Osnove crtanja signala

Nedeljko Stojaković
Marko Pejić

Jun, 2021.

Cilj ovog materijala je upoznavanje sa vizuelizacijom podataka u programskom jeziku *Julia*. Na ovom kursu, za potrebe vizuelizacije podataka koristićemo paket *Plots*, sa kojim ćemo se detaljnije upoznati kroz primere koji slede. Programski paket *Julia* sadrži više različitih paketa za vizuelizaciju, ali mi ćemo se fokusirati na paket *Plots*, zbog njegove fleksibilnosti i jednostavnosti.

1. Uvod

Vizuelizacija podataka ima bogatu i kompleksnu istoriju. Softver za iscrtavanje grafika čini kompromise između naprednih opcija i jednostavnosti, brzine izvršavanja i lepote grafika, te statičkog i dinamičkog interfejsa. Neki paketi kreiraju grafik i nikada ga ne menjaju, dok drugi ažuriraju prikaz u realnom vremenu.

Plots je skup alata i interfejs za vizuelizaciju podataka u programskom paketu *Julia*. Nalazi se iznad ostalih *backend*-a, kao što su *GR*, *PyPlot*, *PGFPlotsX* ili *Plotly*, povezujući komande sa njihovom implementacijom. Ukoliko jedan *backend* ne podržava željene karakteristike, jednostavno je moguće prebaciti se na drugi *backend*, upotrebom jedne komande, bez potrebe za izmenom koda ili učenja nove sintakse.

Da bi mogli da koristimo paket *Plots*, neophodno je prvo ga instalirati [1], a potom i uključiti u okviru skripte u kojoj kucamo naredbe.

[1] Ukoliko je na korisničkom nalogu već dodat paket *Plots*, na korisniku ostaje samo korišćenje tog paketa, odnosno naredba *using Plots*.

```
using Pkg
Pkg.add("Plots")

using Plots
```

2. Potrebno predznanje

Budući da neke signale možemo predstaviti pomoću više različitih pravih, neophodno je znati definisati jednačinu prave kroz dve zadate tačke (1).

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{1}$$

S obzirom da ćemo u nastavku ovog materijala raditi i sa periodičnim signalima

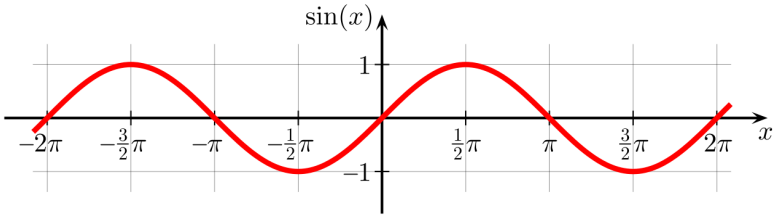
(sekcija 4), neophodno je podsetiti se osnovnih pojmova vezanih za ovakve signale. Signal je periodičan ako nastaje beskonačnim ponavljanjem neke sekvence konačnog trajanja. Trajanje sekvence koja se ponavlja naziva se perioda signala i označava se sa T. Učestanost ili frekvencija je broj ponavljanja u jedinici vremena, označava se sa f, a osnovna jedinica mere je Hz (herc). Frekvencija predstavlja recipročnu vrednost periode (2), odnosno 1 Hz ekvivalentan je jednom ponavljanju u sekundi.

$$f = \frac{1}{T} \tag{2}$$

Veza između kružne učestanosti i učestanosti (ili periodom) data je sledećom jednačinom (3). Jedinica mere kružne učestanosti je $\frac{rad}{s}$.

$$\omega = 2\pi f = \frac{2\pi}{T} \tag{3}$$

Periodične funkcije sa kojima ste se do sada verovatno najviše susretali su sinus (sin) i kosinus (cos). Sa slike 1 može se primetiti da je perioda funkcije $y = sin(x)$ jednaka 2π.



Slika 1: Grafik funkcije $y = sin(x)$

Međutim, kako definisati sinusnu funkciju sa periodom T različitom od 2π? Jednačina (4) predstavlja izraz na osnovu kog možemo definisati sinusnu funkciju proizvoljne periode.

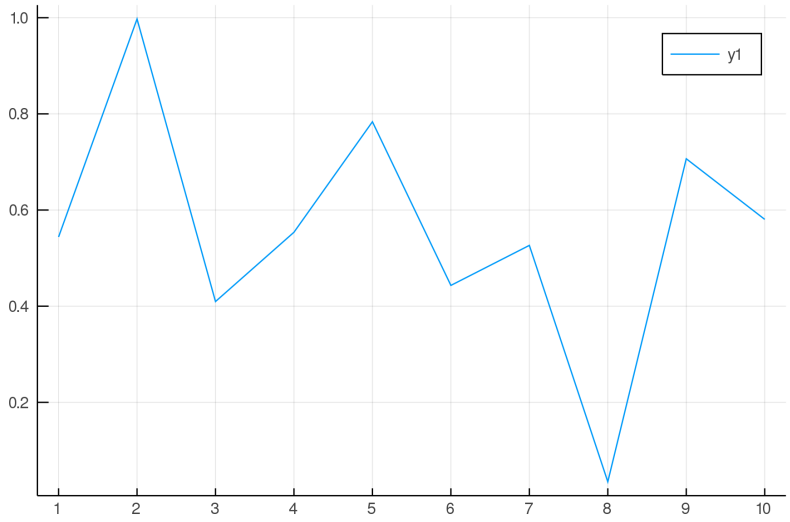
$$y = sin(\omega x) = sin(\frac{2\pi}{T}x) \tag{4}$$

3. Linijski plot

U ovom poglavlju dati su primeri jednostavnih linijskih plotova, uz prikaz različitih opcija uređivanja grafika.

Funkcija koja se koristi za iscrtavanje grafika u paketu *Plots* je *plot*. Postoji puno različitih načina da se pozove ova funkcija, ali za nas najznačajniji je prosleđivanje dva vektora iste dužine. Funkcija *plot* zatim iscrtava grafik tako što pravim linijama povezuje tačke definisane korespodentnim elementima ulaznih vektora. Sledeći primer prikazuje jednostavan linijski plot, definisan pomoću vektora t i y. Vektor t sadrži koordinate tačaka na x-osi, a vektor y koordinate tačaka na y-osi. Za generisanje vektora y iskorišćena je funkcija *rand*, čime je dobijen vektor slučajnih vrednosti od deset elemenata. Parametar *xticks* definiše podeoke na x-osi.

