# 101: Model vožnje skozi semafor

Peter Rupnik

10. oktober 2018

## 1 Prepis problema v brezdimenzijsko obliko

Za brezdimenzijsko časovno spremenljivko vpeljem

$$x = \frac{t}{t_0}, \tag{1}$$

kjer je $t_0$ čas do prižiga zelene luči. Za brezdimenzijsko hitrost sem si izbral

$$y = v \cdot \frac{t_0}{L} = \frac{v}{\langle v \rangle}. \tag{2}$$

S temi spremenljivkami zapišem akcijo, ki jo bomo minimizirali:

$$\mathcal{S} = \int_0^1 \mathcal{L} \mathrm{d}x = \int_0^1 (y')^2 - \lambda y \, \mathrm{d}x, \tag{3}$$

upoštevam pa tudi vez, da naj v času $t_0$ avto prevozi ravno $L$:

$$\int_0^{t_0} v \, \mathrm{d}t = \int_0^1 \frac{L}{t_0} y \cdot t_0 \cdot \mathrm{d}x = L \tag{4}$$

$$\int_0^1 y \, \mathrm{d}x = 1. \tag{5}$$

Naslednji korak je Euler–Langrangeva enačba:

$$\frac{\partial \mathcal{L}}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathcal{L}}{\partial} y' = 0, \tag{6}$$

vanjo vstavimo pripravljeno $\mathcal{L}$ in dobimo

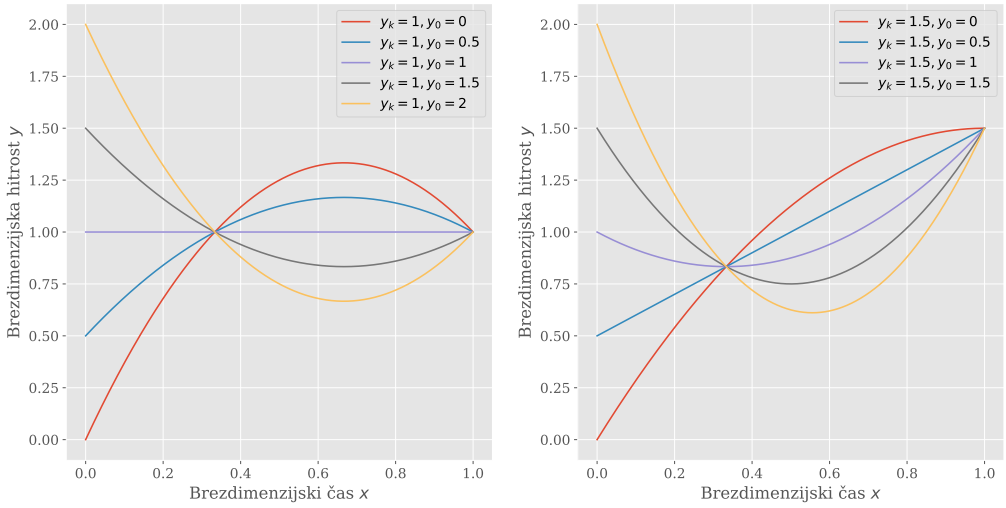
$$-\lambda - 2 \, y'' = 0, \tag{7}$$

iz česar po dvakratni integraciji dobimo hitrost v odvisnosti od časa:

$$y(x) = Ax^2 + Bx + C. \tag{8}$$

Konstante $A, B, C$ določimo iz začetnih in končnih pogojev, upoštevamo tudi, da mora biti ploščina pod grafom enaka 1.

$$\int_0^1 y \, \mathrm{d}x = 1 \implies \frac{A}{3} + \frac{B}{2} + C = 1. \tag{9}$$




Slika 1: Splošna družina rešitev z dvema prostima parametroma: začetno hitrostjo $y_0$ in končno hitrostjo $y_k$. Ploščina pod vsemi krivuljami znaša vselej 1, kot to zahteva (9). Levi in desni graf prikazujeta različni končni hitrosti, na vsakem grafu sem prikazal več začetnih hitrosti.

Če poznamo začetno in končno hitrost $y_0$ in $y_k$, izgleda hitrostni profil tako:

$$y(x) = (3y_k + 3y_0 - 6)(x^2 - \frac{2x}{3}) + 2(1 - y_0)x + y_0$$

(10)

## 2 Optimalna vožnja pri poljubni končni hitrosti

Optimalno vožnjo pri poljubni končni hitrosti vpeljemo z zahtevo $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y'} = 0$.

