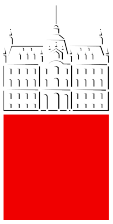
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Naloga 6: Enačbe hoda

Matematično-fizikalni praktikum

Dimitrije Pešić
Vpisna številka: 28201072

Predavatelj: prof. dr. Borut Paul Kerševan

Šmarješke Toplice, 15.11.2023

1 Uvod

Naloga je od nas zahtevala, da preizkusimo preprosto Eulerjevo metodo ter nato še čim več naprednejših metod (Midpoint, Runge-Kutto 4. reda, Adams-Bashfort-Moultonov prediktor-korektor ...) na primeru z začetnima temperaturama $y(0) = 21$ ali $y(0) = -15$, zunanjo temperaturo $y_{\text{zun}} = -5$ in parametrom $k = 0.1$. Ugotoviti smo morali kako velik (ali majhen) korak h je potreben? Za konec pa še izračun družine rešitev pri različnih vrednostih parametra k.

2 Ozadje

Za opis najpreprostejših fizikalnih procesov uporabljamo navadne diferencialne enačbe, ki povezujejo vrednosti spremenljivk sistema z njihovimi časovnimi spremembami. Tak primer je na primer enačba za časovno odvisnost temperature v stanovanju, ki je obdano s stenami z neko toplotno prevodnostjo in določeno zunanjo temperaturo. V najpreprostejšem primeru ima enačba obliko

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -k\,(T - T_{\text{zun}}) \tag{1}$$

z analitično rešitvijo

$$T(t) = T_{\text{zun}} + \mathrm{e}^{-kt}\,(T(0) - T_{\text{zun}})\ .$$

Enačbam, ki opisujejo razvoj spremenljivk sistema y po času ali drugi neodvisni spremenljivki x, pravimo *enačbe hoda*. Pri tej nalogi bomo proučili uporabnost različnih numeričnih metod za reševanje enačbe hoda oblike $\mathrm{d}y/\mathrm{d}x = f(x, y)$, kot na primer (1). Najbolj groba prva inačica, tako imenovana osnovna Eulerjeva metoda, je le prepisana aproksimacija za prvi odvod $y' \approx (y(x+h) - y(x))/h$, torej

$$y(x + h) = y(x) + h\ \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_x\ . \tag{2}$$

Diferencialno enačbo smo prepisali v diferenčno: sistem spremljamo v ekvidistantnih korakih dolžine h. Metoda je večinoma stabilna, le groba: za večjo natančnost moramo ustrezno zmanjšati korak. Za red boljša ($\mathcal{O}\,(h^3)$, t.j. lokalna natančnost drugega reda) je simetrizirana Eulerjeva (ali sredinska) formula, ki sledi iz simetriziranega približka za prvi odvod, $y' \approx (y(x+h) - y(x-h))/2h$. Računamo po shemi

$$y(x + h) = y(x - h) + 2h\ \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_x\ , \tag{3}$$

ki pa je praviloma nestabilna. Želeli bi si pravzaprav nekaj takega

$$y(x + h) = y(x) + \frac{h}{2}\ \left[\left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_x + \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_{x+h}\right]\ , \tag{4}$$

le da to pot ne poznamo odvoda v končni točki intervala (shema je implicitna). Pomagamo si lahko z iteracijo. Zapišimo odvod kot:

$$\left.\frac{dy}{dx}\right|_x = f(x, y)$$

ter

$$x_{n+1} = x_n + h, \quad y_n = y(x_n)$$

Heunova metoda ($\mathcal{O}\,(h^3)$ lokalno) je približek idealne formule z:

$$\hat{y}_{n+1} = y_n + h \cdot f(x_n, y_n) \tag{5}$$

$$y_{n+1} = y_n + \frac{h}{2}\,[f(x_n, y_n) + f(x_{n+1}, \hat{y}_{n+1})] \tag{6}$$

Izvedenka tega je nato Midpoint metoda (tudi $\mathcal{O}\left(h^3\right)$ lokalno):

$$k_1 = f(x_n, y_n) \tag{7}$$
$$k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}\,h\,k_1) \tag{8}$$
$$y_{n+1} = y_n + h\,k_2 \tag{9}$$