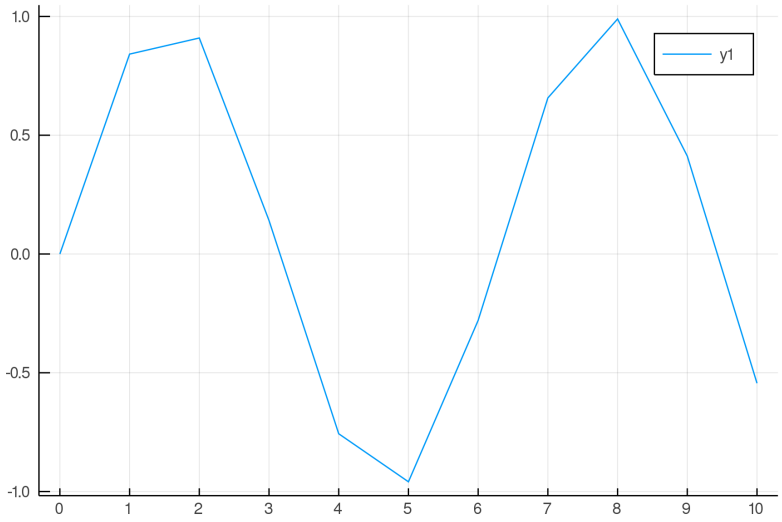
```
t = 1:10;
y = rand(10);

plot(t, y, xticks=1:10)
```



Slika 2: Linijski plot - primer 1

U sledećem primeru linijski plot definisan je takođe pomoću dva vektora. Vektor y definisan je tako što je izračunata vrednost sinusa za svaku od tačaka vektora t, pozivom funkcije $sin.(t)$. Ukoliko pažljivo pogledamo dobijeni grafik (slika 3), primetićemo da dobijena sinusoida ne izgleda dovoljno glatko. Zašto?

```
t = 0:10;
y = sin.(t);

plot(t, y, xticks=0:10)
```

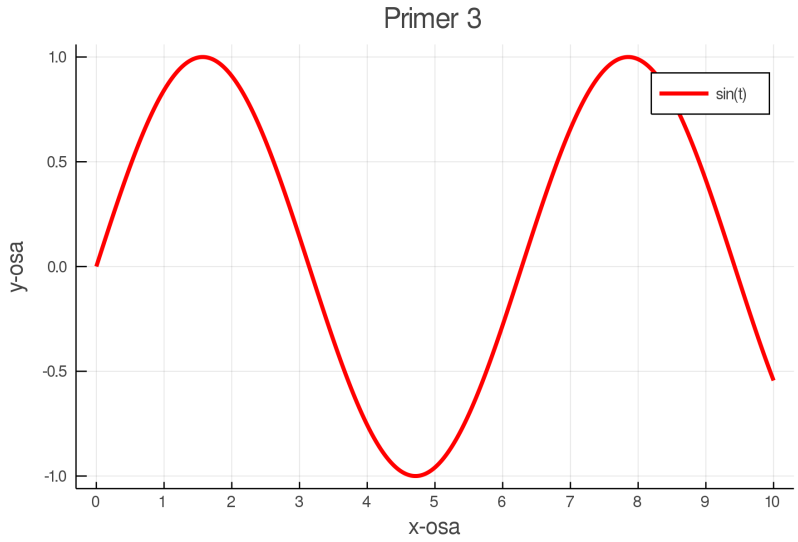


Slika 3: Linijski plot - primer 2

Odgovor na pitanje postavljeno u prethodnom primeru krije se u činjenici da funkcija *plot* iscrtava grafik tako što povezuje prosleđene tačke pravim linijama. Prema tome, ukoliko želimo da iscrtamo funkciju koja nije definisana isključivo preko pravih, neophodno je da koristimo više tačaka. U sledećem primeru, vektor t je definisan pomoću opsega *0:0.01:10*, što znači da će rezultujući vektor sadržati 1001 vrednost. Samim tim, i vektor y će sadržati isti broj vrednosti, što će rezultovati iscrtavanjem funkcije $y(t) = sin(t)$ kroz znatno veći broj tačaka, odnosno funkcija će izgledati glatko. U ovom primeru je dodatno prikazano i na koji način je moguće izvršiti uređivanje grafika, tačnije postavljanje naslova, legende, boje, debljine linije, teksta na osama i slično.[1]

```
t = 0:0.01:10;
y = sin.(t);

plot(t, y, title="Primer 3", label="sin(t)", lw=3, color=:red, xticks=0:10)
xlabel!("x-osa")
ylabel!("y-osa")
```

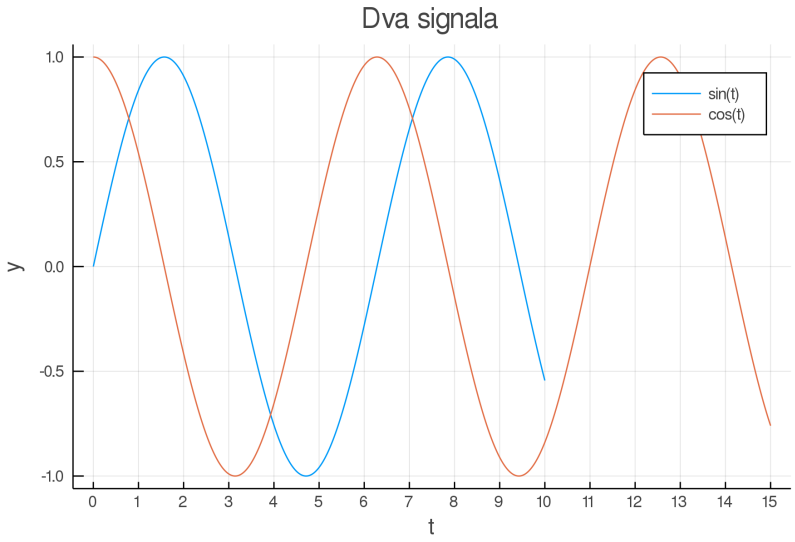
[1] Funkcija *plot* je funkcija sa imenovanim parametrima, gde su imenovani parametri u stvari atributi za podešavanje grafika. Za dodatne informacije o atributima pogledati dokumentaciju za paket *Plots*.