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y'} = 2y' = 2\frac{\mathrm{d}y}{\mathrm{d}x} = 0 \tag{11}$$

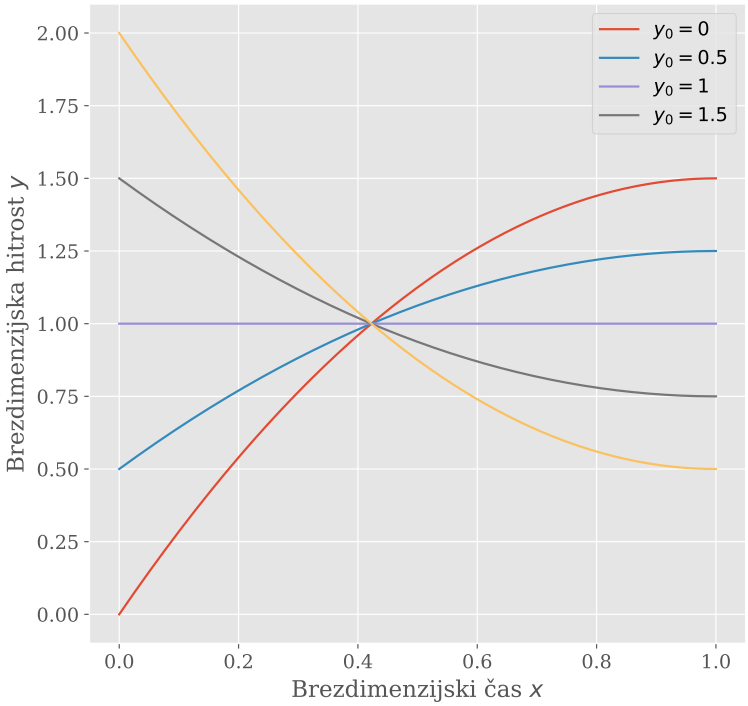
$y(x)$ za primer, ko nimamo predpisane končne hitrosti, izgleda takole:

$$y(x) = A \cdot (x^2 - \frac{2x}{3}) + 2(1 - y_0)x + y_0, \tag{12}$$

iz česar dobimo za $A = -\frac{3}{2}(1 - y_0)$. Če vstavimo v (12), dobimo za $y(x)$:

$$y(x) = \frac{3}{2}x\,(1 - y_0)\,(2 - x) + y_0. \tag{13}$$

Začetna hitrost $y_0$ nam določa točen potek hitrosti.




Slika 2: Pri poljubni končni hitrosti potek hitrosti izgleda tako. Narisal sem več začetnih hitrosti. Vidimo, da se naša zahteva $\mathrm{d}\mathcal{L}/\mathrm{d}y' = 0$ odraža z ničelnim odvodom hitrosti pri semaforju.

# 3 Višje potence

Ker želim, da se pospeševanje in zaviranje tretira enako (za varčno vožnjo minimiziramo oboje), bom raziskal sode višje potence. Tedaj Lagranžian zapišem tako:

$$\mathcal{L} = (y')^{2n} - \lambda y \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial \mathcal{L}}{\partial y'} = 0 \tag{15}$$

$$-\lambda - 2n\frac{\mathrm{d}}{\mathrm{d}x}y'^{2n-1} = 0 \tag{16}$$

$$-\lambda - 2n(2n-1)y'^{2n-2}y'' = 0 \tag{17}$$

Drugi člen v (17) polepšamo tako, da ga prepišemo kot

$$(2n-1)y'^{2n-2}y'' = \frac{\mathrm{d}}{\mathrm{d}x}\left(y'^{2n-1}\right), \tag{18}$$

nadalje izpeljava poteka tako:

$$\frac{\mathrm{d}}{\mathrm{d}x}2ny'^{2n-1} = -\lambda \tag{19}$$

$$2n\int \frac{\mathrm{d}}{\mathrm{d}x}y'^{2n-1}\,\mathrm{d}x = -\int \lambda\,\mathrm{d}x \tag{20}$$