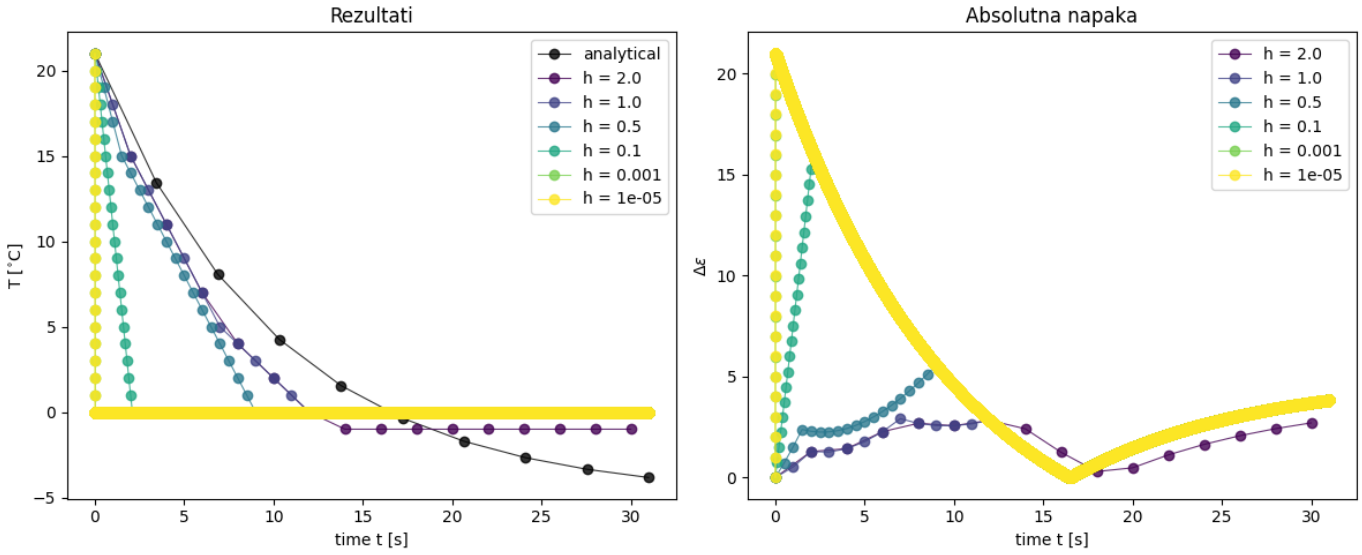
Le-to lahko potem izboljšamo kot modificirano Midpoint metodo itd...

V praksi zahtevamo natančnost in numerično učinkovitost, ki sta neprimerno boljši kot pri opisanih preprostih metodah. Uporabimo metode, zasnovane na algoritmih prediktor-korektor, metode višjih redov iz družine Runge-Kutta (z adaptivnimi koraki), ali ekstrapolacijske metode. Brez dvoma ena najbolj priljubljenih je metoda RK4,

$$\begin{aligned} k_1 &= f\left(x,\, y(x)\right)\,, \\ k_2 &= f\left(x + \tfrac{1}{2}h,\, y(x) + \tfrac{h}{2}k_1\right)\,, \\ k_3 &= f\left(x + \tfrac{1}{2}h,\, y(x) + \tfrac{h}{2}k_2\right)\,, \\ k_4 &= f\left(x + h,\, y(x) + hk_3\right)\,, \\ y(x+h) &= y(x) + \tfrac{h}{6}\,\left(k_1 + 2k_2 + 2k_3 + k_4\right) + \mathcal{O}(h^5)\,. \end{aligned} \tag{10}$$

