i ugrađenu funkciju `specgram`.

```
1.    function [Si, wi, ti] = spektrogram(x, fs)
2.    % SPEKTROGRAM - Racunanje i 2D prikaz spektorgrama signala.
3.    %   SPEKTROGRAM(X) crta spektrogram zadanog signala X.
4.    %   Frekvencija je normalizirana od 0 do 1, gdje jedinica
5.    %   odgovara polovini frekvencije otipkavanja.
6.    %
7.    %   SPEKTROGRAM(X, Fs) crta spektrogram zadanog signala X
8.    %   otipkanog s frekvencijom Fs.
9.    %
10.   %   [S, w, t] = SPEKTROGRAM(X, Fs) vraca izracunati spektogram
11.   %   u matrici S. Vektori w i t sadrze vremenske trenutke i frekvenciju.
12.
13.   % SIS - Laboratorijske vjezbe - 20. 10. 2004.
14.
15.   % $Revizija: 1.0 $  $Datum: 2004/10/20 $
16.   % $Autor(i): Tomislav Petkovic $
17.
18.   % Signal mora biti zadan.
19.   error(nargchk(1,2,nargin));
20.
```

```
% Ako frekvencija nije zadana uzmi normaliziranu frekvenciju.
if 2 ~= nargin
    fs = 2;
end

% Odredi duljinu signala.
N = length(x);

% Odredi korak analize i duljinu vremenskog otvora. Pri tome
% je doboro uzeti duljinu vremenskog otvora koja je potencija broja
% dva jer time iskoristavamo sve mogucnosti FFT algoritma.
% Neka otvor bude fiksan duljine 128 uzoraka (128 je 2^7). DFT tada
% takoder racunamo u 128 tocaka.
Nw = 128;

% Prilikom pomicanja otvora preko signala nema smisla racunati DFT
% za svaki pomak jer time znacajno usporavamo postupak bez dobivanja
% novih informacija. Korak pomicanja otvora stoga uzimamo jednak
% polovini trajanja otvora.
korak = Nw / 2;

% Sada za svaki polozaj otvora moramo odrediti DFT. No da bi mogli
% spremiti sve izracunate spektre najprije moramo inicijalizirati
% matricu S. Neka jedan stupac matrice S odgovara jednom spektru.
SNx = Nw;
SNy = floor(abs(N - korak) / korak);
S = zeros(SNx, SNy);

% Sada racunamo spektrogram korak po korak. Zapocinjemo od prvog
% okvira analize kada se pocetak otvora i signala podudaraju.
% Tada pomicemo okvir sve dok ne stignemo do predzadnjeg polozaja
% vremenskog otvora. Za zadnji polozaj vremenskog otvora moramo
% naknadno odrediti spektar jer se moze dogoditi vremenski otvor
% zahvaca dio izvan snimljenog signala.
for i = 0 : SNy - 1

    % Izdvajamo dio signala odreden vremenskim otvorom. Pri tome
    % mnozenje signala s otvorom nije potrebno jer mozemo jednostavno
    % izdvojiti dio signala koji nas zanima.
    xw = x(i*korak+1 : i*korak+Nw);

    % Racunamo DFT izdvojenog signala. Kako izdvojeni signal mora biti
    % vektor-stupac pretvaramo ga u stupac koristeci operator :.
    S(:, i+1) = fft(xw(:), Nw);

end

% Za zadnji okvir analize dio signala uzimamo do zadnjeg uzorka koji
% postoji, dok sve nepostojece uzorke proglasavamo jednakim nuli.
if isempty(i)
    i = 0;
end
xw = x(i*korak+1 : N);
S(:, i+1) = fft(xw(:), Nw);

% Ovime smo odredili cijeli DFT spektar, no zanima nas samo dio od 0 do
% polovine frekvencije otpikavanja.
S = S(1 : Nw/2, :);
```

```
79.
80.   % Sada moramo odrediti vektore t i w koji definiraju osi.
81.   % Vremenski trenutci neka odgovaraju polozaju centra vremenskog otvora.
82.   % Frekvencijska skala je odredena brojem uzoraka DFT-a i frekvencijom
83.   % otipkvanja.
84.   t = [1 : SNy] * korak / fs;
85.   w = [1 : Nw/2] * fs / Nw;
86.
87.   % Ako korisnik sprema rezultat vrati ga, u protivnom nacrtaj spektar.
88.   % Crtamo spektar u obicnoj skali kao sliku.
89.   if 0 == nargout
90.
91.       % Brisemo postojecu sliku.
92.       h = gcf;
93.       clf;
94.
95.       % Crtamo spektrogram. Funkcija imagesc koristi koordinatni sustav s
96.       % ishodistem u gornjem lijevom kutu pa naknadno moramo okrenuti osi.
97.       imagesc(t, w, abs(S));
98.       set(gca,'YDir','normal');
99.
100.      % Postavi paletu i oznaci graf.
101.      colormap(jet);
102.      ylabel('frekvencija');
103.      xlabel('vrijeme');
104.
105.  else
106.      Si = S;
107.      wi = w;
108.      ti = t;
109.
110.  end
```