$$2ny'^{2n-1} = -\lambda x + A \tag{21}$$

$$y'^{2n-1} = -\frac{\lambda x}{2n} + \frac{A}{2n} \tag{22}$$

$$y' = \left(\frac{A-\lambda x}{2n}\right)^{\frac{1}{2n-1}} \tag{23}$$

$$\tag{24}$$

Tu izkoristimo priložnost za uporabo druge točke (12) in z zahtevo $2y'(1) = 0$ poračunamo konstanto $A$:

$$y'(1) = 0 \implies A = \lambda \tag{25}$$

Še enkrat integriramo:

$$y(x) = \left(\frac{\lambda}{2n}\right)^{\frac{1}{2n-1}} \int (1-x)^{\frac{1}{2n-1}}\,\mathrm{d}x + B \tag{26}$$

$$y(x) = \left(\frac{\lambda}{2n}\right)^{\frac{1}{2n-1}} \frac{(1-x)^{\frac{1}{2n-1}+1}}{\frac{1}{2n-1}+1} + B \tag{27}$$

Uporabimo začetni pogoj $y(0) = y_0$ in dobimo še konstanto $B$:

$$B = y_0 - \frac{\left(\frac{\lambda}{2n}\right)^{\frac{1}{2n-1}}}{\frac{1}{2n-1}+1} \tag{28}$$

Če vse skupaj združimo, dobimo za potek hitrosti pri poljubni sodi potenci $2n$:

$$y(x) = y_0 + \left(\frac{\lambda}{2n}\right)^{\frac{1}{2n-1}} \frac{1}{\frac{1}{2n-1}+1}\left((1-x)^{\frac{1}{2n-1}+1} - 1\right) \tag{29}$$

$\lambda$ določimo z zahtevo (9):

$$\lambda = 2n \left[ \frac{(1-y_0)\left(\frac{1}{2n-1}+1\right)}{-\frac{(1-x)^{\frac{1}{2n-1}+2}}{\frac{1}{2n-1}+2} - 1} \right]^{2n-1} \tag{30}$$

# 4 Kvadratični člen v funkcionalu

V Lagranževo funkcijo dodamo člen, ki je sorazmeren kvadratu hitrosti:

$$\mathcal{L} = y'^2 + ay^2 - \lambda y. \tag{31}$$

To nam po vstavljanju v Euler–Lagrangevo enačbo (6) porodi diferencialno enačbo:

$$-\lambda + 2ay - 2y'' = 0 \tag{32}$$

$$y'' - ay = -\frac{\lambda}{2} \tag{33}$$

Wolfram|Alpha pravi, da tako diferencialno enačbo reši

$$y = Ae^{\sqrt{a}x} + Be^{-\sqrt{a}x} + \frac{\lambda}{2a} \tag{34}$$

Uvedem $b^2 = a$, da polepšam enačbo, privzamem torej, da je $a > 0$, kar lahko povežemo z eksponentnim naraščanjem in padanjem hitrosti, kar se zdi bolj pričakovano kot pa harmonska nihanja hitrosti, ki bi jih dobili za negativne $a$.

$$y = Ae^{bx} + Be^{-bx} + \frac{\lambda}{2b^2} \tag{35}$$

$$\tag{36}$$

Za robni pogoj spet uporabimo (12):