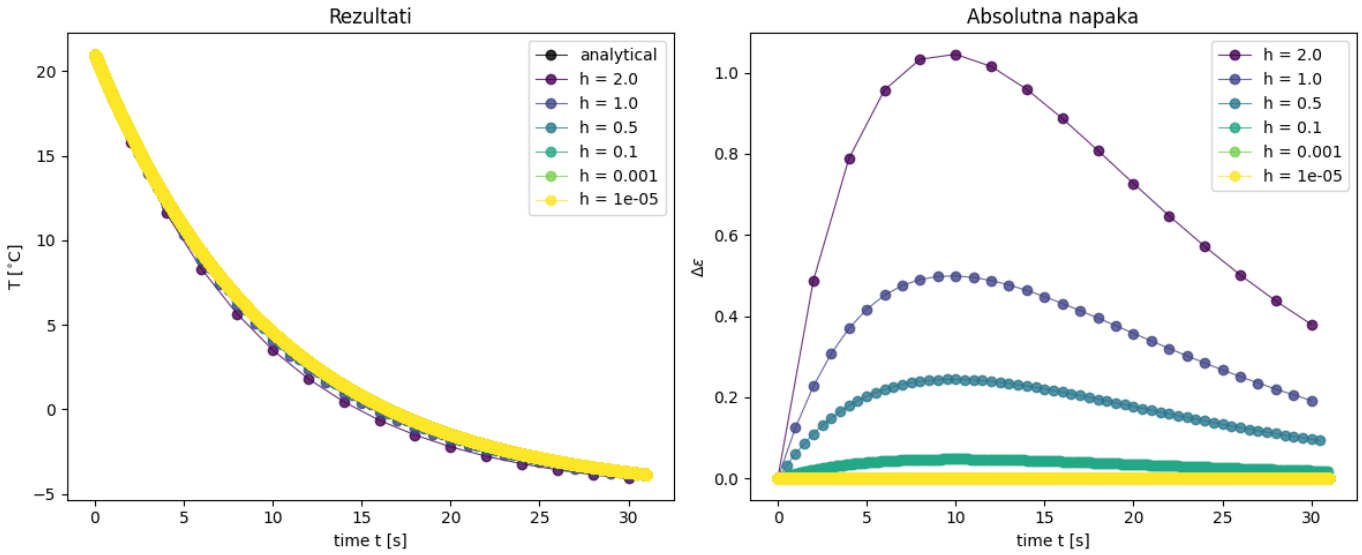
3 Implementacija

Pri implementaciji sem sprva imel veliko težav z že implementiranimi numeričnimi metodami in sicer manjši kot je bil korak pri integraciji, slabša je bila natančnost (Prikazano za Eulerjevo metodo na grafu 1



Slika 1: Artefakt pri integraciji.

Za razreševanje sem porabil več kot pol dneva. Očitno sem naletel na isti problem kot nekdo v prejšnjem letniku, saj me je njegov/njen komentar rešil iz zadrege. Napaka je bila v tem da sem za začetni pogoj vzel 'integer' vrednost namesto 'float' (očitno se malo preveč zanašam na pythonov 'typecasting'). Po razrešeni napaki se metode obnašajo kot bi le pričakovali 2.

