Modeliranje i Simulacija Sistema

Helouuu!

U nadi da ti Modeliranje i Simulacija Sistema ne budu totalni "bauk" nastala je skripta koju trenutno čitaš i verujem lutaš u potrazi za rešenjima zadataka.

Pred tobom se nalaze rešenja zadataka za samostalni rad sa računarskih i auditornih vežbi urađeni na lakši ili teži način.

Puno uspeha u učenju, Danijel Jovanović

01 Uvod u Juliju

```
# Zadatak 1: Za proizvoljnu kvadratnu matricu A, izdvojiti sve parne kolone.
using LinearAlgebra

A = [1 4 -2 9 6; -1 0 0 3 7; 99 3 -3 4 7; 5 -6 0 -8 3; 1 2 3 4 5]
parneKolone = A[:, 2:2:end]
```

```
5×2 Matrix{Int64}:
  4   9
  0   3
  3   4
 -6  -8
  2   4
```

```
# Zadatak 2: Za proizvoljnu kvadratnu matricu A, izdvojiti sve elemente koji su deljivi sa 9
using LinearAlgebra

A = [1 4 -2 9 6; -1 0 0 3 7; 99 3 -3 4 7; 5 -6 0 -8 3; 1 2 3 4 5]
deljiviSa9 = A[rem.(A, 9) .== 0]
```

```
5-element Vector{Int64}:
 99
  0
  0
  0
  9
```

```
# Zadatak 3. Za proizvoljnu kvadratnu matricu A, izdvojiti elemente koji se nalaze
# na preseku parnih vrsta i parnih kolona.
using LinearAlgebra

A = [1 4 -2 9 6; -1 0 0 3 7; 99 3 -3 4 7; 5 -6 0 -8 3; 1 2 3 4 5]
presekParneVrsteParneKolone = A[2:2:end, 2:2:end]
```

```
2×2 Matrix{Int64}:
  0   3
 -6  -8
```

```
# Zadatak 4: Napisati funkciju koja određuje zbir svih elemenata matrice A
# gde je m broj vrsta, a n broj kolona, koji imaju osobinu da je zbir indeksa (i + j)
# paran broj (A11 + A13 + ...)
A = [1 4 -2 9 6; -10 -10 0 3 7; 99 3 -3 4 7; 5 -6 0 -8 3; 1 2 3 4 5]

function sumiranje(A)
```

```
    redova, kolona = size(A)
    suma = 0;

    for i in 1:redova
        for j in 1:kolona
            if (i + j) % 2 == 0
                suma += A[i, j]
            end
        end
    end

    return suma
end

sumiranje(A)
```

```
96
```

```
# Zadatak 5. Napisati funkciju koja za zadatu kvadratnu matricu A, određuje:
# vektor m koji se formira od elemenata sa glavne dijagonale matrice A.
# skalar s koji predstavlja srednju vrednost elemenata iznad glavne dijagonale matrice
# A. (može se koristiti funkcija mean() iz programskog paketa Statistics)
using LinearAlgebra
using Statistics

A = [1 2 3; 4 5 6; 7 8 9]

# vektor m koji se formira od elemenata sa glavne dijagonale matrice A
function glavnaDijagonala(A)
    return diag(A)
end

# skalar s koji predstavlja srednju vrednost elemenata iznad glavne dijagonale matrice
# A. (može se koristiti funkcija mean() iz programskog paketa Statistics)
function avgIznadGlavneDijagonale(A)
    gornjaTrougaona = ones(size(A))
    gornjaTrougaona = triu(gornjaTrougaona, 1)
    maska = convert.(Bool, gornjaTrougaona)

    s = mean(A[maska])

    return s
end

println("m = ", glavnaDijagonala(A))
println("s = ", avgIznadGlavneDijagonale(A))
```

```
m = [1, 5, 9]
s = 3.6666666666666665
```

```
# Zadatak 6. Napisati funkciju koja za zadate kvadratne matrice A i B istih dimenzija određuje:
# vektor m koji se sastoji od elemenata ispod glavne dijagonale matrice A koji su
# pozitivni celi brojevi deljivi sa 3.
# skalar s koji predstavlja srednju vrednost elemenata sa sporedne dijagonale matrice
# B koji su veći od srednje vrednosti elemenata sa glavne dijagonale matrice A.

using LinearAlgebra
using Statistics

# vektor m koji se sastoji od elemenata ispod glavne dijagonale matrice A koji su
# pozitivni celi brojevi deljivi sa 3.
function pozitivniCeliBrojeviDeljiviSa3(A)
    donjaTrougaona = ones(size(A))
    donjaTrougaona = tril(donjaTrougaona, -1)
    maska = convert.(Bool, donjaTrougaona)

    m = A[maska]
    m = m[(m .>= 0) .& (rem.(m, 3) .== 0)]
```

```
    return m;
end

# skalar s koji predstavlja srednju vrednost elemenata sa sporedne dijagonale matrice
# B koji su veći od srednje vrednosti elemenata sa glavne dijagonale matrice A.
function avgSporednaDijagonalaAGlavnaB(A, B)
    # srednja vrednost elemenata sa glavne dijagonale matrice a
    glavnaDijagonala = diag(A)
    avgGlavnaDijagonala = mean(glavnaDijagonala)

    # vrednosti matrice B sa sporedne dijagonale veci od proseka glavne dijagonale A
    sporednaDijagonala = diag(reverse(B, dims = 2))
    s = mean(sporednaDijagonala[sporednaDijagonala .>= avgGlavnaDijagonala])

    return s
end

# main.jl
A = [1 -2 3; 3 3 6; 7 9 8]
B = [9 8 7; 6 5 4; 3 2 1]

println("m = ", pozitivniCeliBrojeviDeljiviSa3(A))
println("s = ", avgSporednaDijagonalaAGlavnaB(A, B))
```

```
m = [3, 9]
s = 6.0
```

```
# Zadatak 7: Za podatke iz tabele T (Primer 4 - Složeni primeri) napisati kod koji određuje:
#    - koliko je ženskih, a koliko muških osoba (poželjno je prikazati i njihova imena),
#    - prosečnu visinu i težinu ženskih osoba,
#    - prosečnu visinu i težinu muških osoba,
#    - najstariju i najmlađu osobu,
#    - standardnu devijaciju za visinu.

############### TABELA ################
# Ime   Pol Starost Težina Visina
# Ana     ž 20      46     160
# Bojan   m 24      52     165
# Vlada   m 24      95     195
# Gordana ž 30      57     160
# Dejan   m 36      84     185
# Zoran   m 22      80     180

using LinearAlgebra
using Statistics

T = ["Ime"    "Pol" "Starost" "Tezina" "Visina";
     "Ana"     "z"        20       46      160;
     "Bojan"   "m"        24       52      165;
     "Vlada"   "m"        24       95      195;
     "Gordana" "z"        30       57      160;
     "Dejan"   "m"        36       84      185;
     "Zoran"   "m"        22       80      180]

# koliko je ženskih, a koliko muških osoba (poželjno je prikazati i njihova imena)
izdvojeniPodaci  = T[:, 2]
zenskihOsoba = findall(izdvojeniPodaci .== "z")
muskihOsoba  = findall(izdvojeniPodaci .== "m")

println("Zenskih osoba: ",  length(zenskihOsoba), "\nMuskih  osoba: ",  length(muskihOsoba))
println("\nZenske osobe -> ", T[zenskihOsoba, 1])
println("Muske  osobe -> ", T[muskihOsoba, 1])

# prosečnu visinu i težinu ženskih osoba
avgVisinaZene = mean(T[zenskihOsoba, 5])
avgTezinaZene = mean(T[zenskihOsoba, 4])

println("\nProsecna visina zenskih osoba: ", avgVisinaZene, " cm")
println("Prosecna tezina zenskih osoba: ", avgTezinaZene, " kg")

# prosečnu visinu i težinu muških osoba
avgVisinaMuskarci = mean(T[muskihOsoba, 5])
```

```
avgTezinaMuskarci = mean(T[muskihOsoba, 4])

println("\nProsecna visina muskih osoba: ", avgVisinaMuskarci, " cm")
println("Prosecna tezina muskih osoba: ", avgTezinaMuskarci, " kg")

# najstariju i najmlađu osobu
starosti = T[2:end, 3]
najmladja  = minimum(starosti)
najstarija = maximum(starosti)

println("\nNajmladja osoba  ima: ", najmladja, " god\nNajstarija osoba ima: ", najstarija, " god")

# standardnu devijaciju za visinu
devijacija = 0.042 * (starosti .- 21) - 0.0015 * (starosti .- 21).^2

println("\nDevijacija visine: ", round.(mean(devijacija) * 100), "%")
```

```
Zenskih osoba: 2
Muskih  osoba: 4

Zenske osobe -> Any["Ana", "Gordana"]
Muske  osobe -> Any["Bojan", "Vlada", "Dejan", "Zoran"]

Prosecna visina zenskih osoba: 160.0 cm
Prosecna tezina zenskih osoba: 51.5 kg

Prosecna visina muskih osoba: 181.25 cm
Prosecna tezina muskih osoba: 77.75 kg

Najmladja osoba  ima: 20 god
Najstarija osoba ima: 36 god

Devijacija visine: 13.0%
```

```
# Zadatak 8: Napisati funkciju koja određuje poziciju nenultih elemenata proizvoljne matrice.
# Zadatak rešiti bez korišćenja funkcije findall.
function indeksiNenultihElemenata(A)
    indeksi = []
    dimenzije = size(A)

    for i in 1:dimenzije[1]
        for j in 1:dimenzije[2]
            if A[i, j] != 0
                push!(indeksi, [i, j])
            end
        end
    end

    return indeksi
end

A = [0 0 5; 0 2 3]
println("Indeksi nenultih elemenata matrice A su: ", indeksiNenultihElemenata(A))
```

```
Indeksi nenultih elemenata matrice A su: Any[[1, 3], [2, 2], [2, 3]]
```

```
# Zadatak 9: Napisati funkciju, po uzoru na funkciju prod, koja određuje proizvod
# svih elemenata vektora.
function _prod(v)
    p = 1

    for i in 1:length(v)
        p *= v[i]
    end

    return p
end
```

```
v = [1, 2, 3, 4, 1, 2]
println("Prod = ", _prod(v))
```

```
Prod = 48
```

