Numeričko rešavanje običnih diferencijalnih jednačina upotrebom paketa *DifferentialEquations*

Nedeljko Stojaković
Marko Pejić

Jul, 2021.

Cilj ovog materijala je upoznavanje sa funkcijama iz paketa *DifferentialEquations* koje implementiraju algoritme za numeričko rešavanje običnih diferencijalnih jednačina. Paket sadrži funkcije za rešavanje i drugačijih tipova problema (npr. parcijalne i stohastičke diferencijalne jednčine), ali ćemo se mi za potrebe ovog predmeta zadržati samo na ovom tipu problema. Kroz ovaj materijal neće se ulaziti u suštinu samih algoritama, već ćemo se zadržati na njihovoj upotrebi za određene probleme.

1. Uvod

Paket *DifferentialEquations* je najpoznatiji i najčešće korišćen za rešavanje problema opisanih diferencijalnim jednačinama (običnim, parcijanlnim, stohastičkim itd.), diskretnim jednačinama i tako dalje. Za potrebe predmeta zadržaćemo se na rešavanju običnih diferencijalnih jednačina.

Za definisanu diferencijalnu jednačinu, rešenje se može dobiti integracijom. Ovakvo rešenje zasniva se na izračunavanju izvoda u određenoj tački, tj. na aproksimaciji određenom funkcijom i izračunavanju vrednosti u sledećoj tački.

Paket *DifferentialEquations* sadrži više imlementiranih algoritama kojima se određuju rešenja diferencijalnih jednačina. Postoji unapred definisana procedura za korišćenje funkcija, sa kojom ćemo se upoznati kroz primere.

Algoritmi za numeričko rešavanje običnih diferencijalnih jednačina rešavaju diferencijalne jednačine prvog reda, uključujući i sisteme običnih diferencijalnih jednačina prvog reda. To znači, da ćemo naše modele, koji će često biti opisanim diferencijalnim jednačinam višeg reda, prilagođavati algoritmima.

Za početak, krenućemo sa jednim jednostavnim primerom obične diferencijalne jednačine prvog reda. Recimo, imamo sledeću diferencijalnu jednačinu:

$$\frac{du}{dt} = f(u, p, t) = sin(t)$$

Potrebno je odrediti rešenje diferencijalne jednačine u vremenskom intervalu $[0, 10]$ sekundi, ako je početni uslov $u(0) = 0$. Dobijeno rešenje prikazati grafički.

Postupak rešavanja, koji uključuje pisanje koda, možemo da podelimo u nekoliko koraka. Kada imamo opisan problem kao diferencijalnu jednačinu prvog reda ili kao sistem diferencijalnih jednačina prvog reda, na prvom mestu, posle dodavanja potrebnih paketa, je pisanje funkcije koja će da sadrži model, tj. diferencijalnu(e) jednačinu(e). U sledećem koraku, definišemo sve potrebne i poznate informacije i parametre, kao što su početni uslovi, vremenski interval nad kojim se traži rešenje, vrednosti ulaza i parametara ukoliko postoje. Nakon toga, pozivamo prvu funkciju iz paketa *DifferentialEquations*, a to je funkcija *ODEProblem* čiji je zadatak da kreira objekat koji će u sebi da nosi sve poznate informacije koje imamo o modelu. Dalje se taj dobijeni objekat prosleđuje drugoj funkciji iz paketa *DifferentialEquations*, funkciji *solve*. U ovoj funkciji se dešava pozivanje određenog algoritma za traženje rešenja.