Slika 4: Uređivanje grafika - primer 3

Sledeći primer prikazuje na koji način je moguće prikazati više od jednog signala na istom grafiku. Za potrebe dodavanja signala na već postojeći grafik koristi se funkcija *plot!*, koja prima identične parametre kao i funkcija *plot*. Razlika između ove dve funkcije je ta što *plot!* iscrtava signal na već otvorenom grafiku, dok *plot* kreira novi grafik i iscrtava signal na njemu. Još jedna bitna stvar koju treba primetiti u ovom primeru jesu različite dužine vektora t_1 i t_2. U ovakvim situacijama, *Julia* skalira ose generisanog grafika prema najdužem vektoru. Prema tome, vrednosti na x-osi idu do vrednosti 15, s tim da je signal iscrtan plavom bojom definisan na intervalu $[0, 10]$.

```
t1 = 0:0.01:10;
t2 = 0:0.01:15;

y1 = sin.(t1);
y2 = cos.(t2);

plot(t1, y1, title="Dva signala", label="sin(t)", xticks=0:10)
plot!(t2, y2, label="cos(t)", xticks=0:15)
xlabel!("t")
ylabel!("y")
```



Slika 5: Dva signala na istom grafiku - primer 4

Vrlo se česte situacije u kojima je signal opisan pomoću više pravih, definisanih na različitim intervalima. Tada je potrebno naći jednačinu svake pojedinačne prave definisane sa dve tačke (videti jednačinu 1), a pomoću logičkih uslova ograničiti intervale u kojima su te prave definisane. Ukoliko pogledamo sliku 6, možemo primetiti da je signal definisan pomoću dve prave, definisane u različitim intervalima. Prva prava prolazi kroz tačke $(0, 0)$ i $(1, 5)$, a druga kroz tačke $(1, 5)$ i $(2, 0)$. Ukoliko koristimo zapis *[interval na kom se prava prostire] - [jednačina prave]*, dve prave koje čine signal su:

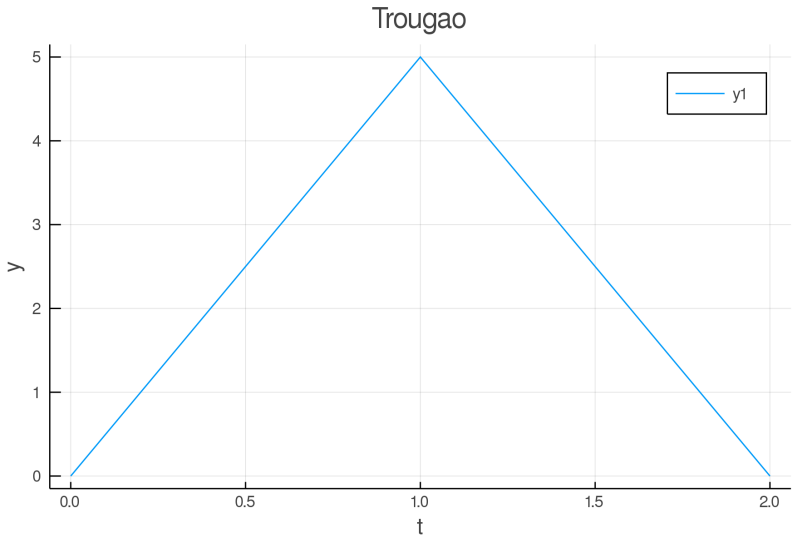
- $[0, 1) \rightarrow y = 5x$
- $[1, 2] \rightarrow y = -5x + 10$

Ovakav zapis znatno olakšava pisanje koda, budući da signal u kodu predstavljamo u formatu:
(jedn. prave 1) . (opseg važenja 1) .+ (jedn. prave 2) .* (opseg važenja 2) .+ ... .+ (jedn. prave n) .* (opseg važenja n).*

Sledeći primer ilustruje kod kojim je generisan signal prikazan na slici 6 [2]. Bitna napomena je da vektor t sadrži vrednosti na x-osi, što znači da je u kodu potrebno jednačine prave izraziti kao funkcije od t, a ne od x (pošto x ne postoji kao promenljiva, ne možemo napisati $5 * x$).

```
t = 0:0.1:2;
y = 5 * t .* (t .< 1) + (-5 * t .+ 10) .* ((t .>= 1) .& (t .<= 2));

plot(t, y, title="Trougao")
xlabel!("t")
ylabel!("y")
```

[2] Da biste razumeli ovakav zapis signala, potrebno je da razmislite šta je rezultat izvršavanja naredbe u formatu (jednačina prave) .* (opseg važenja). Npr. u sledećem kodu, šta je rezultal izvršavanja naredbe: 5 * t .* (t .< 1)?



Slika 6: Signal definisan pomoću pravih i logičkih uslova - primer 5

4. Periodični signali

U slučaju periodičnih signala možemo da primenimo istu logiku kao u primerima iz prethodnog poglavlja. Za signale sa više perioda ovaj pristup postaje nepraktičan, jer se kod umnožava što povećava šansu za grešku, a ujedno takvo rešenje je nepregledno.