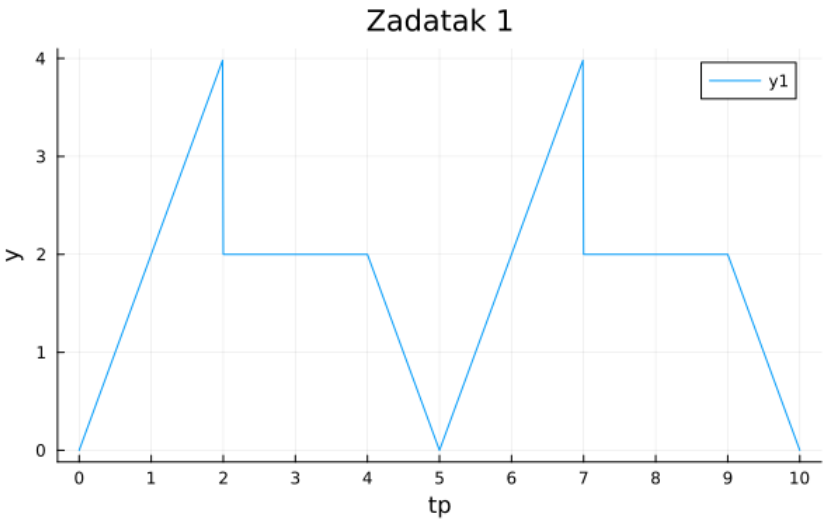
```
# Zadatak 10: Napisati funkciju, po uzoru na funkciju sum, koja određuje sumu elemenata
# proizvoljne matrice. Implementirati opcioni ili imenovani parametar funkcije na osnovu koga
# će se računati suma elemenata po vrstama ili po kolonama matrice.
function _sum(A; dims = 2) # racuna se po vrstama podrazumevano
    redovi, kolone  = size(A)
    suma = []
    tmpSuma    = 0

    for i in 1:redovi
        for j in 1:kolone
            if dims == 2 # suma po vrstama
                tmpSuma += A[i, j]
            else         # suma po kolonama
                tmpSuma += A[j, i]
            end
        end
        push!(suma, tmpSuma)
        tmpSuma = 0
    end

    return suma
end

A = [1 4 -2 9 6; -10 -10 0 3 7; 99 3 -3 4 7; 5 -6 0 -8 3; 1 2 3 4 5]
println("Suma po vrstama  matrice A iznosi: ", _sum(A, dims = 2))
println("Suma po kolonama matrice A iznosi: ", _sum(A, dims = 1))
```

```
Suma po vrstama  matrice A iznosi: Any[18, -10, 110, -6, 15]
Suma po kolonama matrice A iznosi: Any[96, -7, -2, 12, 28]
```

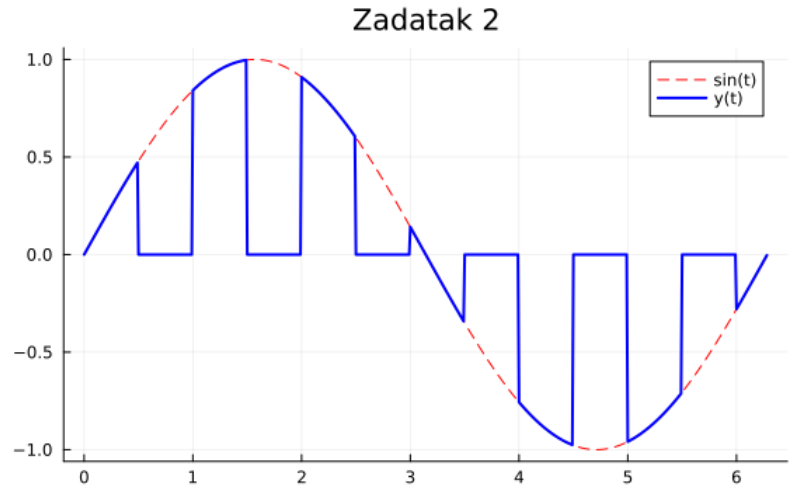
---

02 Osnove Crtanja Signala

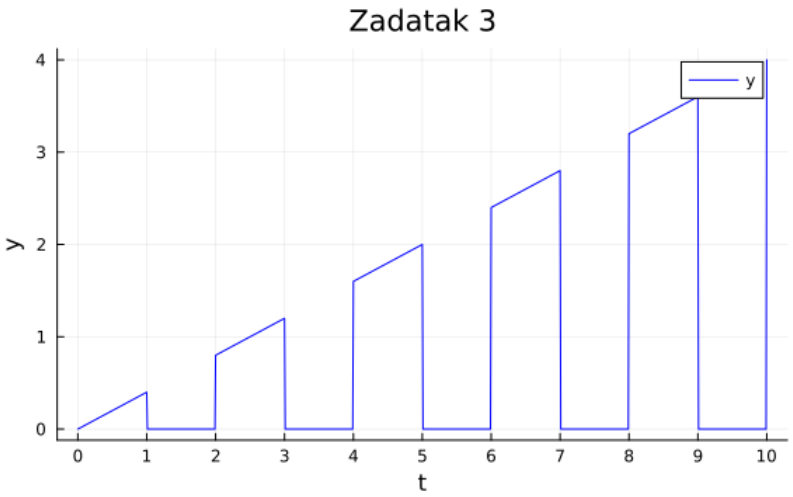
```
# Zadatak 1. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots

t = 0:0.01:10
tp = rem.(t, 5)

y1 = (2 * tp) .* (tp .< 2)
y2 = 2 * ((tp .>= 2) .& (tp .< 4))

# prava simetricna na y1 -> y = -2t sada je potrebno
# izracunati na
# y = k*t + n, uvrsti se jedna tacka npr A(4, 2)
# n = 2 + 2 * 4 -> n = 10
y3 = ((- 2 * tp) .+ 10) .* ((tp .>= 4) .& (tp .<= 5))

y = y1 .+ y2 .+ y3

plot(t, y, title="\nZadatak 1\n", xticks=0:10)
xlabel!("tp")
ylabel!("y")
```



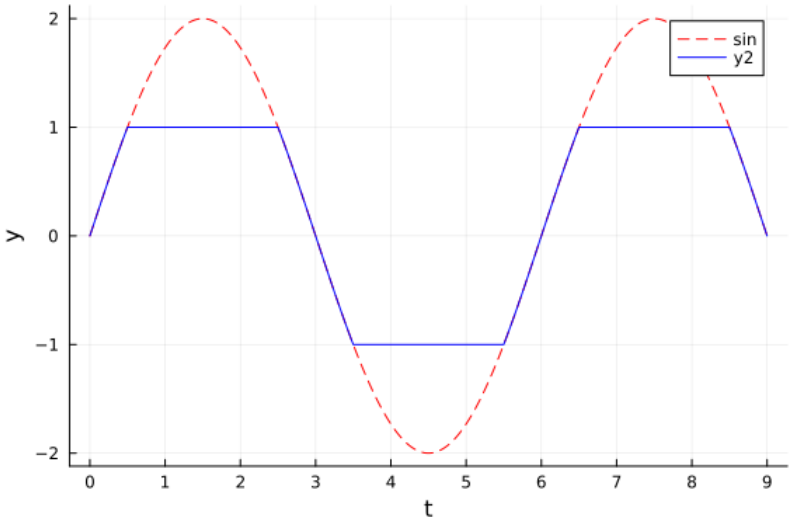
```julia
# Zadatak 2. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots
using LinearAlgebra

t  = 0:0.01:2π;
tp = rem.(t, 1)

y = sin.(t .* (tp .< 0.5));
ys = sin.(t);

plot(t, ys, color=:red, label="sin(t)", linestyle=:dash)
plot!(t, y, lw=2, title="\nZadatak 2\n", label="y(t)", color=:blue)
```



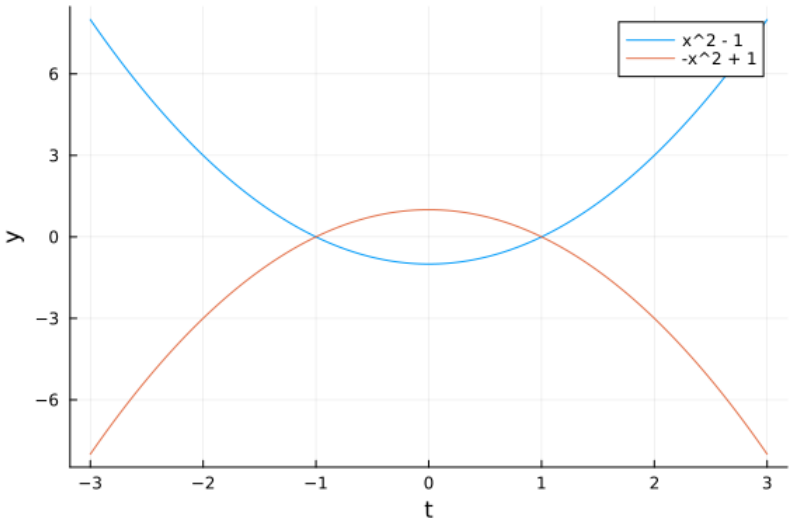
```julia
# Zadatak 3. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots

t = 0:0.01:10
tp = rem.(t, 2)

# jednacina prave kroz dve tacke za pravu
y1 = 4/10 * t
y = y1 .* (tp .<= 1) # ako je manje od 1 onda je y = 4, u suprotnom y = 0

plot(t, y, title="\nZadatak 3\n", label="y", xticks=0:10, color=:blue)

xlabel!("t")
ylabel!("y")
```



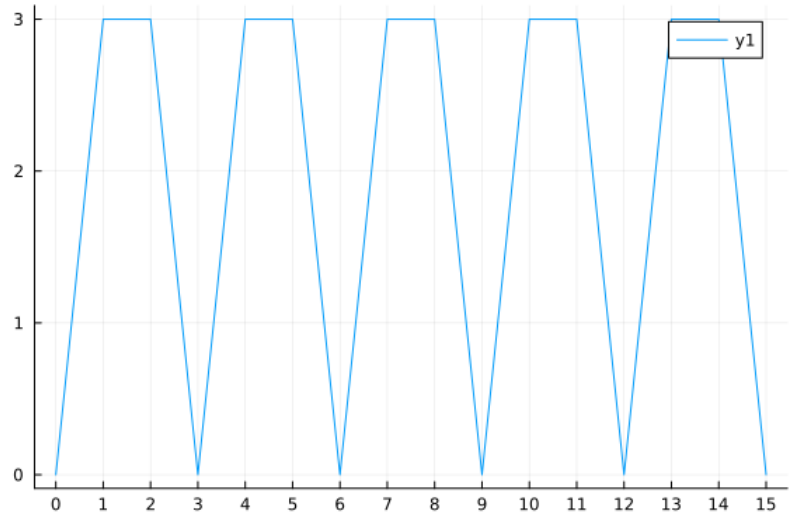
```julia
# Zadatak 4. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots

t = 0:0.01:9

y1 = 2 * sin.(pi / 3 * t)
y2 = min.(y1, 1)
y = max.(y2, -1)

plot(t, y1, linestyle=:dash, color=:red, label="sin")
plot!(t, y, xticks=0:9, lw=1, color=:blue)

xlabel!("t")
ylabel!("y")
```