Za ovaj primer mi već znamo analitičko rešenje diferencijalne jednačine:

$$u = -cos(t) + 1$$

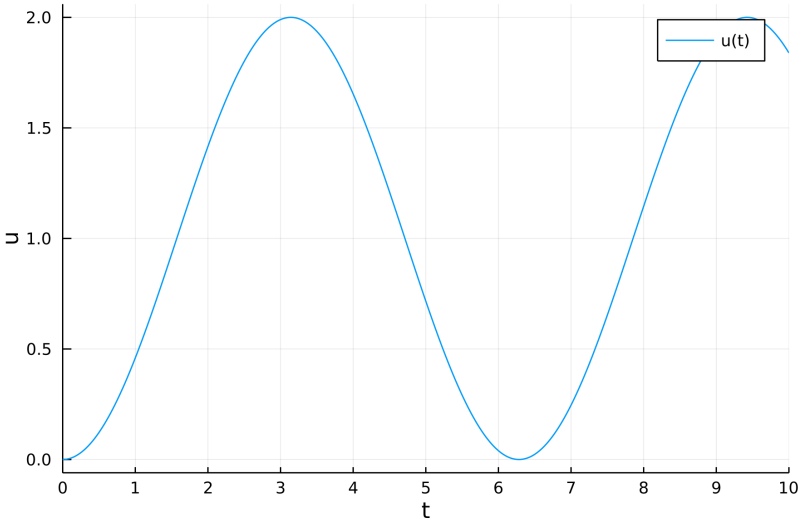
Julia kod koji nam daje numeričko rešenje diferencijalne jednačine iz primera dat je u nastavku. Potrebno je potvrditi da li dobijeno rešenje odgovara očekivanom analitičkom rešenju, što možemo uraditi vizuelno tako što ćemo rešenje prikazati na grafiku pomoću paketa *Plots*.

```
# Prvi korak - dodavanje potrebnih paketa.
using DifferentialEquations
using Plots

# Drugi korak - opis problema pomocu funkcije
f(u, p, t) = sin(t)

# Treci korak - definisanje svih poznatih informacija
u0 = 0.0
tspan = (0.0, 10.0)  # potrebno je da granice opsega budu racionalni brojevi!

# Cetvrti korak - pozivanje funkcije ODEProblem
prob = ODEProblem(f, u0, tspan)

# Peti korak - odredjivanje resenja diferencijalne jednacine
sol = solve(prob)

# Sesti korak - prikaz resenja i analiza
plot(sol)
```

Funkcija *solve* vraća rešenje diferencijalne jednačine za vremenski opseg definisan promenljivom koju smo u ovom primeru nazvali *tspan*, a to je od 0 do 10 sekundi. Algoritam[1] uzima određen broj vrednosti sa tog opsega i za te vrednosti računa rešenja diferencijalne jednačine. Povratna vrednost funkcije *solve* je struktura koju smo nazvali *sol*, koja u sebi ima dva niza t i u. To znači da naredbama *sol.t* i *sol.u* pristupamo vektorima koji kao elemente sadrže vremenske trenutke za koje su nađena rešenja diferencijalnih jednačina, kao i dobijena rešenja, respektivno. Rešenje diferencijalne jednačine prikazano je na slici 1.

[1] Detaljnije o algoritmima će biti više reči u nastavku predmeta.

Sledećim primerom ilustrovaćemo postupak rešavanja sistema običnih diferen-



Slika 1: Rešenje diferencijalne jednačine $\dot{u} = sin(t)$

cijalnih jednačina prvog reda. Data je diferencijalna jednačina Van der Pol-ovog oscilatora drugog reda:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0 \tag{1}$$

Potrebno je odrediti rešenja diferencijalne jednačine za prvih 30 sekundi, ako su svi početni uslovi 0.25.[2] Grafički prikazati dobijena rešenja, ako je $\mu = 1$.[3]

Da bismo mogli primeniti funkcije iz paketa *DiffernetialEquations*, potrebno je izvršiti transformaciju diferencijalne jednačine višeg reda u sistem diferencijalnih jednačina prvog reda. Za ovo postoji više načina, ali ćemo mi koristiti jedan opšte prihvaćen princip koji podrazumeva uvođenje smena koje se nazivaju promenljive stanja. Broj promenljivih stanja zavisi od reda diferencijalne jednačine, tj. od najvišeg izvoda nezavisne promenljive. Pošto u ovom primeru imamo diferencijalnu jednačinu drugog reda, to znači da ćemo imati dve smene. Definišemo ih na sledeći način:

$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} \end{aligned} \tag{2}$$

Dalje, tražimo izvode promenljivih stanja x_1 i x_2:

$$\begin{aligned} \dot{x}_1 &= \dot{x} \\ \dot{x}_2 &= \ddot{x} \end{aligned} \tag{3}$$

[2] Broj početnih uslova zavisi od reda diferencijalne jednačine, tj. od reda sistema.