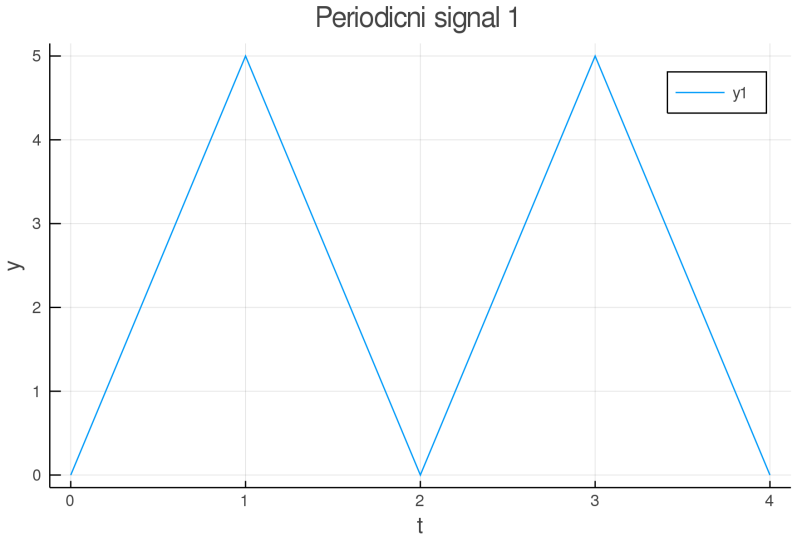
Na primer, ukoliko želimo da proširimo prethodni primer tako da dobijemo periodični signal, za 2 periode signala imamo 4 jednačine prave, odnosno za 10 perioda to je 20 jednačina pravih. Sledeći primer ilustruje proširenje prethodnog primera, bez potrebe za računanjem dodatnih jednačina pravih. Ako uporedimo kod za ova dva primera, primetićemo da je razlika veoma mala. Prva razlika se ogleda u tome što sada imamo signal koji traje $4s$, tako da je vektor t izmenjen. Druga, bitnija razlika je uvođenje nove promenljive koju najčešće označavamo sa *tp* (kao *t-periodično*) koju definišemo pomoću ugrađene funkcije *rem*.[1] Promenljiva *tp* se definiše tako što čitav vremenski interval trajanja signala t delimo sa brojem koji predstavlja dužinu jedne periode[2] i posmatramo ostatak pri tom deljenju. Nakon toga, prva perioda signala se opisuje na isti način kao u primeru 5 (Slika 6), ali korišćenjem nove promenljive *tp*. Treba obratiti pažnju da se funkcija *plot* poziva sa vektorom t, kako bi iscrtali sve periode signala.

Na ovaj način obezbedili smo periodičnost signala, bez računanja dodatnih jednačina pravih. Da bismo bolje razumeli kako ovo radi, najjednostavnije je da ručno prođemo kroz deo računa koji obavlja kod. To ćemo uraditi tako što ćemo sa intervala t odabrati nekoliko tačaka i izračunati vrednost promenljive *tp*.

[1] Funkcija *rem(X, Y)* vraća ostatak pri deljenju X sa Y. Funkcija je dobila ime od engleske reči za ostatak (*remainder*).

[2] U ovom primeru to je broj 2.

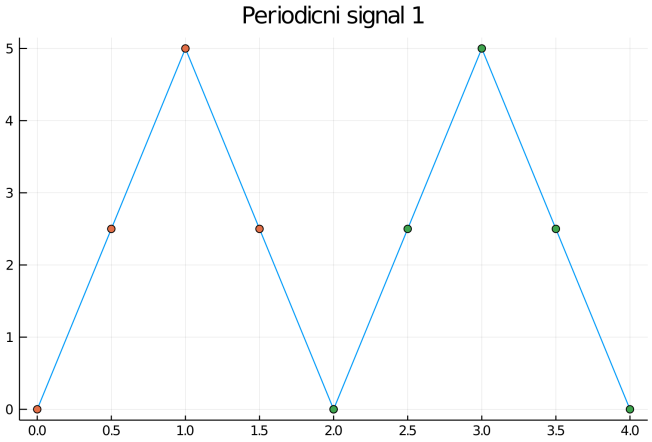
```
t = 0:0.1:4;
tp = rem.(t, 2);
y = 5 * tp .* (tp .< 1) + (-5 * tp .+ 10) .* ((tp .>= 1) .& (tp .<= 2));

plot(t, y, title="Periodicni signal 1")
xlabel!("t")
ylabel!("y")
```



Slika 7: Periodični signal - Primer 1.

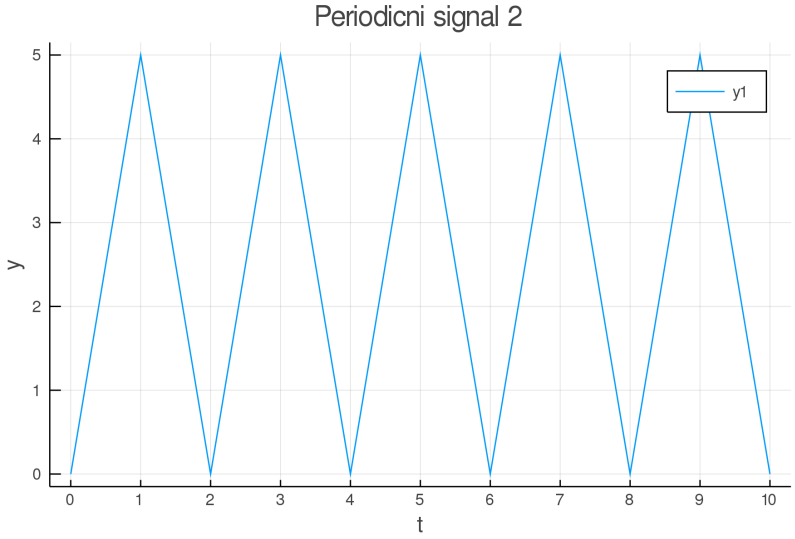
Odabraćemo nekoliko vrednosti sa prve i druge periode signala iz prethodnog primera. Za svaku tačku izračunata je vrednost *tp*, a dobijeni rezultat predstavljen je u tabeli 1. Na slici 8 markerima su prikazane vrednosti iz tabele. Iz ovog jednostavnog računa, na osnovu svega nekoliko tačaka, možemo zaključiti da će se vrednosti u vektoru *tp* ponavljati periodično u zavisnosti od dužine periode signala. Na taj način vršimo translaciju prve periode signala u vremenu što rezultuje periodičnim signalom. Promenom dužine vektora *t* menjamo dužinu periodičnog signala, kao što možemo da vidimo na slici 9.

```
t = 0:0.1:10;
tp = rem.(t, 2);
y = 5 * tp .* (tp .< 1) + (-5 * tp .+ 10) .* ((tp .>= 1) .& (tp .<= 2));

plot(t, y, title="Periodicni signal 2", xticks=0:10)
xlabel!("t")
ylabel!("y")
```



t	$rem(t,2)$
0	0
0.5	0.5
1	1
1.5	1.5
2	0
2.5	0.5
3	1
3.5	1.5
4	0

Slika 8 i Tabela 1: Vrednosti tp za različite tačke.



Slika 9: Periodični signal - Primer 2.