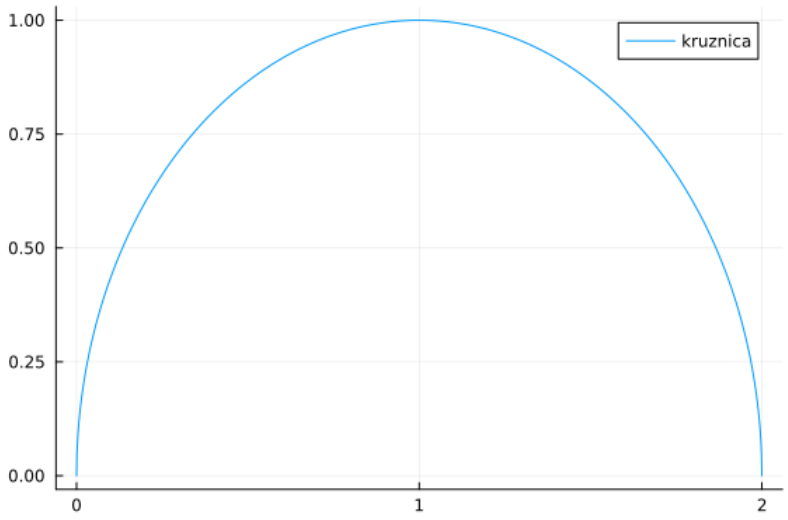
```julia
# Zadatak 5. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots

t = -3:0.01:3

y1 = t.^2 .- 1
y2  = -y1

plot(t, y1, label="x^2 - 1")
plot!(t, y2, label="-x^2 + 1")

xlabel!("t")
ylabel!("y")
```



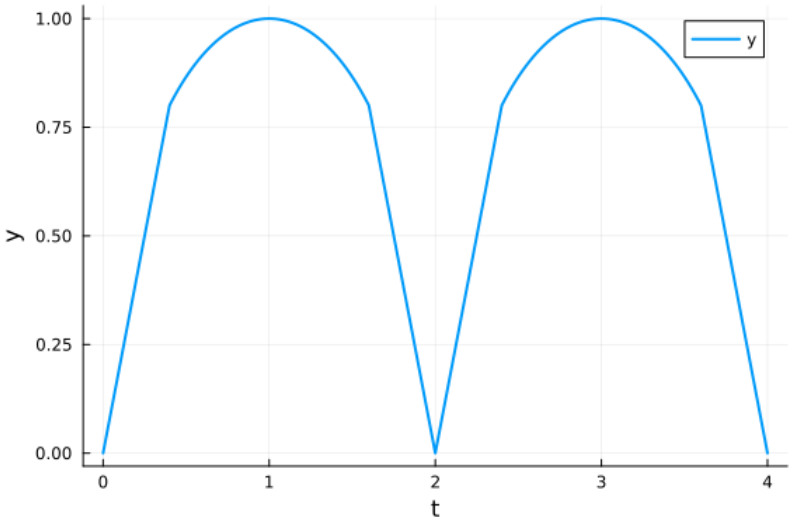
```
# Zadatak 6. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots

t = 0:0.001:15
tp = rem.(t, 3)

y = (3 * tp) .* (tp .<= 1) .+
    (3 * ((tp .> 1) .& (tp .<= 2))) .+
    ((-3 * tp .+ 9) .* (tp .> 2))

plot(t, y, xticks=0:15)
```



```
# Zadatak 7. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots

t = 0:0.001:2
y = sqrt.(1 .- ((t .- 1) .^ 2))

plot(t, y, xticks=0:2, label="kruznica")
```

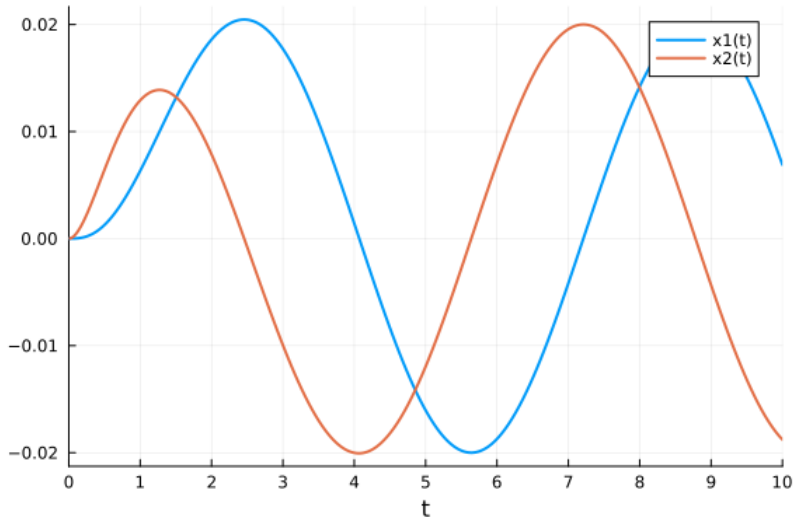


```julia
# Zadatak 8. Napisati kod koji generiše periodični signal prikazan na slici ispod.
using Plots

# kruznica
t_kruznica_domen = 0:0.001:4

t_kruznica = rem.(t_kruznica_domen, 2)
y_kruznica = sqrt.(1 .- ((t_kruznica .- 1) .^ 2))

# trougao
t_trougla = 0:0.001:4
tp = rem.(t_trougla, 2)

y1 = (2 * tp) .* (tp .<= 1)
y2 = (-2 * tp .+ 4) .* (tp .> 1)

y_trougao = y1 + y2

# konacni signal
t_domen = 0:0.001:4

y1_a = y_trougao .* (tp .< 0.4)
y1_b = y_trougao .* (tp .> 1.6)
y2 = y_kruznica .* ((tp .>= 0.4) .& (tp .<= 1.6))

y = y1_a .+ y1_b .+ y2

plot(t_domen, y, xticks=0:4, label="y", lw=2)

xlabel!("t")
ylabel!("y")
```

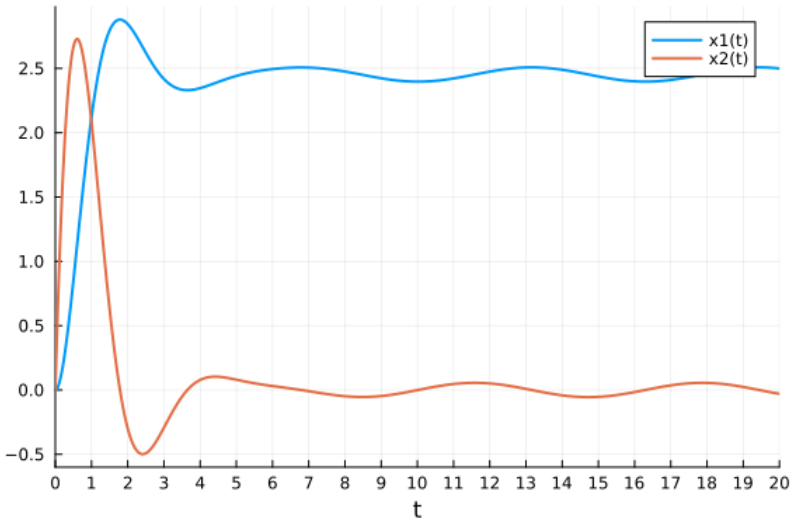


03b Mehanički Translatorni Sistemi

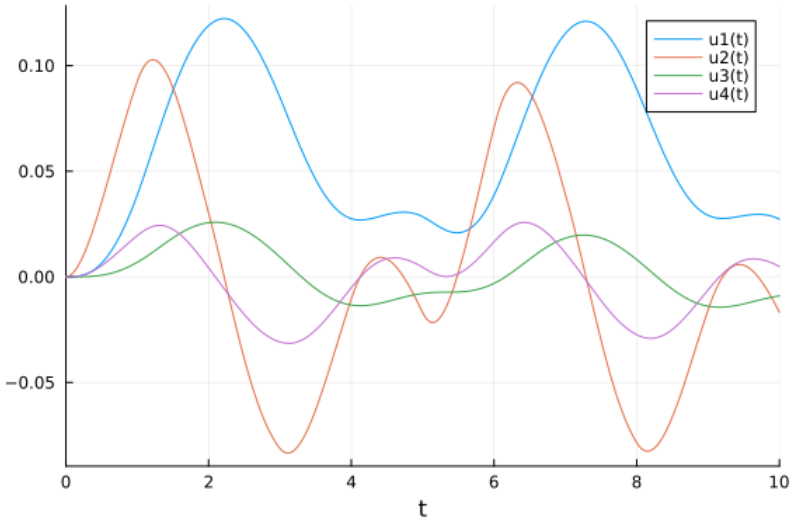
```
# Primer 1
using LinearAlgebra, Plots, DifferentialEquations

function diferencijalneJednačine!(dx, x, p, t)
    m, c1, c2, k = p
    f = sin(t)

    dx[1] = x[2]
    dx[2] = 1 / m * (f - k * x[1] - x[2] * (c1 + c2))
end

# main.jl
t = (0.0, 10.0)
x0 = [0, 0]
p = (10.0, 20.0, 20.0, 40.0)

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)
plot(rešenje, label = ["x1(t)" "x2(t)"], lw = 2, xticks = 0:10)
```