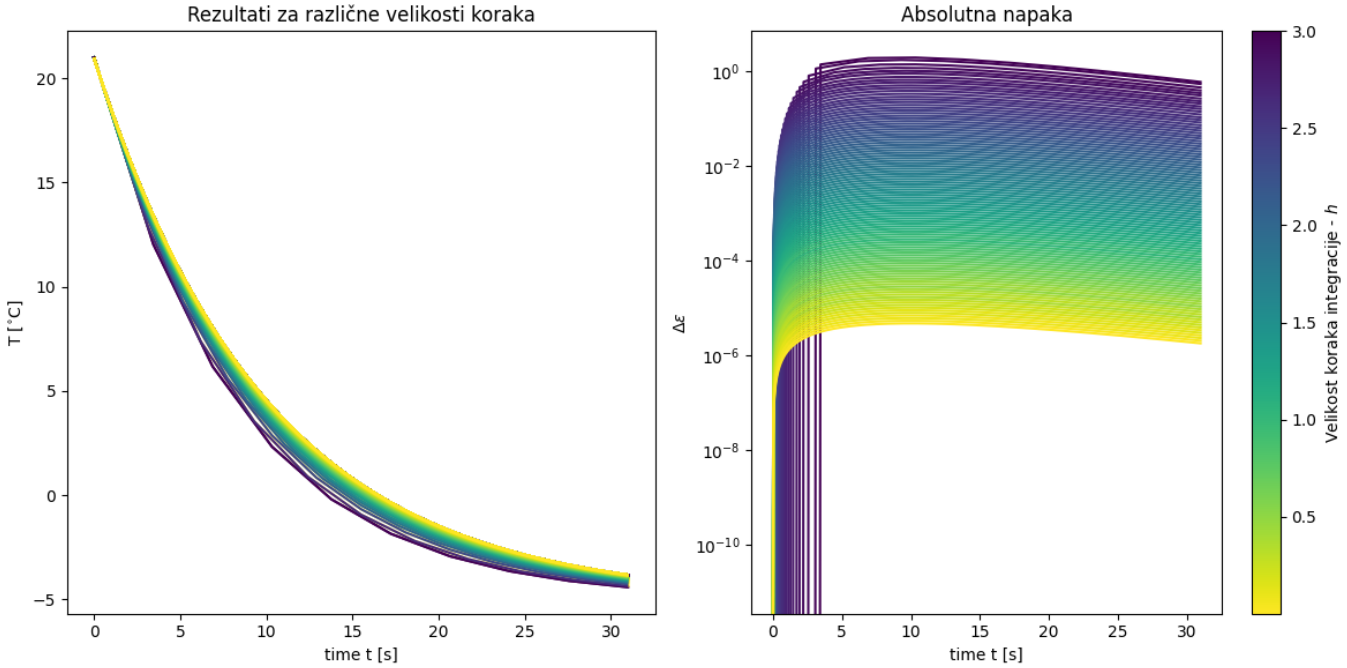


Slika 2: Rezultati integracije.

4 Natančnost

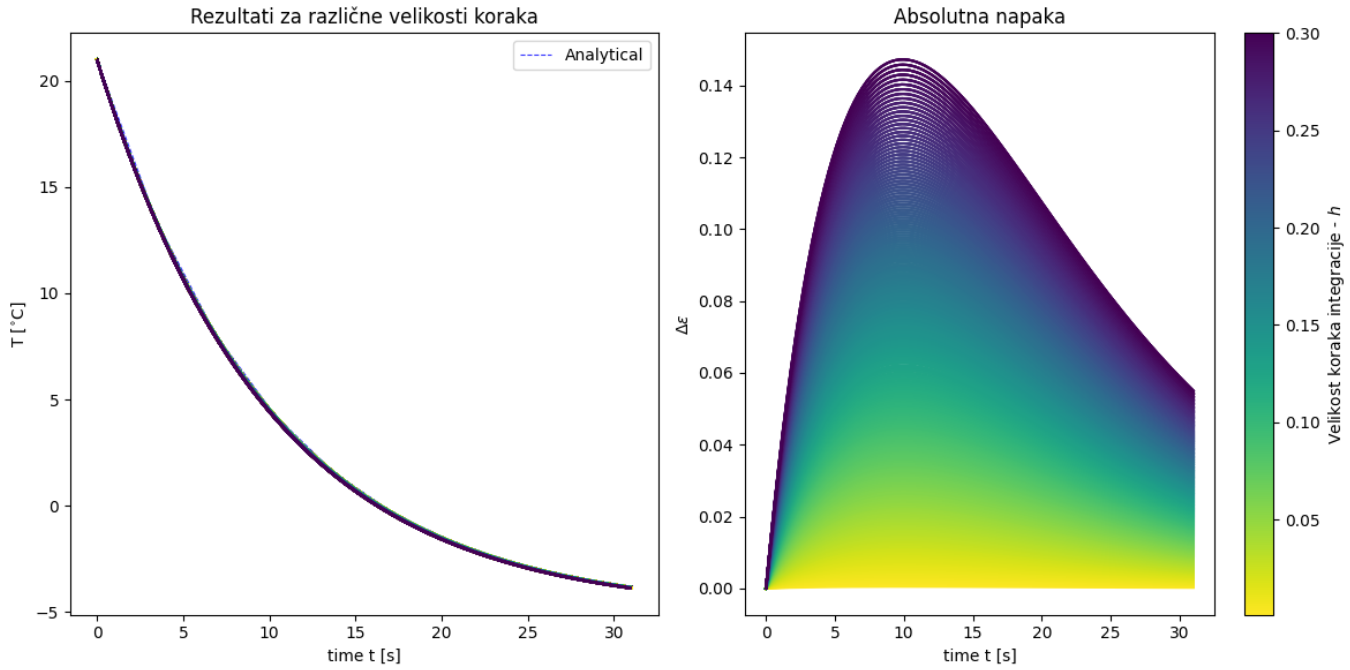
Sedaj ko imamo implementirane metode, sem preveril natančnost metod za različne velikosti korakov pri integraciji.