12.5. Funkcija spektrogram3d

Funkcija `spektrogram3d` je gotovo identična funkciji `spektrogram` samo što prikazuje spektrogram u 3D.

```
1.    function [Si, wi, ti] = spektrogram3d(x, fs)
2.    % SPEKTROGRAM3D - Racunanje i 3D prikaz spektorgrama signala.
3.    %   SPEKTROGRAM3D(X) crta spektrogram zadanog signala X.
4.    %   Frekvencija je normalizirana od 0 do 1, gdje jedinica
5.    %   odgovara polovini frekvencije otipkavanja.
6.    %
7.    %   SPEKTROGRAM3D(X, Fs) crta spektrogram zadanog signala X
8.    %   otipkanog s frekvencijom Fs.
9.    %
10.   %   [S, w, t] = SPEKTROGRAM3D(X, Fs) vraca izracunati spektogram
11.   %   u matrici S. Vektori w i t sadrze vremenske trenutke i frekvenciju.
12.
13.   % SIS - Laboratorijske vjezbe - 20. 10. 2004.
14.
15.   % $Revizija: 1.0 $  $Datum: 2004/10/20 $
16.   % $Autor(i): Tomislav Petkovic $
17.
18.   % Signal mora biti zadan.
19.   error(nargchk(1,2,nargin));
20.
```

```
% Ako frekvencija nije zadana uzmi normaliziranu frekvenciju.
if 2 ~= nargin
    fs = 2;
end

% Odredi duljinu signala.
N = length(x);

% Odredi korak analize i duljinu vremenskog otvora. Pri tome
% je doboro uzeti duljinu vremenskog otvora koja je potencija broja
% dva jer time iskoristavamo sve mogucnosti FFT algoritma.
% Neka otvor bude fiksan duljine 128 uzoraka (128 je 2^7). DFT tada
% takoder racunamo u 128 tocaka.
Nw = 128;

% Prilikom pomicanja otvora preko signala nema smisla racunati DFT
% za svaki pomak jer time znacajno usporavamo postupak bez dobivanja
% novih informacija. Korak pomicanja otvora stoga uzimamo jednak
% polovini trajanja otvora.
korak = Nw / 2;

% Sada za svaki polozaj otvora moramo odrediti DFT. No da bi mogli
% spremiti sve izracunate spektre najprije moramo inicijalizirati
% matricu S. Neka jedan stupac matrice S odgovara jednom spektru.
SNx = Nw;
SNy = floor(abs(N - korak) / korak);
S = zeros(SNx, SNy);

% Sada racunamo spektrogram korak po korak. Zapocinjemo od prvog
% okvira analize kada se pocetak otvora i signala podudaraju.
% Tada pomicemo okvir sve dok ne stignemo do predzadnjeg polozaja
% vremenskog otvora. Za zadnji polozaj vremenskog otvora moramo
% naknadno odrediti spektar jer se moze dogoditi vremenski otvor
% zahvaca dio izvan snimljenog signala.
for i = 0 : SNy - 1

    % Izdvajamo dio signala odreden vremenskim otvorom. Pri tome
    % mnozenje signala s otvorom nije potrebno jer mozemo jednostavno
    % izdvojiti dio signala koji nas zanima.
    xw = x(i*korak+1 : i*korak+Nw);

    % Racunamo DFT izdvojenog signala. Kako izdvojeni signal mora biti
    % vektor-stupac pretvaramo ga u stupac koristeci operator :.
    S(:, i+1) = fft(xw(:), Nw);

end

% Za zadnji okvir analize dio signala uzimamo do zadnjeg uzorka koji
% postoji, dok sve nepostojece uzorke proglasavamo jednakim nuli.
if isempty(i)
    i = 0;
end
xw = x(i*korak+1 : N);
S(:, i+1) = fft(xw(:), Nw);

% Ovime smo odredili cijeli DFT spektar, no zanima nas samo dio od 0 do
% polovine frekvencije otpikavanja.
S = S(1 : Nw/2, :);
```

```

% Sada moramo odrediti vektore t i w koji definiraju osi.
% Vremenski trenutci neka odgovaraju polozaju centra vremenskog otvora.
% Frekvencijska skala je odredena brojem uzoraka DFT-a i frekvencijom
% otipkvanja.
t = [1 : SNy] * korak / fs;
w = [1 : Nw/2] * fs / Nw;

% Ako korisnik sprema rezultat vrati ga, u protivnom nacrtaj spektar.
% Crtamo 3D prikaz spektrograma.
if 0 == nargout

    % Brisemo postojecu sliku.
    h = gcf;
    clf;

    % Crtamo spektrogram.
    waterfall(t, w, abs(S));

    % Postavi paletu i oznaci graf.
    colormap(jet);
    ylabel('frekvencija');
    xlabel('vrijeme');
    zlabel('amplituda');

else
    Si = S;
    wi = w;
    ti = t;

end
```

Literatura

1. K. Aleksić-Maslać, H. Babić, B. Jeren, *Analiza i simulacija sustava digitalnim računalom, Priručnik za laboratorijske vježbe iz predmeta Signali i sustavi (zavodska skripta)*, ZESOI-FER, Zagreb, 1997., http://sis.zesoi.fer.hr/laboratorij/doc/sis_2001_labupute.doc
2. Edward A. Lee, Pravin Varaiya, *Structure and Interpretation of Signals and Systems*, Addison Wesley, 2003.
3. H. Babić, *Signali i sustavi (zavodska skripta)*, ZESOI-FER, Zagreb 1996., http://sis.zesoi.fer.hr/predavanja/pdf/sis_2001_skripta.pdf
4. Sanjit K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, McGraw-Hill, 1998.
5. Nino Boccara, *Modeling Complex Systems*, Springer-Verlag, New York 2004.
6. *Getting Started with MATLAB version 6*, The MathWorks, srpanj 2002.
7. *Using Simulink Version 5,* The MathWorks, srpanj 2002.
8. *Getting Started with MATLAB version 5*, 2. izdanje, The MathWorks, svibanj 1997.
9. *Simulink Users Guide Version 2.1*, The MathWorks, svibanj 1997.
10. Adrian Biran, Moshe Breiner, *MATLAB 5 for Engineers*, 2. izdanje, Addison-Wesley, 1999.