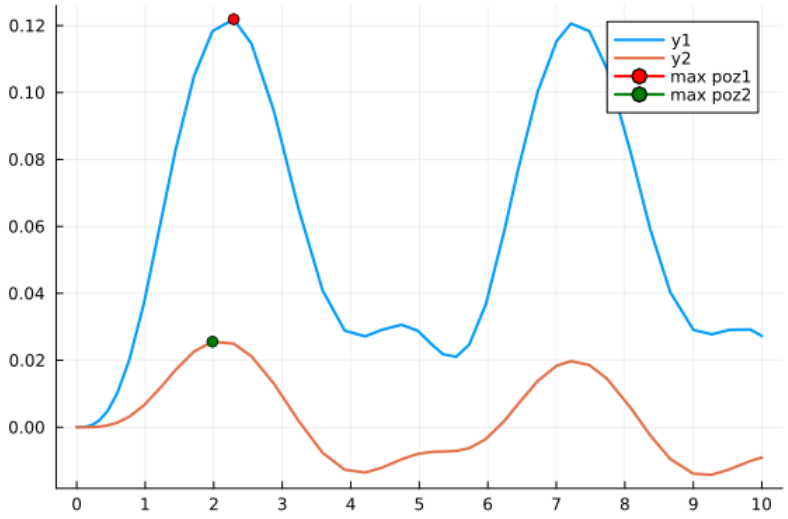

```
# Primer 2
using LinearAlgebra, Plots, DifferentialEquations

function sistem!(dx, x, p, t)
    m, c, k, g = p
    f = cos(t)
    dx[1] = x[2]
    dx[2] = 1 / m * (f + m * g - k * x[1] - c * x[2])
end

t = (0.0, 20.0)
x0 = [0.0, 0.0]
p = (5.0, 10.0, 20.0, 9.81)

prob = ODEProblem(sistem!, x0, t, p)
sol = solve(prob)

plot(sol, label=["x1(t)" "x2(t)"], lw=2, xticks=0:20)
```



```
# Primer 3
using LinearAlgebra, Plots, DifferentialEquations

function signal(t)
        tp = rem.(t, 5)

        y1 = (( 4 * tp) .* (tp .< 1)) .+ (4 .* ((tp .>= 1) .& (tp .< 2)))
        y2 = ((-2 * tp .+ 8) .* ((tp .>= 2) .& (tp .< 3)))
        y3 = (( 2 * ((tp .>= 3) .& (tp .< 4)))) .+ ((-2 * tp .+ 10) .* (tp .>= 4))

        y = y1 + y2 + y3
end

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c1, c2, c3, k1, k2 = p
    f = signal(t)

    dx[1] = x[2]
    dx[2] = -1 / m1 * (-f + c1 * (x[2] - x[4]) + k1 * x[1])
    dx[3] = x[4]
    dx[4] = -1 / m2 * (-c1 * (x[2] - x[4]) + (c2 + c3) * x[4] + k2 * x[3])
end

# main.jl
t = (0.0, 10.0)
p = (10.0, 15.0, 20.0, 20.0, 20.0, 40.0, 40.0)
x0 = [0.0, 0.0, 0.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```



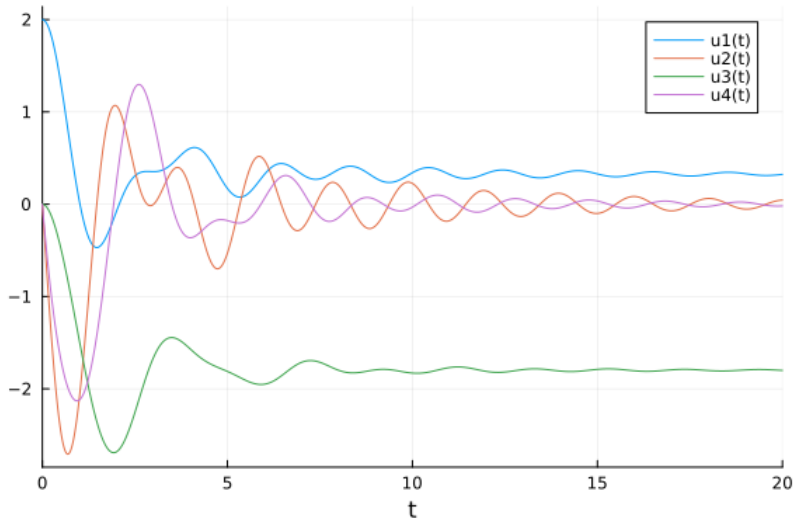
```
# pod d)
pozicija1 = [x[1] for x in rešenje.u]
pozicija2 = [x[3] for x in rešenje.u]

~, index1 = findmax(abs.(pozicija1))
~, index2 = findmax(abs.(pozicija2))

plot(rešenje.t, [pozicija1, pozicija2], lw = 2, xticks = 0:10)
plot!([rešenje.t[index1]], [pozicija1[index1]], markershape = :o, color = :red, lw = 2, label = "max poz1")
plot!([rešenje.t[index2]], [pozicija2[index2]], markershape = :o, color = :green, lw = 2, label = "max poz2")
```



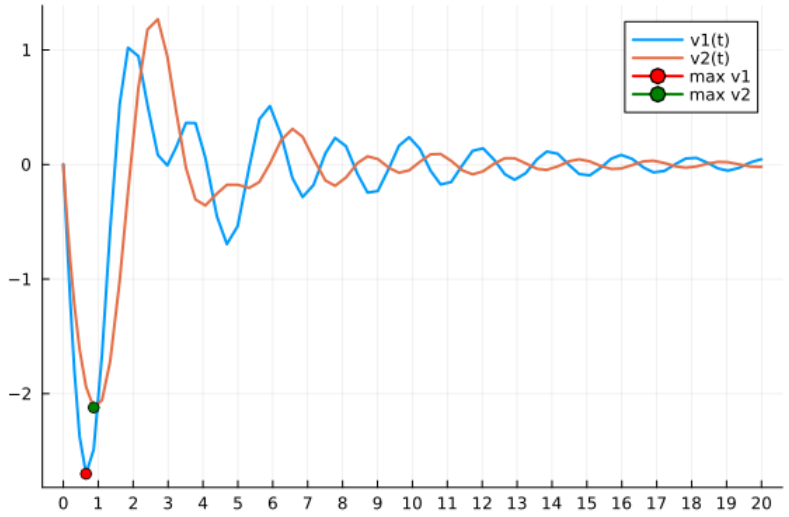

```
# pod e)
put1 = sum(abs.(diff(pozicija1)))
put2 = sum(abs.(diff(pozicija2)))

println("Predjeni put prvog tela iznosi: ", put1, "\nPredjeni put drugog tela iznosi: ", put2)
```

```
Predjeni put prvog tela iznosi: 0.4254534389217422
Predjeni put drugog tela iznosi: 0.13700337136224794
```

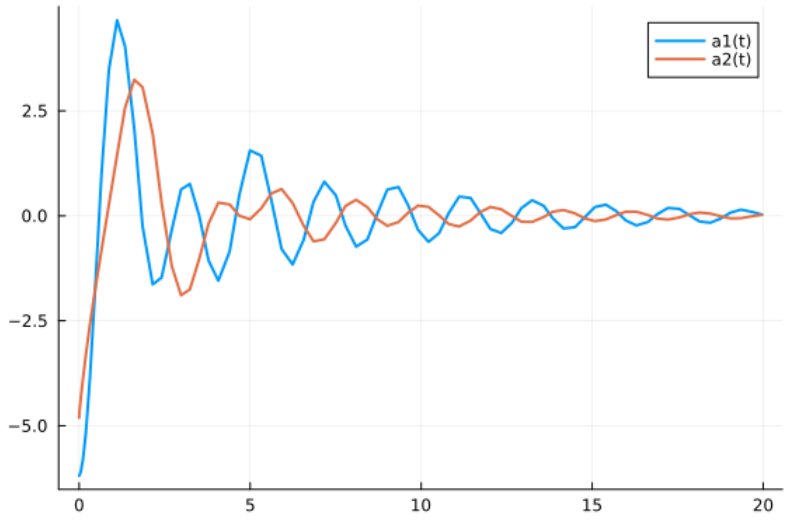
```
# Primer 4
using LinearAlgebra, Plots, DifferentialEquations

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c, k1, k2, k3, g = p

    dx[1] = x[2]
    dx[2] = 1 / m1 * (m1 * g + k2 * x[3] - (k1 + k2) * x[1])
    dx[3] = x[4]
    dx[4] = 1 / m2 * (k2 * x[1] - m2 * g - c * x[4] - (k2 + k3) * x[3])
end

# main.jl
t = (0.0, 20.0)
p = (5.0, 8.0, 10.0, 20.0, 20.0, 20.0, 9.81)
x0 = [2.0, 0.0, 0.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```



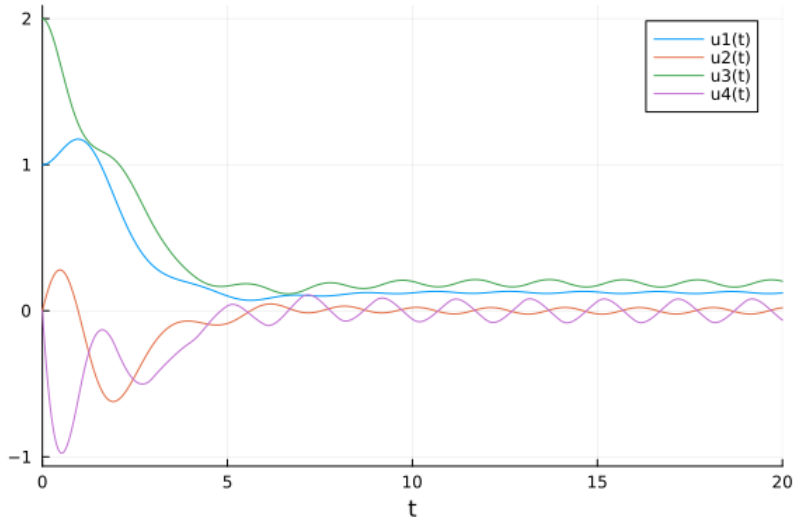

```
# pod d)
v1 = [x[2] for x in rešenje.u]
v2 = [x[4] for x in rešenje.u]

~, index1 = findmax(abs.(v1))
~, index2 = findmax(abs.(v2))

plot(rešenje.t, [v1 v2], lw = 2, xticks = 0:20, label = ["v1(t)" "v2(t)"])
plot!([rešenje.t[index1]], [v1[index1]], markershape = :o, color = :red, lw = 2, label = "max v1")
plot!([rešenje.t[index2]], [v2[index2]], markershape = :o, color = :green, lw = 2, label = "max v2")
```




```
# pod e)
a1 = diff(v1) ./ diff(rešenje.t)
a2 = diff(v2) ./ diff(rešenje.t)

plot(rešenje.t[1:end-1], [a1, a2], lw = 2, label = ["a1(t)" "a2(t)"])
```