5. Složeni signali

Složeni signali podrazumevaju signale nastale kao kombinacija više različitih jednostavnijih signala. Veoma čest način formiranja složenih signala jeste prepoznavanjem svih jednostavnijih komponenti signala, a potom se vrši odsecanje. Sledeći primer ilustruje formiranje složenog signala, prikazanog na slici 10 i označenog punom linijom. Može se primetiti da je signal sastavljen iz dve jednostavnije komponente:

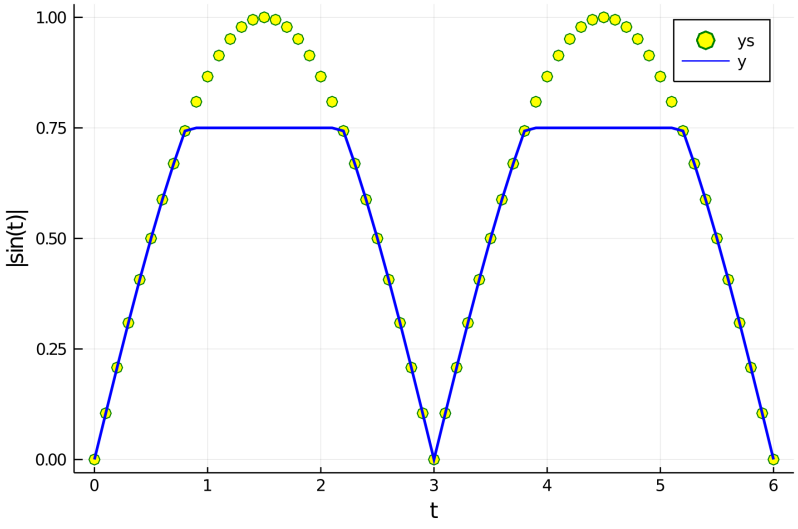
- $y_1 = |sin(\frac{\pi}{3}t)|$
- $y_2 = 0.75$

Odsecanje signala vrši se tako što za definisane dve funkcije, u svakoj tački biramo manju ili veću vrednost, zavisno od izgleda traženog složenog signala. Budući da je vrednost funkcije y_1 potrebno ograničiti sa gornje strane vrednošću funkcije y_2, neophodno je za svaku tačku izabrati manju od dveju vrednosti funkcija, odnosno iskoristiti funkciju *min*.[1] Naredni kod ilustruje način na koji je generisan signal predstavljen na slici 10. Funkcija *scatter* iskorišćena je za iscrtavanje žutih tačaka, kako bi komponente složenog signala bile jednostavnije uočljive.

[1] Primer upotrebe funkcije *min* nad vektorom v:

$$[1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9]$$
$$\downarrow\ min.(v,\ 5)$$
$$[1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 5\ \ 5\ \ 5\ \ 5]$$

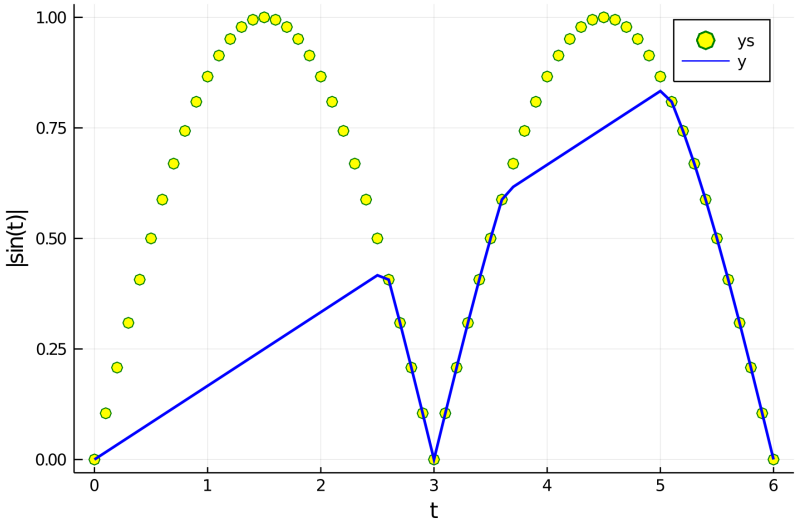
```
t = 0:0.1:6;
ys = abs.(sin.(pi/3 * t));
y = min.(ys, 0.75);

scatter(t, ys, markershape=:o, markerstrokecolor=:green, color=:yellow)
plot!(t, y, lw=2, xlabel="t", ylabel="|sin(t)|", color=:blue)
```



Slika 10: Složeni signal - Primer 1.

Sledeći primer prikazuje veoma sličan složen signal, sa nešto drugačijom jednačinom prave. Jedna komponenta signala je i dalje funkcija $y_s = |sin(\frac{\pi}{3}t)|$, dok je druga komponenta jednačina prave definisana funkcijom $p = \frac{1}{6}t$.

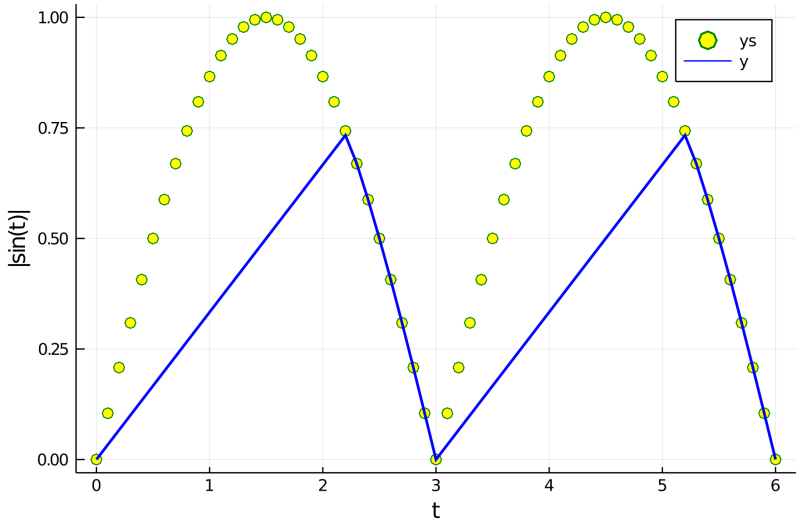
```
t = 0:0.1:6;
ys = abs.(sin.(pi/3 * t));
p = 1/6 * t;
y = min.(ys, p);

scatter(t, ys, markershape=:o, markerstrokecolor=:green, color=:yellow, label = "ys")
plot!(t, y, lw=2, xlabel="t", ylabel="|sin(t)|", color=:blue, label = "y")
```



Slika 11: Složeni signal - Primer 2.