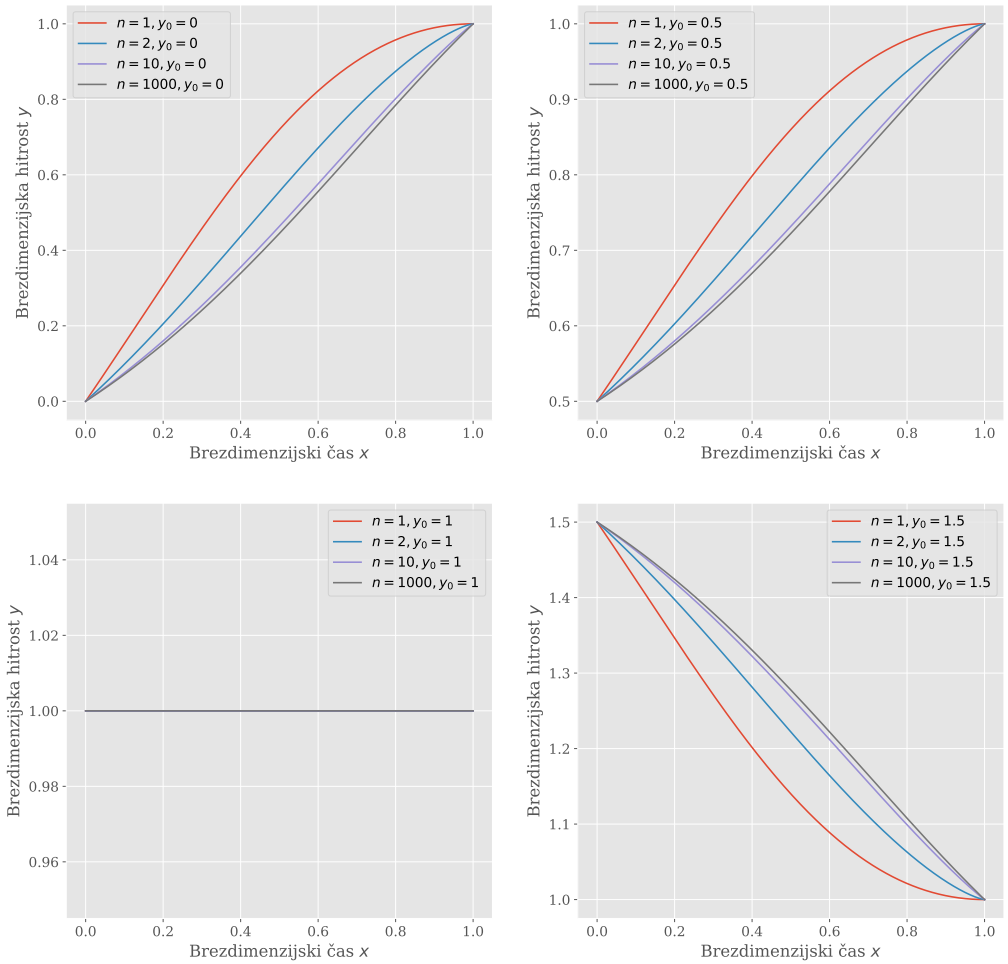
$$y'(1) = b\left(Ae^{bx} + Be^{-bx}\right) = 0 \tag{37}$$

$$B = Ae^{2b} \tag{38}$$

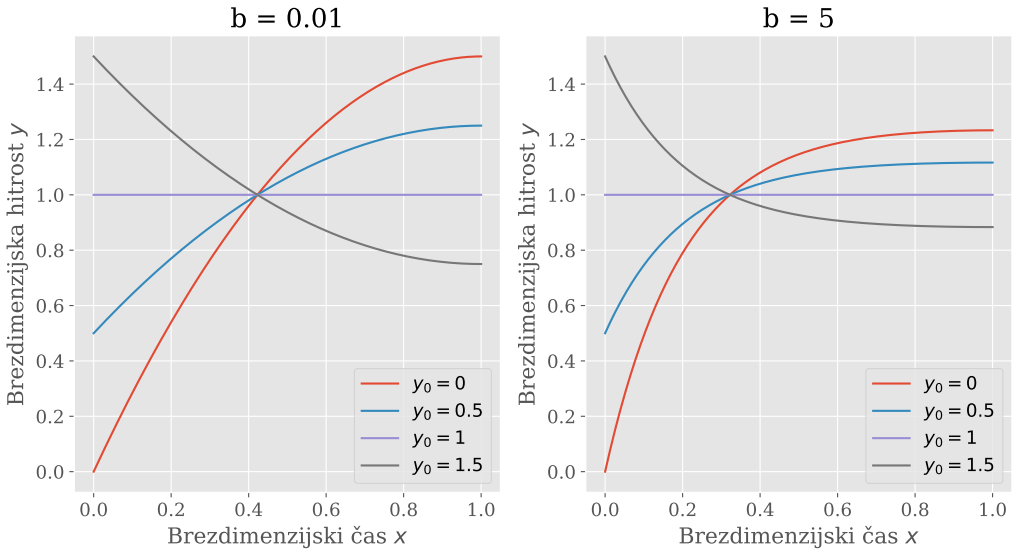
$$y = A\left(e^{bx} + e^{2b}e^{-bx}\right) + \frac{\lambda}{2b^2} \tag{39}$$