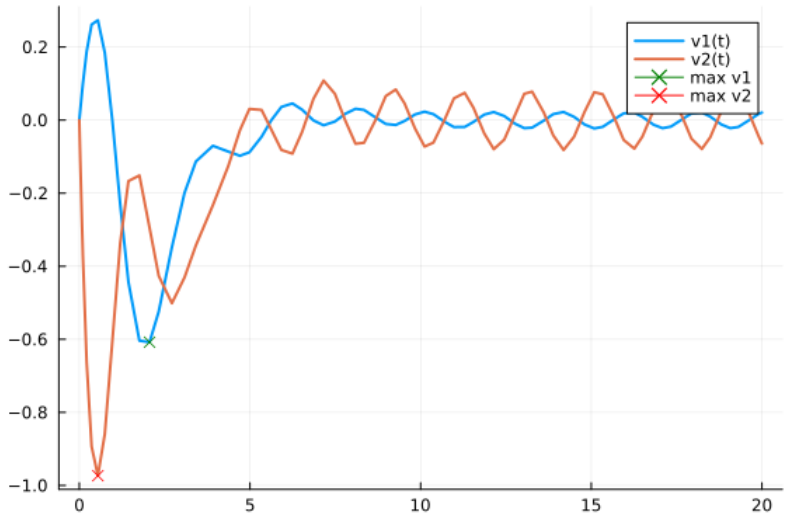


```julia
# Zadatak 1
using LinearAlgebra, Plots, DifferentialEquations

function signal(t)
    tp = rem.(t, 2)

    y = ((5 * tp) .* (tp .< 1)) +
        ((-5 * tp .+ 10) .* (tp .>= 1))
end

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c1, c2, c3, k1, k2 = p
    f = signal(t)

    dx[1] = x[2]
    dx[2] = -1 / m1 * (c1 * x[2] + k1 * x[1] + k2 * (x[1] - x[3]))
    dx[3] = x[4]
    dx[4] = 1 / m2 * (f + k2 * (x[1] - x[3]) - x[4] * (c2 + c3))
end

# main.jl
t = (0.0, 20.0)
p = (20.0, 10.0, 10.0, 10.0, 10.0, 20.0, 40.0)
x0 = [1.0, 0.0, 2.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```




```julia
# pod d)
v1 = [x[2] for x in rešenje.u]
v2 = [x[4] for x in rešenje.u]
```

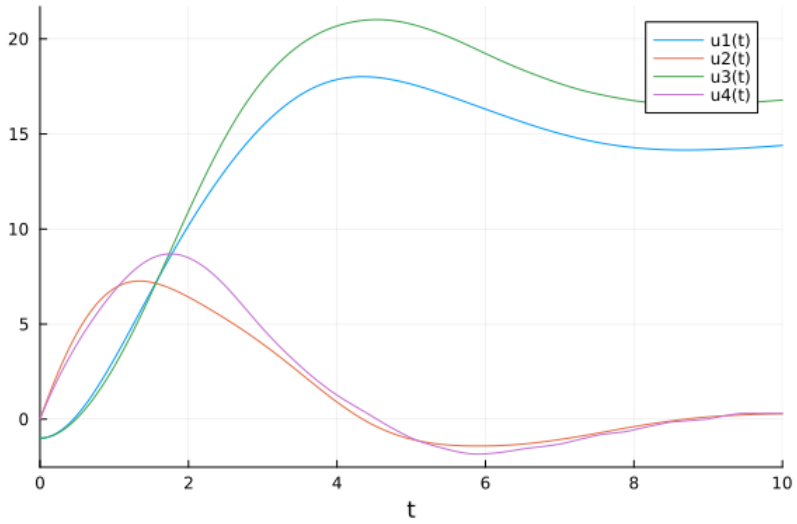
```
~, index1 = findmax(abs.(v1))
~, index2 = findmax(abs.(v2))

plot(rešenje.t, [v1 v2], lw = 2, label = ["v1(t)" "v2(t)"])
plot!([rešenje.t[index1]], [v1[index1]], label = "max v1", markershape = :x, color = :green)
plot!([rešenje.t[index2]], [v2[index2]], label = "max v2", markershape = :x, color = :red)
```



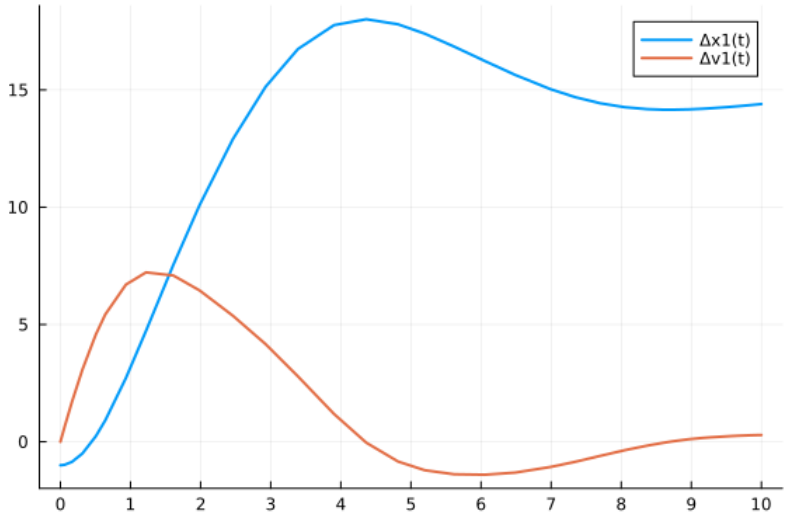

```
# pod e)
pozicija1 = [x[1] for x in rešenje.u]
pozicija2 = [x[3] for x in rešenje.u]

put1 = sum(abs.(diff(pozicija1)))
put2 = sum(abs.(diff(pozicija2)))

println("Predjeni put prvog tela je: ", put1, "\nPredjeni put drugog tela je: ", put2)
```

```
Predjeni put prvog tela je: 1.4916707139370056
Predjeni put drugog tela je: 2.5645552223731674
```

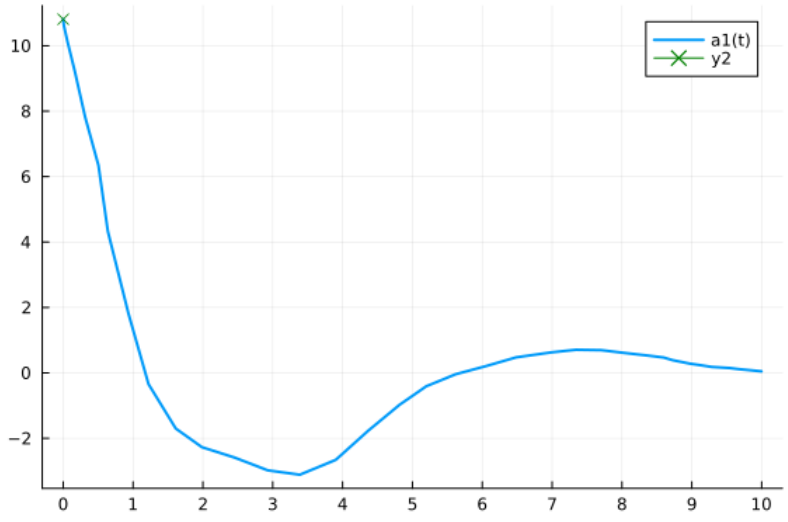
```
# Zadatak 2
using LinearAlgebra, Plots, DifferentialEquations

function signal(t)
    tp = rem.(t, 1)

    y = (1 / 2 * t) .* (tp .< 0.5) +
         (0 * tp) .* (tp .>= 0.5)
end

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c1, c2, c3, k1, k2, g = p
    f = signal(t)

    dx[1] = x[2]
    dx[2] = -1 / m1 * (c1 * x[2] + k2 * (x[1] - x[3]) + k1 * x[1] - m1 * g)
    dx[3] = x[4]
    dx[4] = -1 / m2 * (c2 * x[4] - m2 * g - k2 * (x[1] - x[3]) - f)
end

# main.jl
t = (0.0, 10.0)
p = (20.0, 10.0, 10.0, 10.0, 10.0, 20.0, 40.0, 9.81)
x0 = [-1.0, 0.0, -1.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```



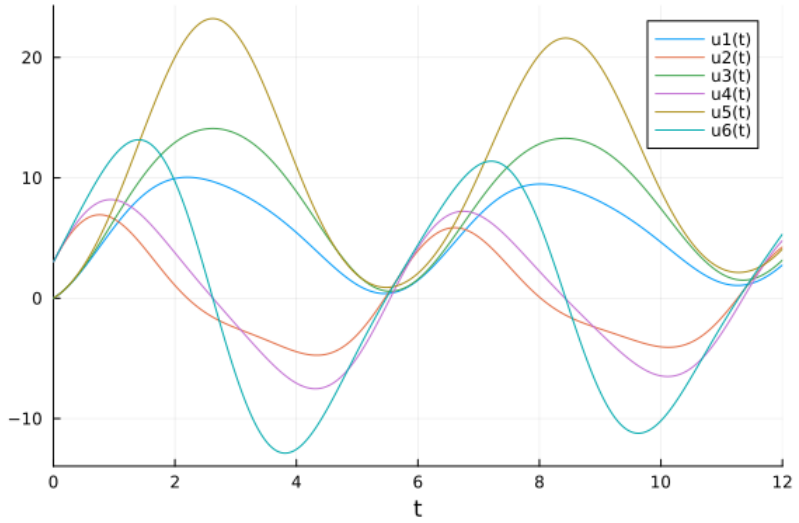

```
# pod d)
p1 = [x[1] for x in rešenje.u]
v1 = [x[2] for x in rešenje.u]

plot(rešenje.t, [p1 v1], xticks = 0:10, lw = 2, label = ["Δx1(t)" "Δv1(t)"] )
```



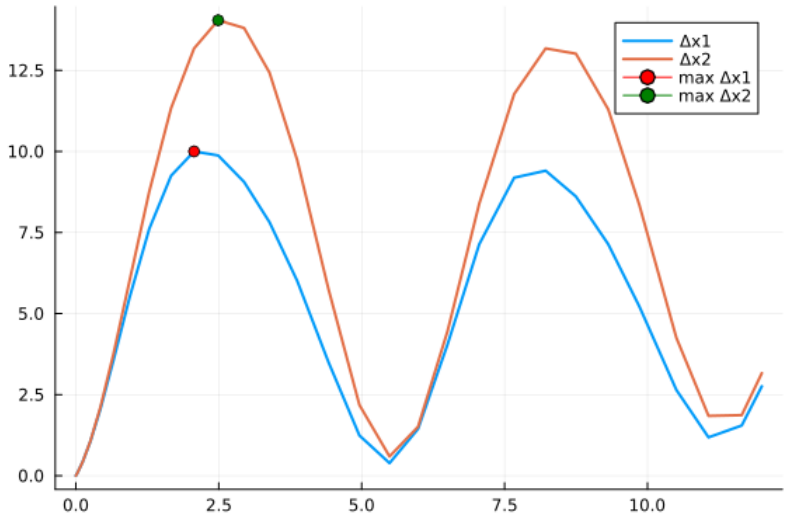

```
# pod e)
a1 = diff(v1) ./ diff(rešenje.t)
~, max_a1 = findmax(abs.(a1))

plot(rešenje.t[1:end-1], [a1], xticks = 0:10, lw = 2, label = "a1(t)")
plot!([rešenje.t[max_a1]], [a1[max_a1]], markershape = :x, color = :green)
```