Sprva samo na Eulerjevi metodi. Iz grafov 3 in 4 je razvidno, da metoda za manjši korak izračuna z večjo natančnostjo. Seveda to natančnost poplačamo z daljšim časom izvajanja.



Slika 3: Rezultati integracije z Eulerjevo metodo.

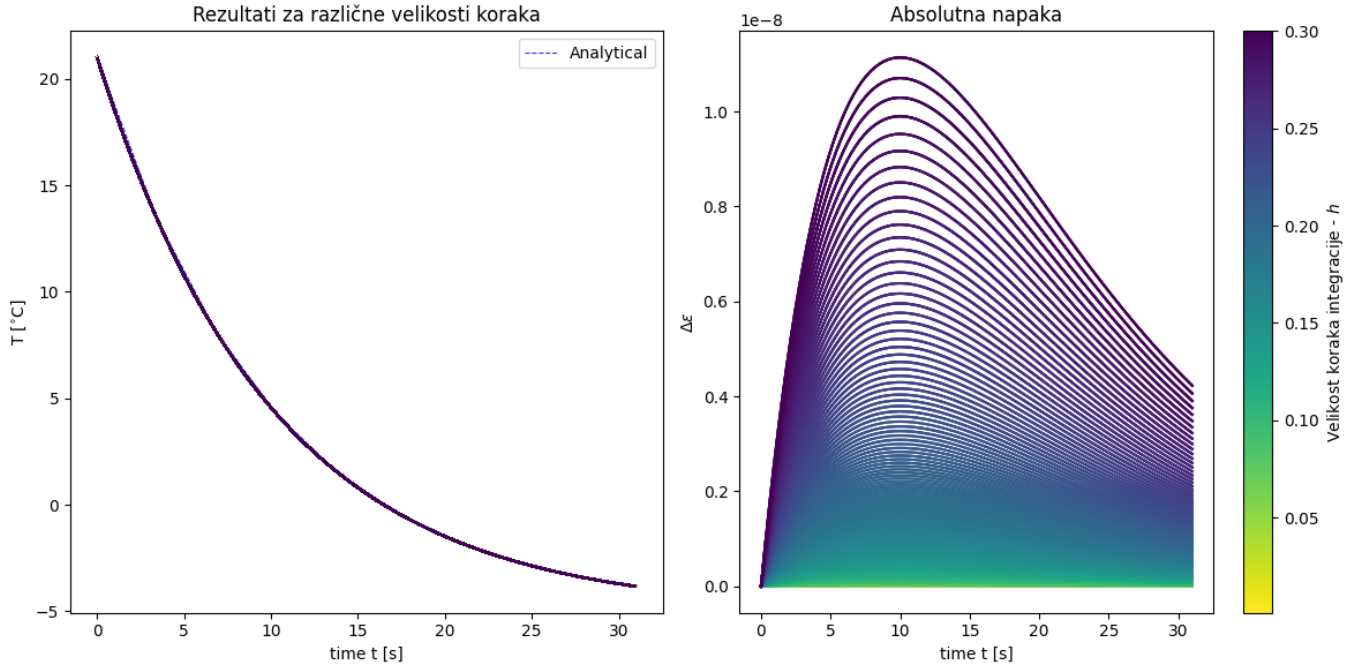
Želel sem prikazati skalarno polje vrednosti, vendar mi ni uspelo pogruntati elegantnega načina brez da bi zapravil še en dan časa ko so kolokviji za vogalom tedna. Zato sem implementiral izračun na star, zanesljiv način, z surovo silo. Implementiral sem *'multithreading'*, to se izkaže kot koristno tudi



Slika 4: Rezultati integracije v logaritemski skali.

kasneje pri meritvi časa, saj mi omogoči relativno hitro izvajanje kode za veliko število integralov.