Jedna komponenta složenog signala iz sledećeg primera ista je kao i u prethodnim primerima, odnosno definisana je funkcijom $y_s = |sin(\frac{\pi}{3}t)|$. Druga komponenta može se predstaviti kao periodičan signal, čija je perioda jednaka 3, a jednačina prave definisana je funkcijom $p = \frac{1}{3}t_p$.

```
t = 0:0.1:6;
ys = abs.(sin.(pi/3 * t));
tp = rem.(t, 3);
p = 1/3 * tp;
y = min.(ys, p);

scatter(t, ys, markershape=:o, markerstrokecolor=:green, color=:yellow, label = "ys")
plot!(t, y, lw=2, xlabel="t", ylabel="|sin(t)|", color=:blue, label = "y")
```

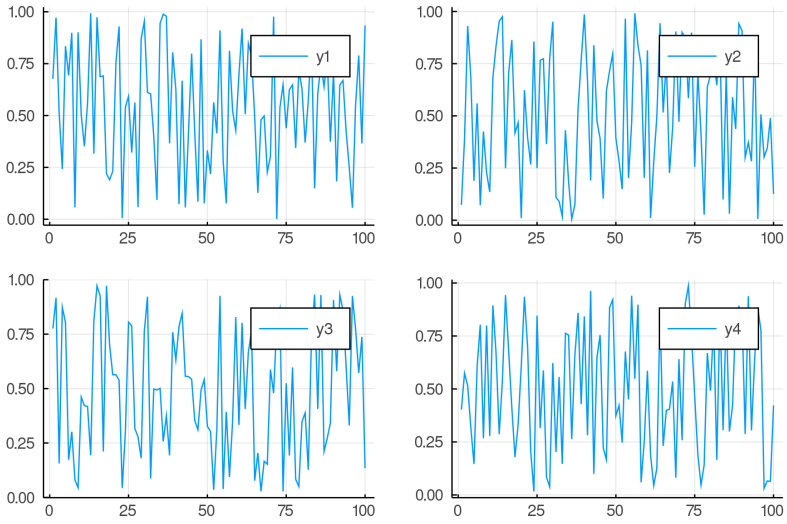


Slika 12: Složeni signal - Primer 3.

6. Raspored više grafika

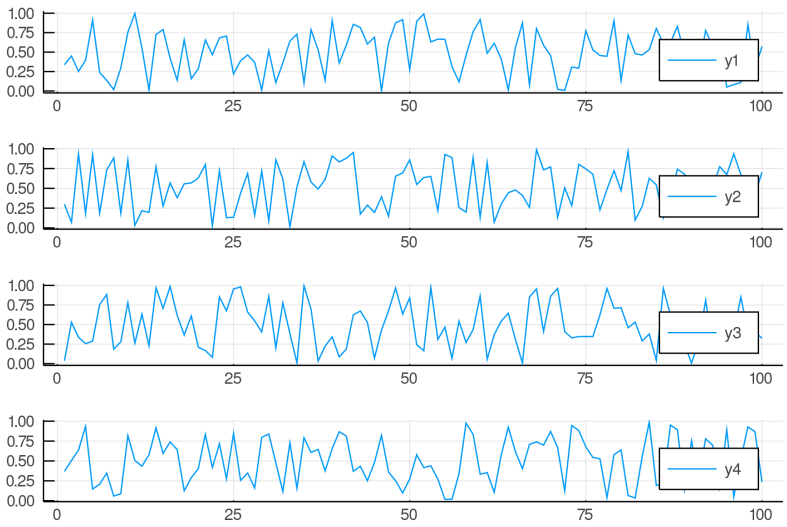
Paket *Plots* omogućuje i raspoređivanje više različitih grafika u okviru iste figure. Ovo je moguće uraditi na više načina, za koje je zajedničko definisanje rasporeda upotrebom *layout* parametra *plot* funkcije. Naredni primeri ilustruju kreiranje različitih rasporeda više grafika, dobijenih generisanjem matrice slučajnih vrednosti, dimenzija 100x4. Matrica ima četiri kolone, odnosno svaki vektor-kolona prestavlja signal koji se iscrtava u okviru jednog od grafika.

```
plot(rand(100, 4), layout=4)
```



Slika 13: Raspored grafika 2x2 - Primer 1.

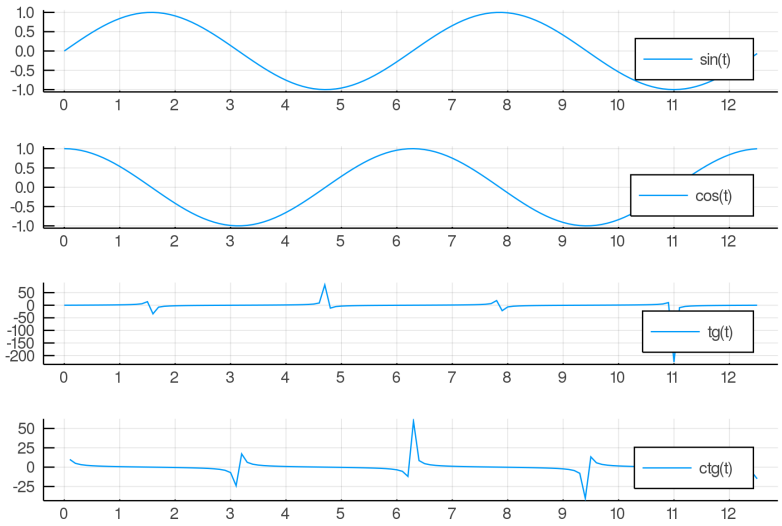
```
plot(rand(100, 4), layout=(4, 1))
```



Slika 14: Raspored grafika 4x1 - Primer 2.

Sledeći primer ilustruje nešto drugačije kreiranje rasporeda više grafika: zasebnim pozivima *plot* funkcije kreirana su četiri odvojena grafika, koji su potom prikazani u okviru jedne figure novim pozivom funkcije *plot*, uz definisan raspored kroz *layout* parametar.