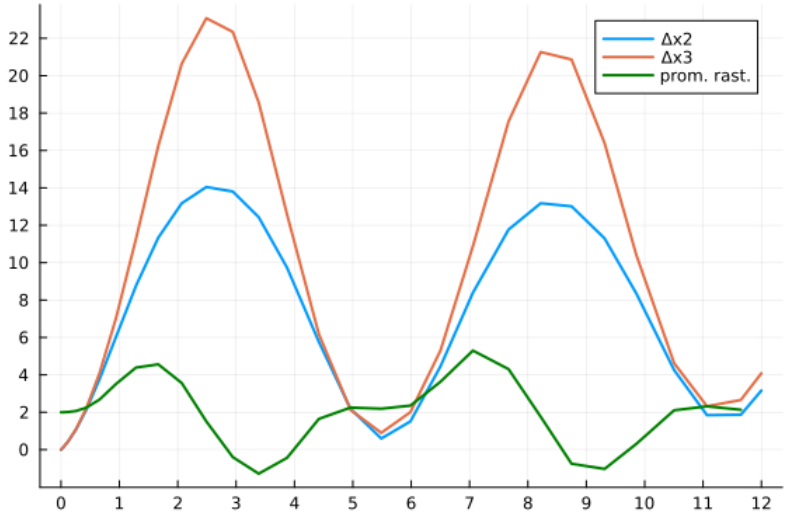
```julia
# Zadatak 3
using LinearAlgebra, Plots, DifferentialEquations

function signal(t)
    y = abs.(sin.(pi / 3 * t))
 end

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, m3, c, k, g = p
    f = signal(t)

    dx[1] = x[2]
    dx[2] = -1 / m1 * (c * (x[2] - x[4]) + k * (2 * x[1] - x[3]) - m1 * g)
    dx[3] = x[4]
    dx[4] = 1 / m2 * (c * (x[2] - x[4]) - k * (x[3] - x[5]) + k * (x[1] - x[3]) - k * x[3] + m2 * g)
    dx[5] = x[6]
    dx[6] = 1 / m3 * (k * (x[3] - x[5]) + m3 * g - f)
end

# main.jl
t = (0.0, 12.0)
p = (5.0, 10.0, 5.0, 10.0, 15.0, 9.81)
x0 = [0.0, 3.0, 0.0, 3.0, 0.0, 3.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```




```julia
# pod d)
# promena pozicija tela m1 i m2 i najudaljenije tacke
p1 = [x[1] for x in rešenje.u]
p2 = [x[3] for x in rešenje.u]

~, idx1 = findmax(abs.(p1))
~, idx2 = findmax(abs.(p2))

plot(rešenje.t, [p1, p2], lw = 2, label = ["Δx1" "Δx2"])
plot!([rešenje.t[idx1]], [p1[idx1]], markershape = :o, color = :red, label = "max Δx1")
plot!([rešenje.t[idx2]], [p2[idx2]], markershape = :o, color = :green, label = "max Δx2")
```

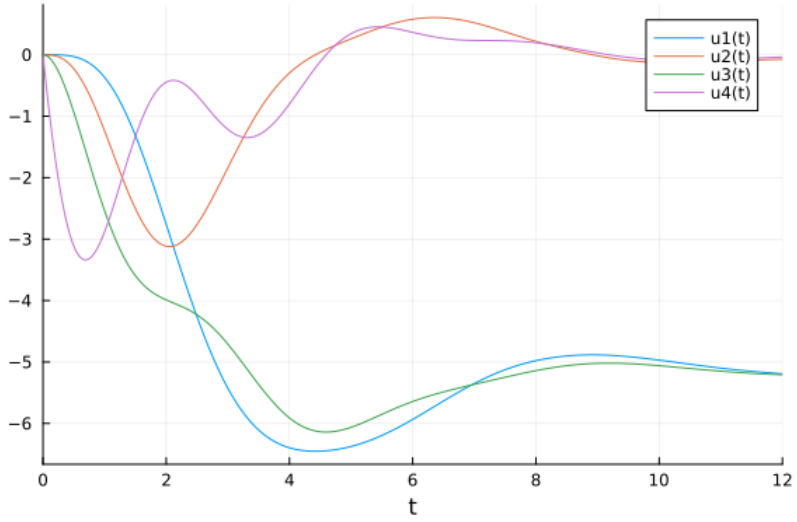



```
# pod e)
# promenu rastojanja između tela mase m2 i m3, ako je rastojanje
# u početnom trenutku iznosilo 2
p2 = [x[3] for x in rešenje.u]
p3 = [x[5] for x in rešenje.u]

abs_rastojanje_m2_m3 = abs.(p2 .- p3)
promena_rastojanja = diff(abs_rastojanje_m2_m3)
promena_rastojanja = promena_rastojanja .+ 2

plot([rešenje.t], [p2, p3], lw = 2, label = ["Δx2" "Δx3"], xticks = 0:12, yticks = 0:2:30)
plot!(rešenje.t[1:end-1], promena_rastojanja, lw = 2, color = :green, label = "prom. rast.")
```

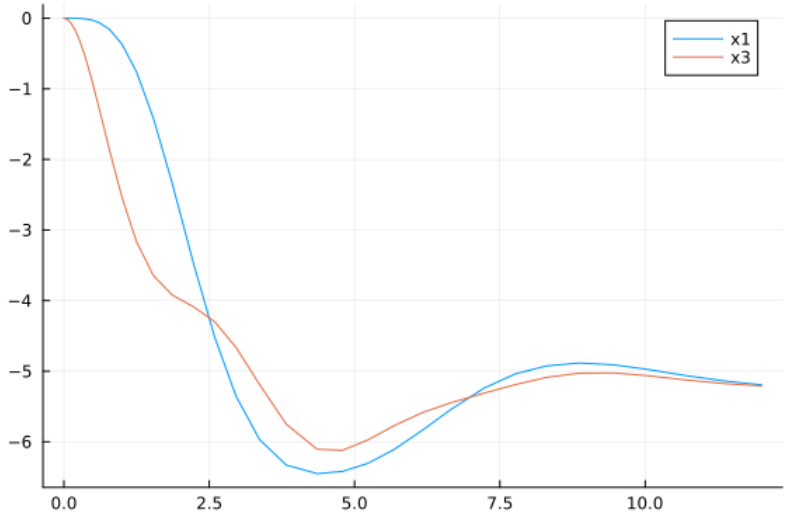


```
# Zadatak 4
using LinearAlgebra, Plots, DifferentialEquations

function signal(t)
    tp = rem.(t, 3)

    y_sinus = abs.(sin.(pi / 3 * t))
    y_prava = 1 / 3 * tp

    y = min.(y_sinus, y_prava)
end

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c1, c2, k1, k2, g = p
    f = signal(t)

    dx[1] = x[2]
    dx[2] = -1 / m1 * (c1 * x[2] + k1 * (x[1] - x[3]) - f)
    dx[3] = x[4]
    dx[4] = -1 / m2 * (c2 * x[4] - k1 * (x[1] - x[3]) + k2 * x[3] + m2 * g)
```

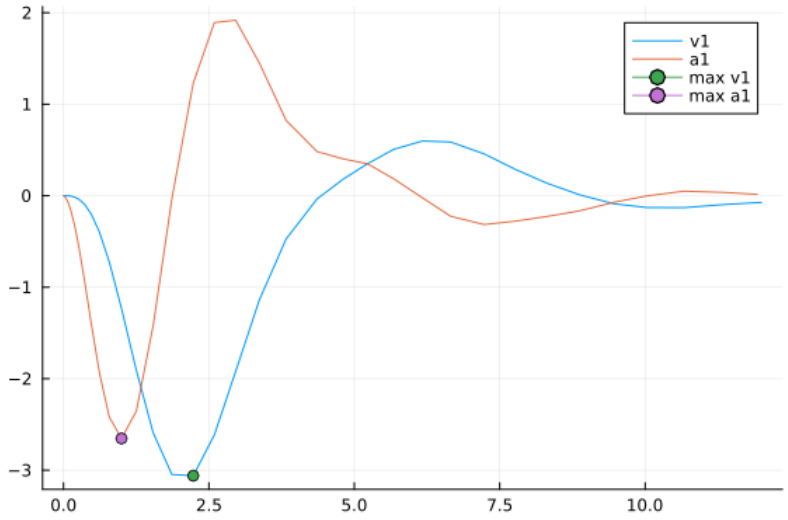
```
end

# main.jl
t = (0.0, 12.0)
p = (10.0, 8.0, 5.0, 10.0, 15.0, 15.0, 9.81)
x0 = [0.0, 0.0, 0.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```




```
# pod d
p1 = [x[1] for x in rešenje.u]
p2 = [x[3] for x in rešenje.u]

plot(rešenje.t, [p1 p2], label = ["x1" "x3"])
```



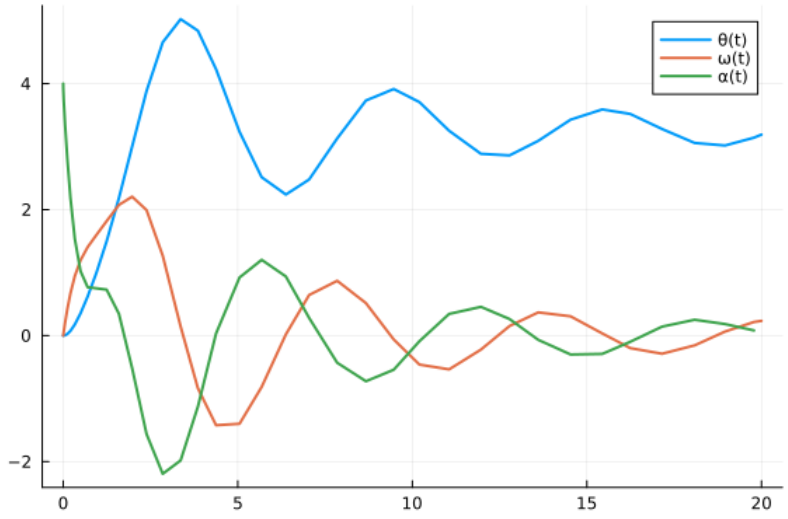

```
# pod e
v1 = [x[2] for x in rešenje.u]
a1 = diff(v1) ./ diff(rešenje.t)

~, idx1 = findmax(abs.(v1))
~, idx2 = findmax(abs.(a1))

plot([rešenje.t], [v1], label = "v1")
plot!([rešenje.t[1:end-1]], [a1], label = "a1")
plot!([rešenje.t[idx1]], [v1[idx1]], markershape = :o, label = "max v1")
plot!([rešenje.t[idx2]], [a1[idx2]], markershape = :o, label = "max a1")
```