$$y = A\left(e^{bx} + e^{b(2-x)}\right) + \frac{\lambda}{2b^2} \tag{40}$$

$$y = 2Ae^{b}\cosh b(1-x) + \frac{\lambda}{2b^2} \tag{41}$$




Slika 3: Vpeljava višjih potenc v Lagranžian. Graf na matrični koordinati (2,1) prikazuje situacijo, ko je začetna hitrost enaka povprečni, zaradi česar seveda avtu ni potrebno ne pospeševati ne zavirati, zato se pri vseh eksponentih obnaša enako: avto vozi z enakomerno hitrostjo ves čas. Z višanjem eksponenta se poteki hitrosti zelo hitro bližajo premici skozi končno in začetno točko, ker višji eksponenti penalizirajo strmejše odvode, dobimo krivuljo s konstantnim naklonom, ki nikoli ne postane položnejša, ker bi potem morala drugje postati spet strmejša, kar pa bi zaradi visokih eksponentov povzročilo visoke vrednosti $\mathcal{L}$.




Slika 4: Posledice vpeljave kvadrata hitrosti v Lagranžian. Odvisno od faktorja dušenja $b$ se hitrost prej ali slej ustali pri različnih vrednostih. Kot smo že navajeni, je tudi tukaj najbolj dolgočasen primer z $y_0 = 1$, ker tudi tukaj nimamo nobene zanimive dinamike, avto le pluje z nespremenjeno hitrostjo proti semaforju.

Uporabimo še notacijo $y(0) = y_0$ in pogoj (9):

$$y_0 = 2Ae^b \cosh b + \frac{\lambda}{2b^2} \tag{42}$$

$$\frac{\lambda}{2b^2} = y_0 - 2Ae^b \cosh b \tag{43}$$

$$y(x) = 2Ae^b \left(\cosh b(1-x) - \cosh b\right) + y_0 \tag{44}$$

$$\int_0^1 y \, \mathrm{d}x = 1 \tag{45}$$

$$2Ae^b \left[\int_0^1 \cosh b(1-x)\mathrm{d}x - \cosh bx\big|_0^1\right] + y_0 x\big|_0^1 = 1 \tag{46}$$

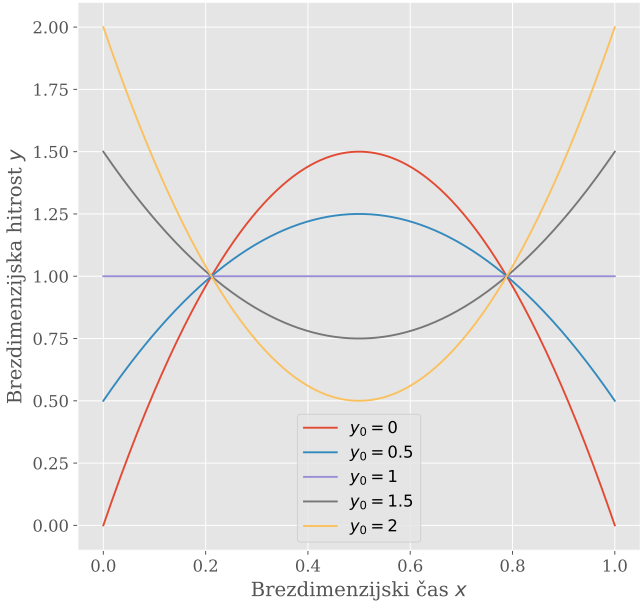
$$\int_0^1 \cosh b(1-x)\mathrm{d}x = \frac{\sinh b(x-1)}{b} \tag{47}$$

$$2Ae^b = \frac{1-y_0}{\frac{\sinh b}{b} - \cosh b} \tag{48}$$

$$y(x) = \frac{1-y_0}{\frac{\sinh b}{b} - \cosh b}\left(\cosh b(1-x) - \cosh b\right) + y_0 \tag{49}$$

# 5 Zaporedni semaforji

Uporabil bom začetno obliko Langranžiana, torej $\mathcal{L} = (y')^2 - \lambda y$, ki porodi že poznano rešitev $y = Ax^2 + Bx + C$. Če imamo več zaporednih semaforjev, pripeljemo na začetek