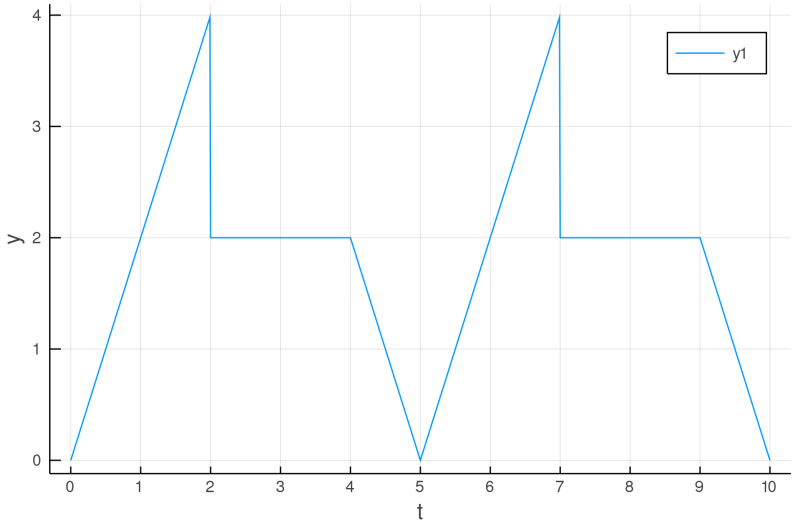
```
t = 0:0.1:4*pi;

y1 = sin.(t);
y2 = cos.(t);
y3 = tan.(t);
y4 = cot.(t);

p1 = plot(t, y1, label="sin(t)", xticks=0:4*pi);
p2 = plot(t, y2, label="cos(t)", xticks=0:4*pi);
p3 = plot(t, y3, label="tg(t)", xticks=0:4*pi);
p4 = plot(t, y4, label="ctg(t)", xticks=0:4*pi);

plot(p1, p2, p3, p4, layout=(4, 1))
```



Slika 15: Raspored grafika 4x1 - Primer 3.

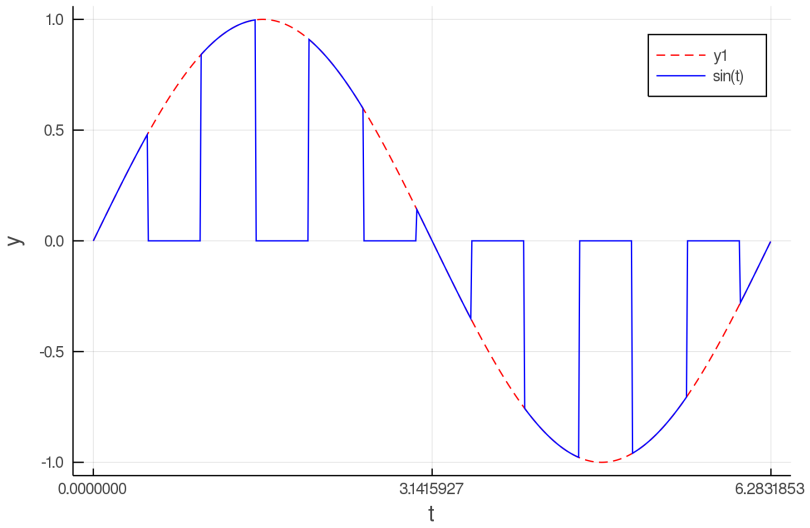
7. Zadaci za vežbu

Zadatak 1. Napisati kod koji generiše periodični signal prikazan na slici ispod.



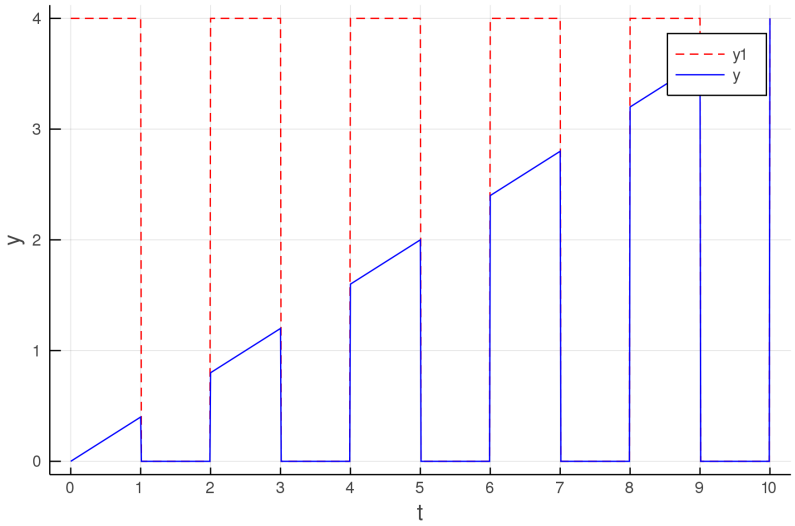
Slika 16: Zadatak 1.

Zadatak 2. Napisati kod koji generiše periodični signal prikazan na slici ispod.



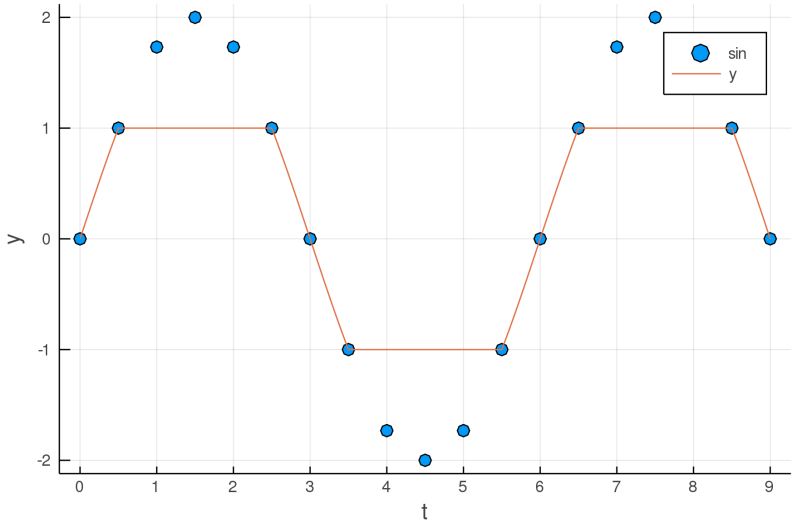
Slika 17: Zadatak 2.

Zadatak 3. Napisati kod koji generiše periodični signal prikazan na slici ispod.