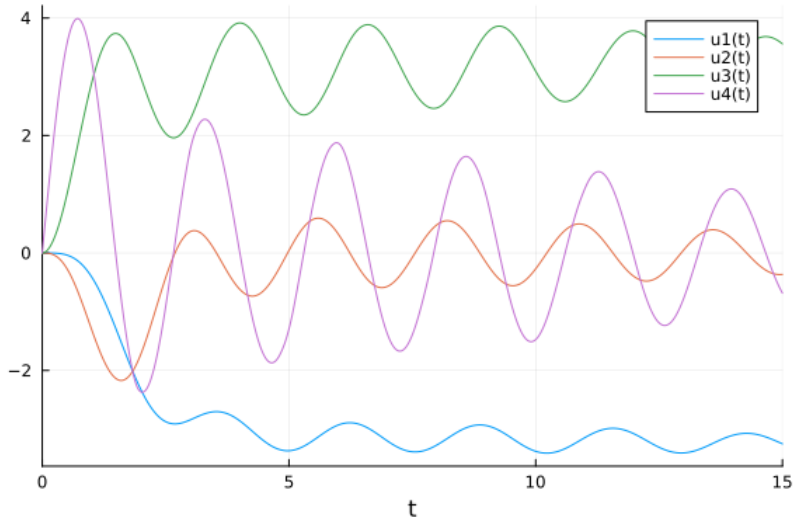
---

04 Mehanički Rotacioni Sistemi

```julia
# Primer 1
using Plots, DifferentialEquations

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c, k1, k2, R, g = p
    f = sin(t)
    J = 1 / 2 * m1 * R ^ 2

    dx[1] = x[2]
    dx[2] = (1 / J) * (k2 * (x[3] - R * x[1]) * R - k1 * x[1] - c * x[2])
    dx[3] = x[4]
    dx[4] = (1 / m2) * (f + m2 * g - k2 * (x[3] - R * x[1]))
end

# main.jl
t = (0.0, 20.0)
p = (10.0, 5.0, 10.0, 15.0, 10.0, 1.0, 9.81)
x0 = [0.0, 0.0, 2.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

θ = [x[1] for x in rešenje.u]
ω = [x[2] for x in rešenje.u]
α = diff(ω) ./ diff(rešenje.t)

plot(rešenje.t, [θ, ω], lw = 2, label = ["θ(t)" "ω(t)"])
plot!(rešenje.t[1:end-1], α, lw = 2, label = "α(t)")
```



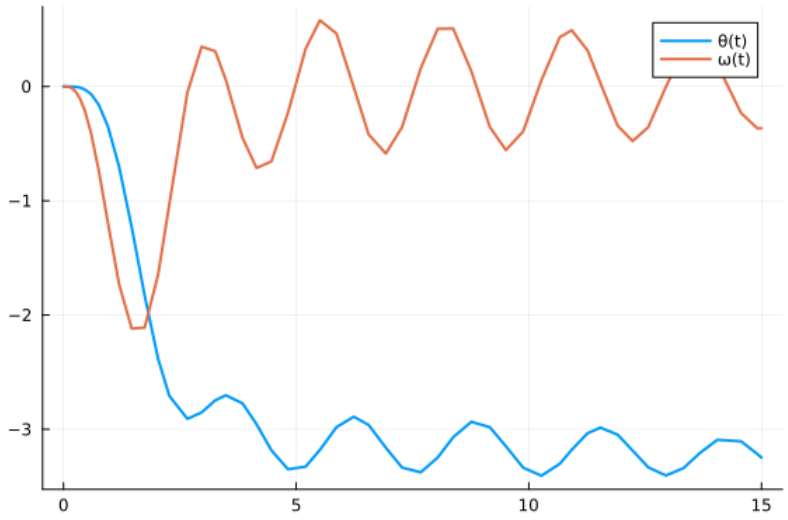
```julia
# Primer 2
using Plots, DifferentialEquations

function signal(t)
    tp = rem.(t, 3)

    y= (-1 / 3 * t .+ 5) .* (tp .<= 2)
end

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c, k1, k2, R, g = p
    J = 1 / 2 * m1 * R ^ 2
    f = signal(t)

    dx[1] = x[2]
    dx[2] = (-1 / J) * (k2 * (x[3] + R * x[1]) * R + c * R ^ 2 * x[2])
    dx[3] = x[4]
    dx[4] = (1 / m2) * (m2 * g - f - k1 * x[3] - k2 * (x[3] + R * x[1]))
end

# main.jl
t = (0.0, 15.0)
p = (10.0, 5.0, 10.0, 15.0, 10.0, 1.0, 9.81)
x0 = [0.0, 0.0, 0.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```



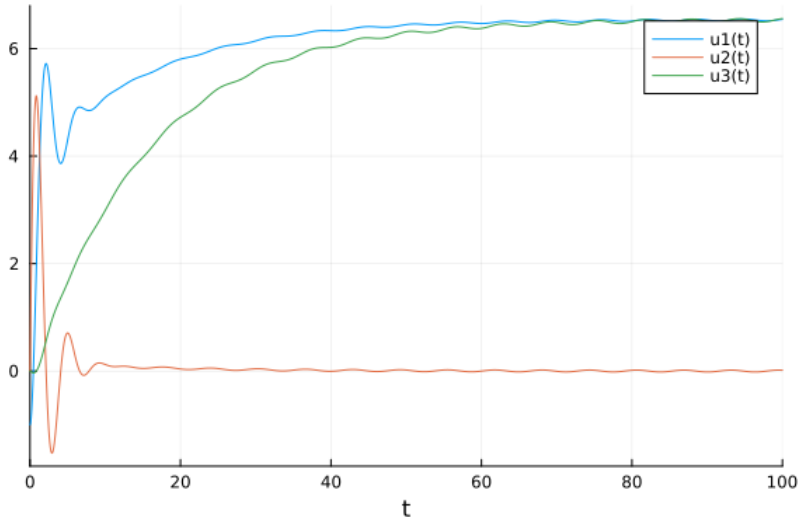

```julia
# pod d)
x = [x[3] for x in rešenje.u]
predjeni_put = sum(abs.(diff(x)))
println("Predjeni put tela mase m2 iznosi: ", predjeni_put)
```

```
Predjeni put tela mase m2 iznosi: 17.713435303740923
```

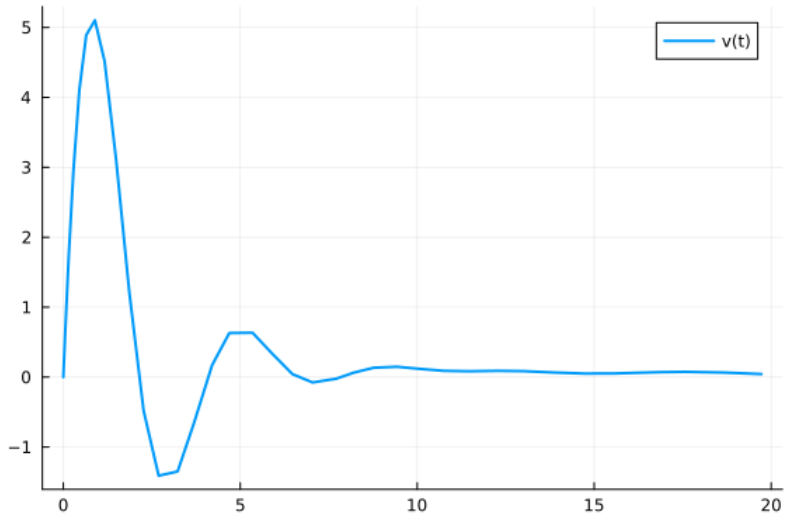
```julia
# pod e)
θ = [x[1] for x in rešenje.u]
ω = [x[2] for x in rešenje.u]

plot(rešenje.t, [θ, ω], lw = 2, label = ["θ(t)" "ω(t)"])
```



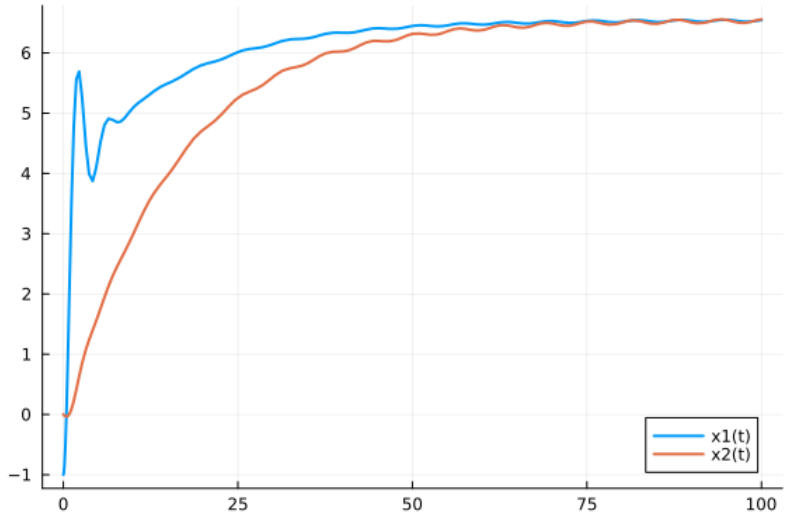
```julia
# Primer 3
using Plots, DifferentialEquations

function diferencijalneJednačine!(dx, x, p, t)
    m, c1, c2, k1, k2, g = p
    f = sin(t)

    dx[1] = x[2]
    dx[2] = (-1 / m) * (c1 * x[2] + k1 * x[1] + k2 * (x[1] - x[3]) - m * g)
    dx[3] = (k2 * (x[1] - x[3]) - 3 * f) / (9 * c2)
end

# main.jl
x0 = [-1.0, 0.0, 0.0]
t = (0.0, 100.0)
p = (10.0, 10.0, 10.0, 15.0, 10.0, 9.81)

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```




```julia
# pod d) -> promena brzina tela mase m prvih 20s
v = [x[2] for x in rešenje.u]
plot(rešenje.t[rešenje.t .<= 20], v[rešenje.t .<= 20], label = "v(t)", lw = 2)
```