Slika 5: Nekaj krivulj poteka hitrosti za različne vrednosti $y_0$. Kot bi pričakovali, obstaja zrcalna ravnina pri $x = 0.5$. Vidimo tudi, da bi v primeru uporabe pogoja (12) izgubili še edini preostal prost parameter $y_0$, zveznost odvoda na robu bi pomenila enakomerno gibanje s povprečno hitrostjo $y = 1,\ v = \langle v \rangle$.

definicijskega območja $x$ z isto hitrostjo, kot ga zapustimo:

$$y(1) = y(0) = y_0 \tag{50}$$

Obenem še vedno velja pogoj normirane ploščine pod grafom (9). Iz teh dveh zvez lahko določimo družino krivulj, ki nam popisujejo hitrostni profil.

$$y(x) = Ax^2 + Bx + C \tag{51}$$
$$y(0) = y(1) = y_0 \implies C = y_0, A = -B \tag{52}$$
$$y = Ax(x-1) + y_0 \tag{53}$$
$$A = 6(y_0 - 1) \tag{54}$$
$$y(x) = 6(y_0 - 1)x(x-1) + y_0 \tag{55}$$

Ostal nam je še parameter $y_0$, v primeru, da zahtevamo še (12), bi ga izgubili. Že iz slike 5 vidimo, da zaradi kvadratne narave rešitve ne obstaja druga možnost, kot da nam pogoj prinese konstantno hitrost, nobena druga kvadratna funkcija namreč na robu ne bo zvezna s svojo predhodnico iz prejšnje cone.

# Opazka

Grafe sem risal z orodjem `matplotlib`. Običajen postopek za vse dele naloge je bilo pisanje funkcije, ki sem jo ročno ali z zankami opremil s potrebnimi parametri. Vedno sem delal s seznami iz okolja `numpy`, ki omogočajo vektoriranje kode; tako recimo za seznam

```
x = np.array([1,2,3,4]); x = 3 * x
```

operacija množenja s številom 3 ne bo vrnila trikrat ponovljenega seznama, kot bi to vrnil 'goli' Python, ampak vrne seznam, katerega elementi so bili pomnoženi s 3.

Pri najbolj kompleksni od definiranih funkcij (koda 5), ki je opisovala Lagranžian z dodanim kvadratom hitrosti, pa je ta pristop nepričakovano odpovedal. Funkcija je bila resda rahlo kompleksnejša, vendar bi to ne smelo biti ovira za uspešno delo z numeričnimi seznami. Rezultat mnogih poskusov prepisa in preoblikovanja kode je bil vselej

```
TypeError: only length-1 arrays can be converted to Python scalars.
```

Težava ni bila locirana, premostil sem jo z manj elegantno verzijo, preko t.i. izpeljanih seznamov:

```
ylist = [cetrti(i, y0, b = 0.01) for i in t]

plt.plot(t, ylist, label=r"$y_0 = {}$".format(y0))
```

Uspešno iskanje in odpravljanje napake ostaja na 'to–do' listku.

```
t = np.linspace(0, 1, 1000)

def peti(t,y0):
    A = 6*y0-6
    return A*t**2 - A*t + y0

plt.figure(figsize=(8,8))
for y0 in [0,0.5,1,1.5,2]:
    plt.plot(t, peti(t, y0), label=r"$y_0 = {}$".format(y0))
    plt.xlabel(r"Brezdimenzijski čas $x$")
    plt.ylabel(r"Brezdimenzijska hitrost $y$")
    plt.legend(loc="lower center")
plt.savefig("peti.pdf")
plt.savefig("peti.png")
plt.show()
```

Koda 1: Koda, s katero sem naredil grafe na sliki 5. Pripravil sem si večparametrično funkcijo, jo opremil s parametri in jo uporabil na `np.array` podatkovni strukturi, ki omogoča vektoriranje kode.

```
def cetrti(x, y0, b = 0.1):
    try:
        c = cosh(b)
        s = sinh(b)
        return (1-y0) / (s/b-c) * (cosh(b-b*x)-c) + y0
    except ZeroDivisionError:
        pass
```

Koda 2: Problematična funkcija (ena izmed mnogih prepisov, s katerimi sem poskušal rešiti težavo), ki nam opiše potek hitrosti pri četrtem delu naloge.