[3] Izmeniti postojeći primer tako da se grafički prikazuje rešenje za različite vrednosti μ istovremeno. Koristiti vrednosti $2, 3, 5, 9$.

Na osnovu uvedenih smena (2) i početne diferencijalne jednačine (1), konačno dobijamo sistem običnih diferencijalnih jednačina:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \mu(1 - x_1^2)x_2 - x_1 \end{aligned} \tag{4}$$

U nastavku ostaje pisanje odgovarajućeg koda kojim ćemo odrediti rešenja sistema običnih diferencijalnih jednčina (4).

```
function van_der_pol!(dx, x, p, t)
    μ = p

    dx[1] = x[2]
    dx[2] = μ*(1 - x[1]^2)*x[2] - x[1]
end

x0 = [0.25, 0.25] # pocetni uslovi
interval = (0.0, 30.0) # posmatrani vremenski period

μ = 1.0
parametri = μ

prob = ODEProblem(van_der_pol!, x0, interval, parametri)
sol = solve(prob)

plot(sol, xlabel="sol.t", ylabel="sol.u", label=["x1" "x2"])
```

Pošto u ovom primeru imamo sistem diferencijalnih jednačina prvog reda koji se sastoji od dve jednačine, funkcija koja opisuje model[4] je malo drugačija u odnosu na prvi primer. Neophodni ulazni parametri koji se navode u funkciji imaju sledeće značenje:

- *dx* - **vektor** koji sadrži sistem diferencijalnih jednačina.
- *x* - **vektor** promenljivih stanja.
- *param* - **torka** parametara modela.
- *t* - **skalar** koji predstavlja vremenski trenutak za koji se traži rešenje sistema diferencijalnih jednačina.

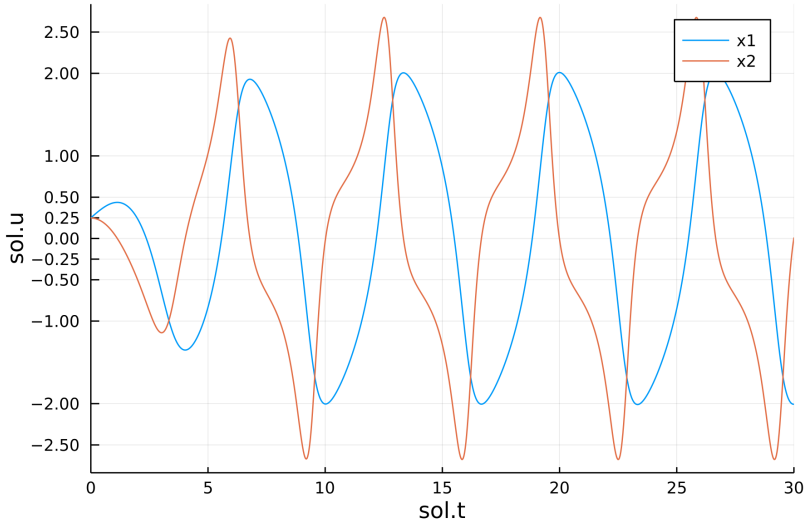
[4] Funkcija van_der_pol!.

Takođe, vektor *u* u strukturi *sol*[5] sada, kao elemente sadrži *n* vektora[6], gde svaki vektor ima po dva elementa koji predstavljaju rešenja diferencijalnih jednačina, tj. x_1 i x_2 za dati vremenski trenutak. Data je ilustracija strukture *sol* za ovaj slučaj.