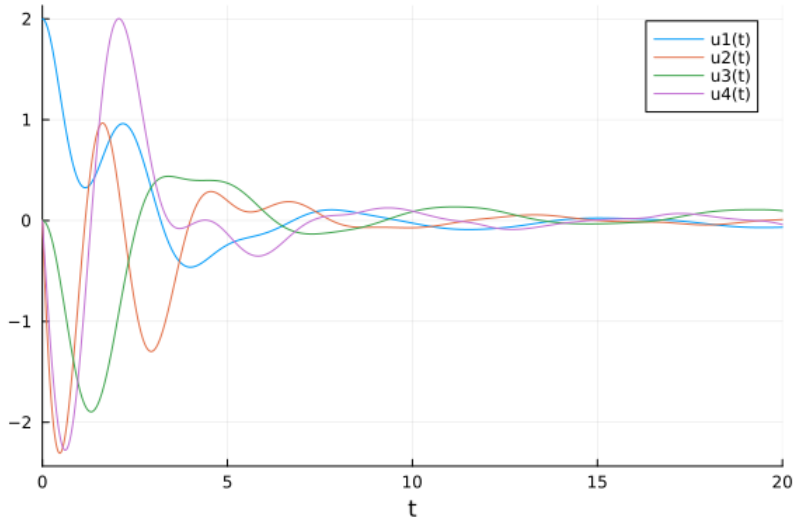



```
# pod e) -> promena pozicije tela mase m i levog kraja poluge
x1 = [x[1] for x in rešenje.u]
x2 = [x[3] for x in rešenje.u]

plot(rešenje.t, [x1, x2], legend=:bottomright, label = ["x1(t)" "x2(t)"], lw = 2, yticks = -1:6)
```

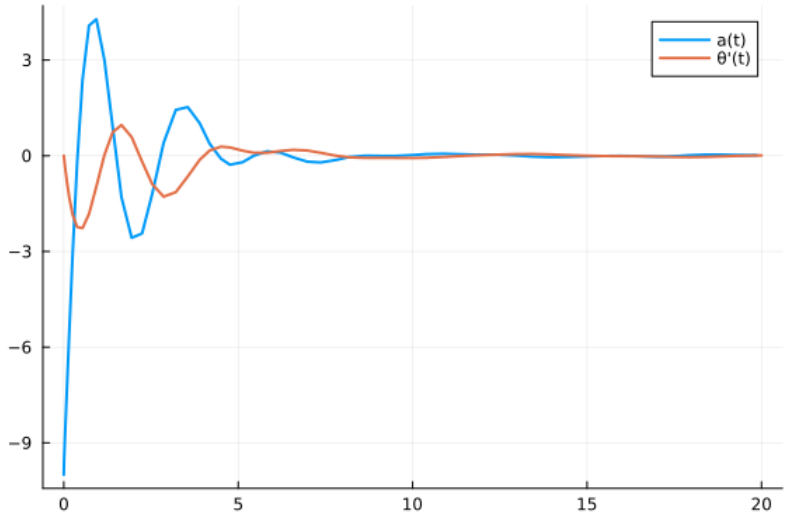


```
# Zadatak 1
using LinearAlgebra, Plots, DifferentialEquations

function signal(t)
    tp = rem.(t, 8)

    y1 = 0.5
    y2 = sin.(2 * pi / 8 * t) .* (tp .<= 4)

    y = min.(y1, y2)
end

# grafik kao iz postavke zadatka
# t = 0:0.001:20
# plot(t, signal(t), xticks = 0:20, lw = 2)
# plot!(t, (sin.(1 / 4 * pi * (rem.(t, 8))) .* (rem.(t, 8) .< 4)), line =:dash)

function diferencijalneJednačine!(dx, x, p, t)
    m1, m2, c, k1, k2, R = p
    f = signal(t)
    J = 1 / 2 * m1 * R ^ 2

    dx[1] = x[2]
    dx[2] = (-1 / J) * (c * R ^ 2 * x[2] + k1 * x[1] + k2 * (R * x[1] + x[3]) * R)
    dx[3] = x[4]
    dx[4] = (-1 / m2) * (k2 * (R * x[1] + x[3]) - f)
end
```

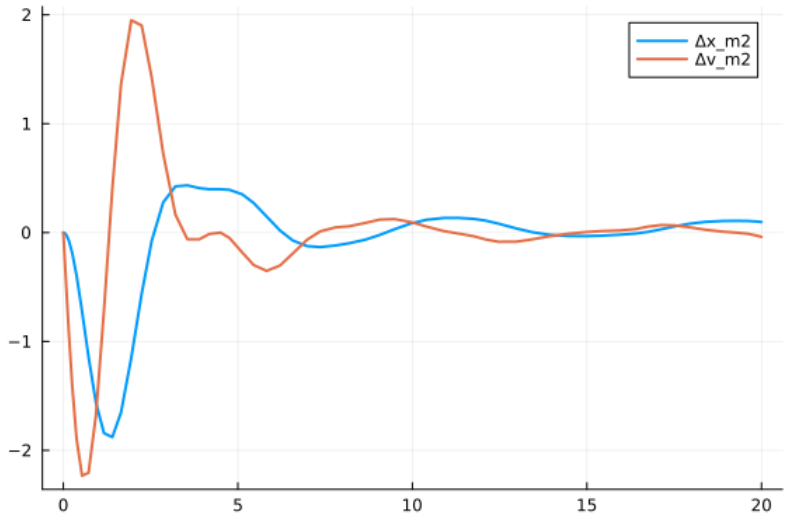
```julia
# main.jl
t = (0.0, 20.0)
p = (10.0, 5.0, 10.0, 10.0, 15.0, 1.0)
x0 = [2.0, 0.0, 0.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```



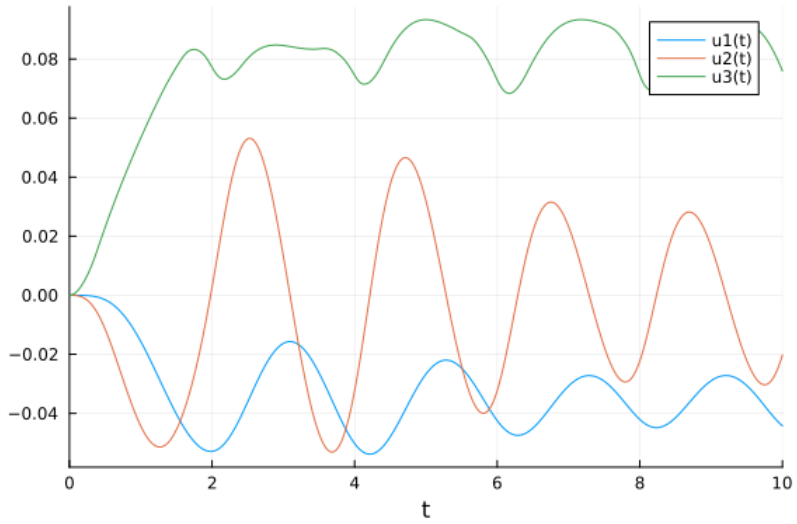

```julia
# pod d)
# odrediti promenu ugaonog ubrzanja diska u vremenu, i prikazati na istom grafiku
# promenu ugaone brzine i ubrzanja
x2 = [x[2] for x in rešenje.u]
ugaono_ubrzanje = diff(x2) ./ diff(rešenje.t)
# iscrtati x2 (promena ugaone brzine)

plot(rešenje.t[1:end-1], ugaono_ubrzanje, lw = 2, label = "a(t)")
plot!(rešenje.t, x2, lw = 2, label = "θ'(t)")
```



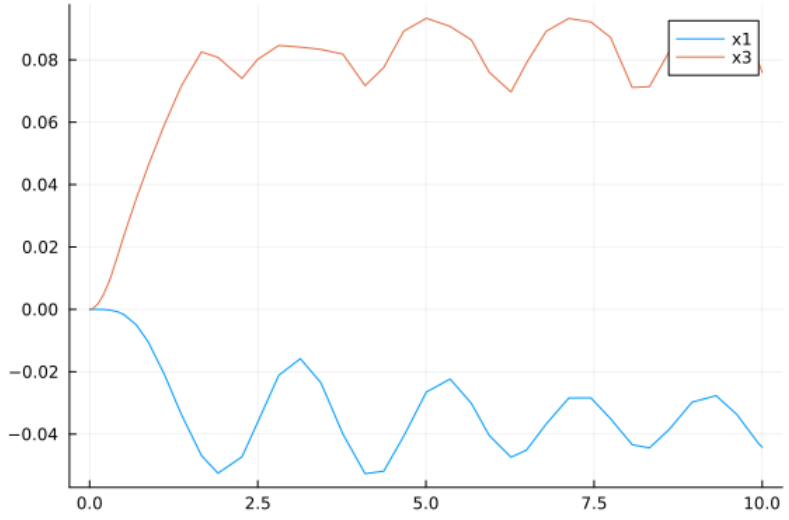

```julia
# pod e)
x3 = [x[3] for x in rešenje.u]
x4 = [x[4] for x in rešenje.u]

plot(rešenje.t, [x3, x4], lw = 2, label = ["Δx_m2" "Δv_m2"])
```



```julia
# Zadatak 2
using LinearAlgebra, Plots, DifferentialEquations

function signal(t)
    tp = rem.(t, 2)

    y_prava1 = (2 * tp) .* (tp .< 1)
    y_prava2 = (-2 * tp .+ 4) .* (tp .>= 1)

    y1 = y_prava1 .+ y_prava2

    y_kruznica = sqrt.(1 .- ((tp .- 1) .^ 2))

    y = min.(y1, y_kruznica)
end

function diferencijalneJednačine!(dx, x, p, t)
    m, c1, c2, k1, k2, R = p
    f = signal(t)
    J = 1 / 2 * m * R ^ 2

    dx[1] = x[2]
    dx[2] = (-1 / J) * (c1 * R ^ 2 * x[1] + k1 * x[1] + k2 * (R * x[1] + x[3]) * R)
    dx[3] = (-1 / c2) * (k2 * (R * x[1] + x[3]) - f)
end

# main.jl
t = (0.0, 10.0)
p = (10.0, 10.0, 8.0, 10.0, 15.0, 1.0)
x0 = [0.0, 0.0, 0.0]

problem = ODEProblem(diferencijalneJednačine!, x0, t, p)
rešenje = solve(problem)

plot(rešenje)
```




```
# pod d)
x1 = [x[1] for x in rešenje.u]
x3 = [x[3] for x in rešenje.u]

plot(rešenje.t, [x1, x3], label = ["x1" "x3"])
```




```
# pod e
x2 = [x[2] for x in rešenje.u]
ugaono_ubrzanje_m = diff(x2) ./ diff(rešenje.t)

~, idx1 = findmax(abs.(ugaono_ubrzanje_m))

plot(rešenje.t[1:end-1], ugaono_ubrzanje_m, label = "ugaono_ubrzanje_m", yticks = -1:0.02:1)
plot!([rešenje.t[idx1]], [ugaono_ubrzanje_m[idx1]], markershape = :o, label = "max ugaono_ubrzanje_m")
```