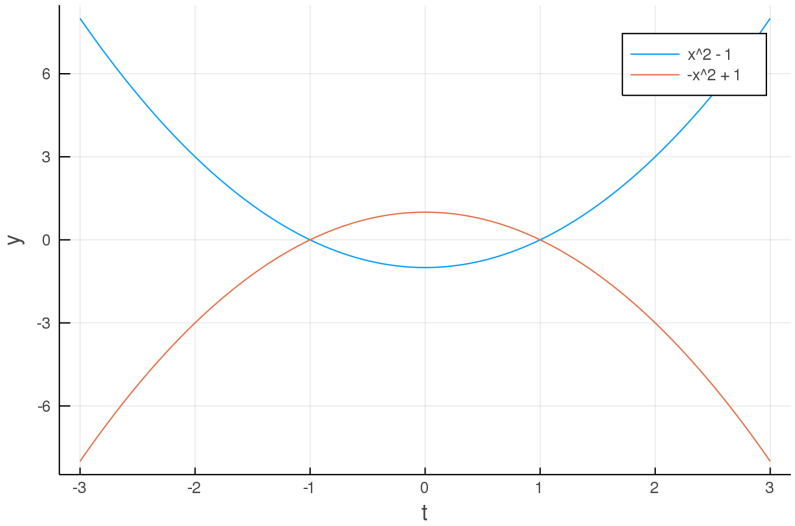
Slika 18: Zadatak 3.

Zadatak 4. Napisati kod koji generiše periodični signal prikazan na slici ispod.



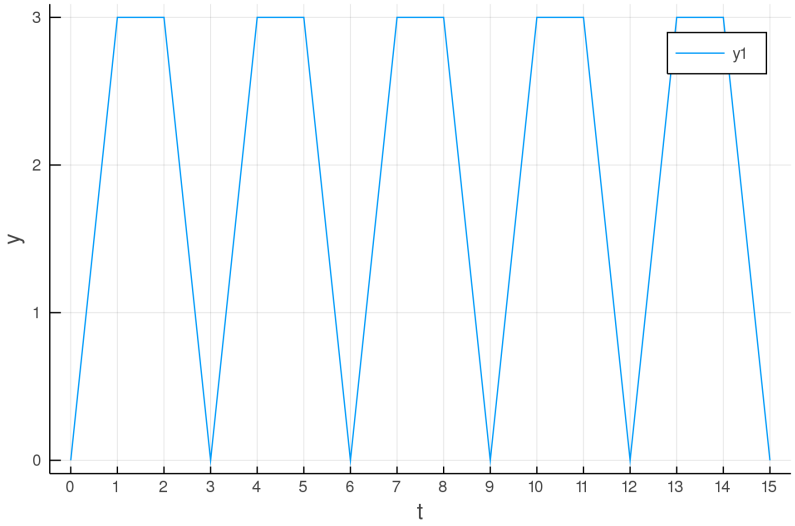
Slika 19: Zadatak 4.

Zadatak 5. Napisati kod koji generiše signal prikazan na slici ispod.



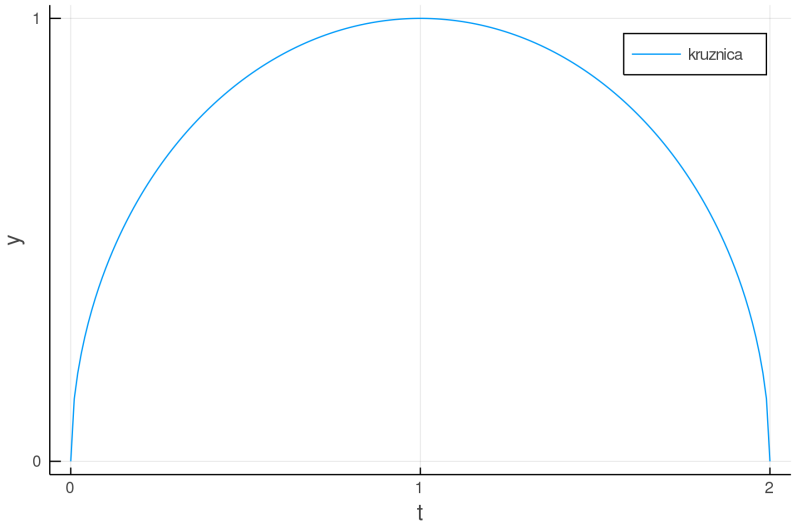
Slika 20: Zadatak 5.

Zadatak 6. Napisati kod koji generiše periodični signal prikazan na slici ispod.



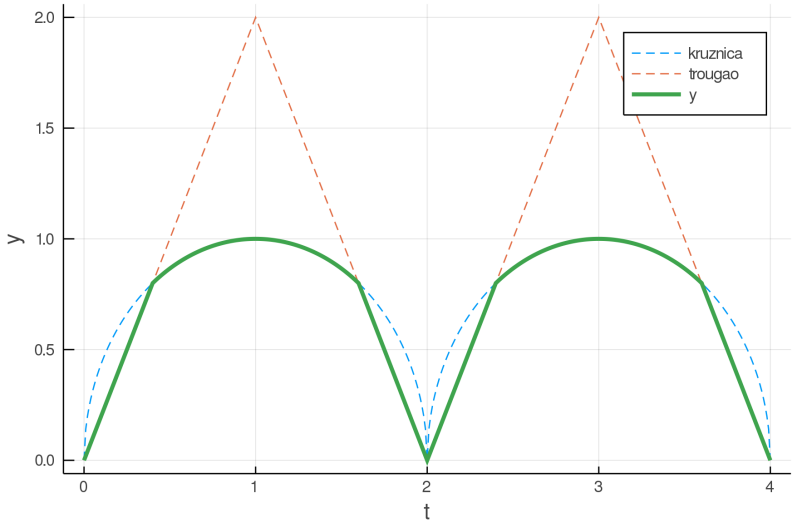
Slika 21: Zadatak 6.

Zadatak 7. Napisati kod koji generiše periodični signal prikazan na slici ispod.



Slika 22: Zadatak 7.

Zadatak 8. Napisati kod koji generiše periodični signal prikazan na slici ispod.



Slika 23: Zadatak 8.

Literatura

- *Plots* dokumentacija http://docs.juliaplots.org/latest/
- *Julia* programski jezik (sajt) https://julialang.org/
- *Think Julia* (*online* knjiga) https://benlauwens.github.io/ThinkJulia.jl/latest/book.html.