[5] Vektor *sol.u*

[6] Broj tih vektora zavisi od broja vremenskih trenutaka za koje se traže rešenja diferencijalnih jednačina.

$$\begin{array}{c} \boldsymbol{sol} \\ \begin{bmatrix} \boldsymbol{sol.t} & \boldsymbol{sol.u} \\ \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} x_1(t_0) & x_2(t_0) \end{bmatrix} \\ \begin{bmatrix} x_1(t_1) & x_2(t_1) \end{bmatrix} \\ \begin{bmatrix} x_1(t_2) & x_2(t_2) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} x_1(t_n) & x_2(t_n) \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{array}$$



Slika 2: Rešenje diferencijalne jednačine Van Der Pol-ovog oscilatora.

Ukoliko bi hteli da iz strukture *sol* izdvojimo rešenja samo za prvu promenljivu stanja x_1, potrebno je iterirati kroz sve elemente vektora *sol.u* i u svakom tom vektoru selektovati prvi element. Za takav pristup koristićemo *array comprehension* na sledeći način:

```
x1 = [x[1] for x in sol.u]
```

Ova naredba nam daje vektor:

$$x_1 = \begin{bmatrix} x_1(t_0) \\ x_1(t_2) \\ x_1(t_3) \\ \vdots \\ x_1(t_n) \end{bmatrix}$$

Na isti način možemo izdvojiti sva ostala rešenja i po potrebi izvršiti dodatnu analizu i proračune.

2. Primeri sa rešenjima

Primer 1. Data je diferencijalna jednačina prvog reda gde postoje parametri modela:

$$\dot{y} = \lambda e^{-\alpha t} y$$
$$y(0) = y_0 = 1$$

Analitičko rešenje je poznato i iznosi:

$$y(t) = y_0 e^{\frac{\lambda}{\alpha}(1-e^{-\alpha t})}$$

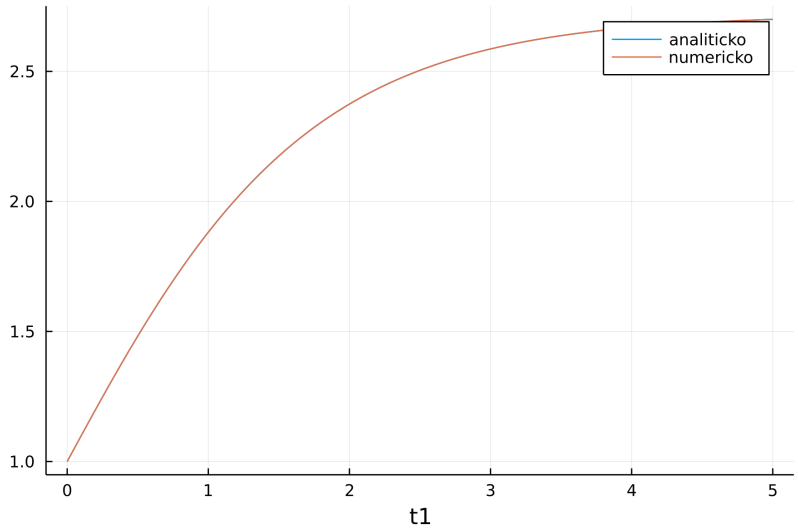
Odrediti numeričko rešenje diferencijalne jednačine za prvih 5 sekundi i uporediti ga sa analitičkim rešenjem, ako je $\alpha = 1$ i $\lambda = 1$.

Rešenje:[1]

[1] Prilikom poređenja analitičkog i numeričkog rešenja moramo koristiti iste vremenske trenutke.

```
function dif_jedn!(y, param, t)
    α, λ = param
    dy = λ * exp(-α*t) * y
    return dy
end

function analiticko_resenje(t, y0, param)
    α, λ = param
    y = y0*exp.(λ/α*(1 .- exp.(-α*t)))
    return y
end

y0 = 1.0
parametri = (1.0, 1.0)
interval = (0.0, 5.0)
problem = ODEProblem(dif_jedn!, y0, interval, parametri)
num = solve(problem)  # resenje

t1 = 0:0.01:5
analiticko = analiticko_resenje(t1, y0, parametri)

numericko = num(t1) # interpolacija resenja novim vektorom t1

plot(t1, analiticko, xlabel="num.t", ylabel="num.u", label="analiticko")
plot!(t1, numericko, label="numericko")
```

Radi poređenja analitičkog i numeričkog rešenja uveden je novi vremenski vektor $t1$ sa kojim su izračunata rešenja analitičkog rešenja, ali je izvršena interpolacija rešenja diferencijalnih jednačina sa istim vektorom.

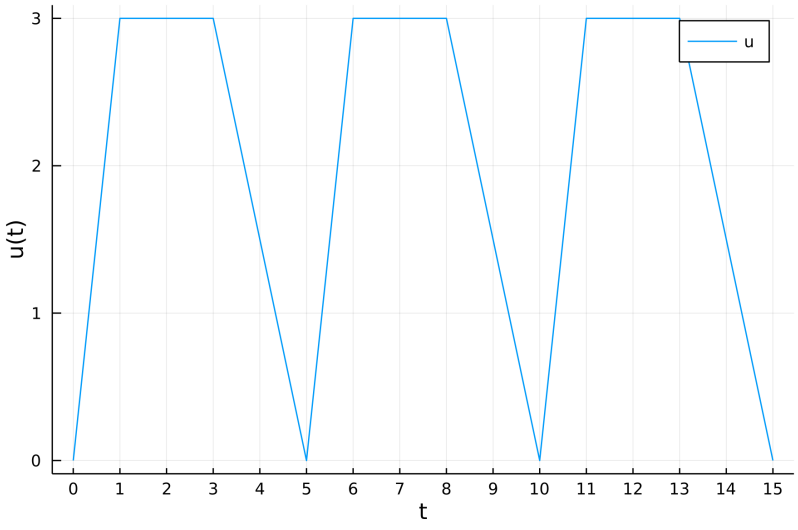


Primer 2. Date su dve diferencijalne jednačine koje su međusobno zavisne:

$$\dot{x} - A\dot{y} = u(t)$$
$$\ddot{y} - y + Bx = 0$$

$$x(0) = -1, \quad y(0) = 1, \quad \dot{y}(0) = 0.5$$

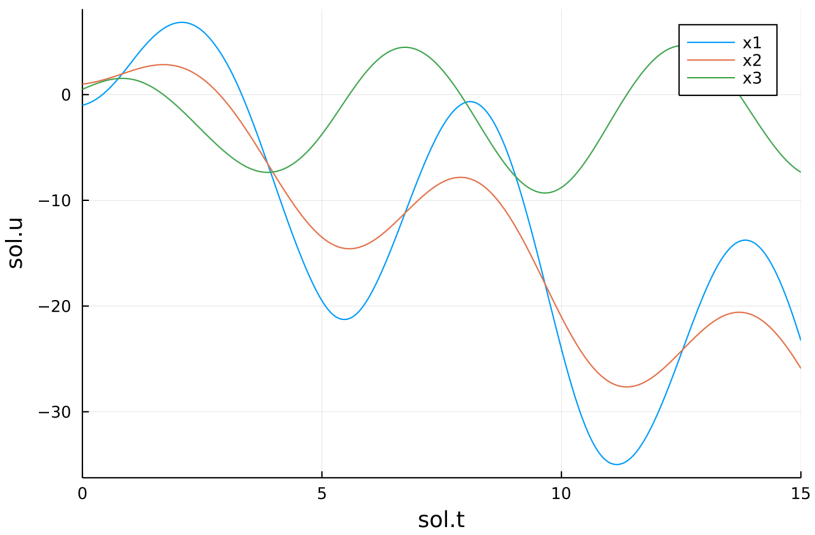
Odrediti rešenja diferencijalnih jednačina za prvih 15 sekundi, ako je $A = 2$, $B = 1$, a ulazni signal $u(t)$ prikazan na grafiku ispod. Grafički prikazati dobijena rešenja.



Rešenje:

$$\begin{aligned} x_1 = x \quad &\Rightarrow \quad \dot{x}_1 = \dot{x} \quad \rightarrow \quad \dot{x}_1 = u(t) + Ax_3 \\ x_2 = y \quad &\Rightarrow \quad \dot{x}_2 = \dot{y} \quad \rightarrow \quad \dot{x}_2 = x_3 \\ x_3 = \dot{y} \quad &\Rightarrow \quad \dot{x}_3 = \ddot{y} \quad \rightarrow \quad \dot{x}_3 = -Bx_1 + x_2 \end{aligned}$$

```
function dif_jedn!(dx, x, param, t)
    A, B = param

    dx[1] = u(t) + A*x[3]
    dx[2] = x[3]
    dx[3] = -B*x[1] + x[2]
end

function u(t)
    tp = rem(t, 5)
    u = 3*tp * (tp < 1) + 3 * ((tp >= 1) & (tp < 3)) +
        (-tp * 3/2 + 15/2) * ((tp >= 3) & (tp <= 5))
end

x0y0 = [-1.0, 1.0, 0.5]
interval = (0.0, 15.0)

A, B = (2.0, 1.0)
parametri = (A, B)

prob = ODEProblem(dif_jedn!, x0y0, interval, parametri)
sol = solve(prob)

plot(sol, xlabel="sol.t", ylabel="sol.u", label = ["x1" "x2" "x3"])
```



3. Zadaci za vežbu

Zadatak 1. Odrediti rešenja diferencijalnih jednačina za početnih 30 sekundi, ako je $\alpha = 10$, $\beta = 27$ i $\gamma = \frac{8}{3}$, za početne uslove:

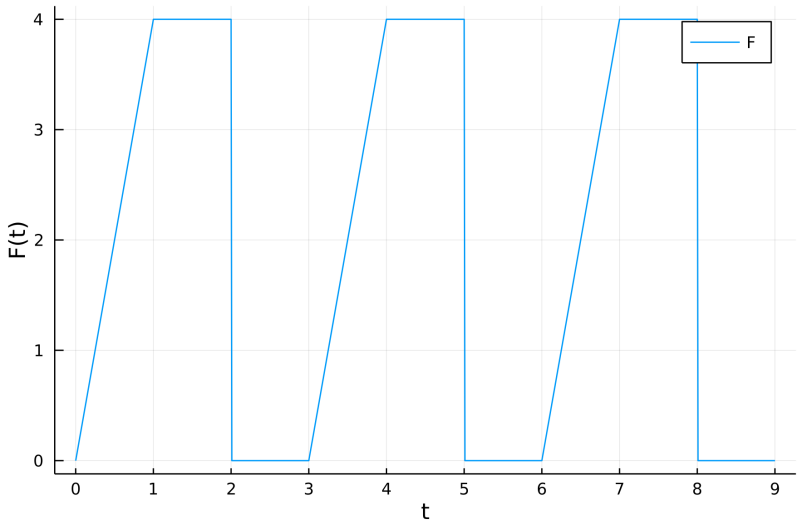
1. $x(0) = 1$, $y(0) = 0$, $z(0) = 0$.
2. $x(0) = 1$, $y(0) = 0.01$, $z(0) = 0.01$.

Dobijena rešenja za oba slučaja prikazati na jednom grafiku.

$$\begin{aligned} \dot{x} &= \alpha(y - x) \\ \dot{y} &= x(\beta - z) - y \\ \dot{z} &= xy - \gamma z \end{aligned}$$

Zadatak 2. Odrediti rešenja diferencijalnih jednačina, ako je $A = 12$, $B = 8$, $C = 4$, dok je ulazni signal $F(t)$ prikazan na grafiku ispod.

$$\begin{aligned} 2\ddot{y} - C(\dot{x} - \dot{y}) + Ay - B(x - y) - F(t) &= 0 \\ \ddot{x} + 3\dot{x} + C(\dot{x} - \dot{y}) + B(x - y) &= 0 \end{aligned}$$



Literatura

- Aleksandar Erdeljan, Darko Čapko: Modelovanje i simulacija sistema - sa primerima; FTN, Novi Sad, 2015.
- *DifferentialEquations* dokumentacija https://diffeq.sciml.ai/stable/
- *Julia* programski jezik (sajt) https://julialang.org/
- *Think Julia* (*online* knjiga) https://benlauwens.github.io/ThinkJulia.jl/latest/book.html.