Na grafu 5 je prikazan še isti izračun za Runge-Kutta-45 metodo. Razvidna je mnogo večja natančnost, kot pri Eulerjevi metodi. Več v naslednjem poglavju

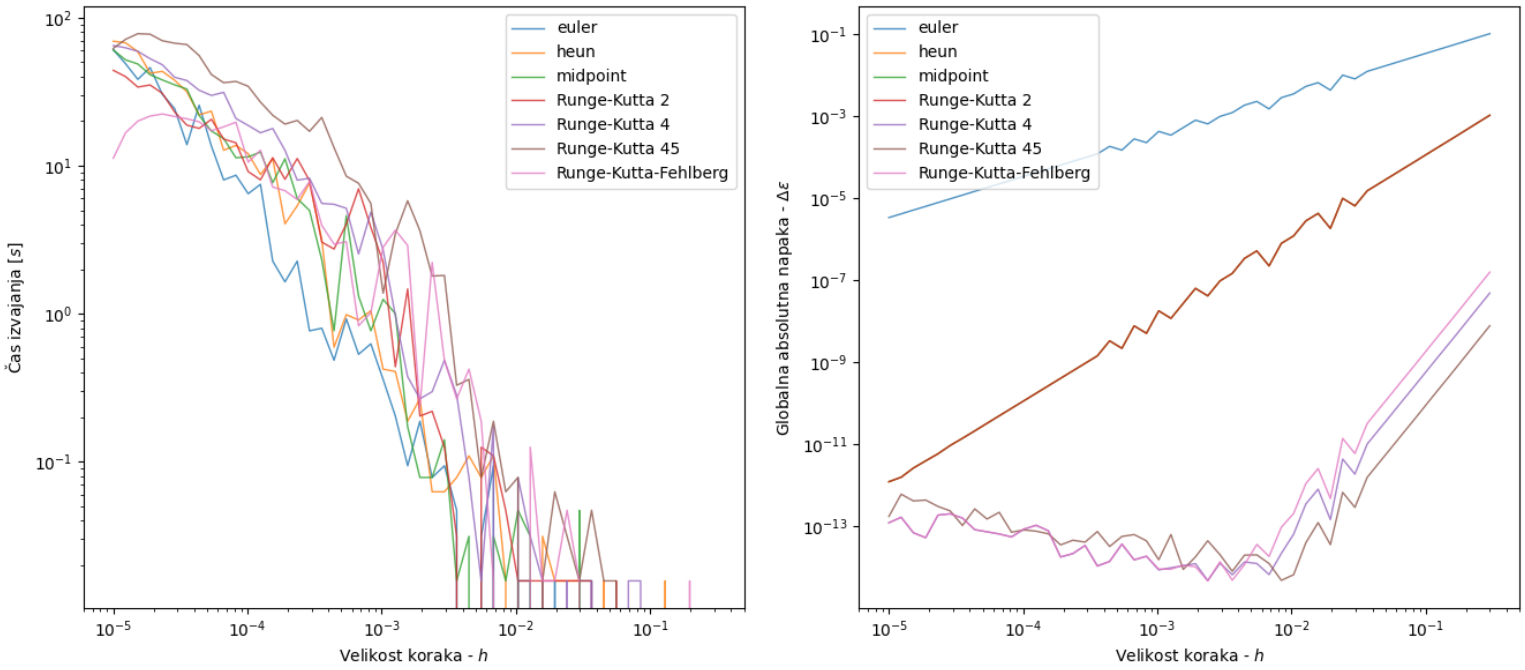


Slika 5: Rezultati integracije z uporabo Runge-Kutta-45.

5 Čas izvajanja

Izmeril sem čas izvajanja za več numeričnih metod. Čas izvajanja sem meril z knjižnico *time* in specifično funkcijo *time.process_time*, ki vrne vrednost (v delčkih sekund) vsote sistemskega in uporabniškega procesorskega časa trenutnega procesa. Ne vključuje časa, ki je pretekel med spanjem.

Na sliki 6 je prikazan čas izvajanja v odvisnosti od velikosti koraka k, ter absolutna napaka tudi v odvisnosti od koraka pri integraciji za vse metode.

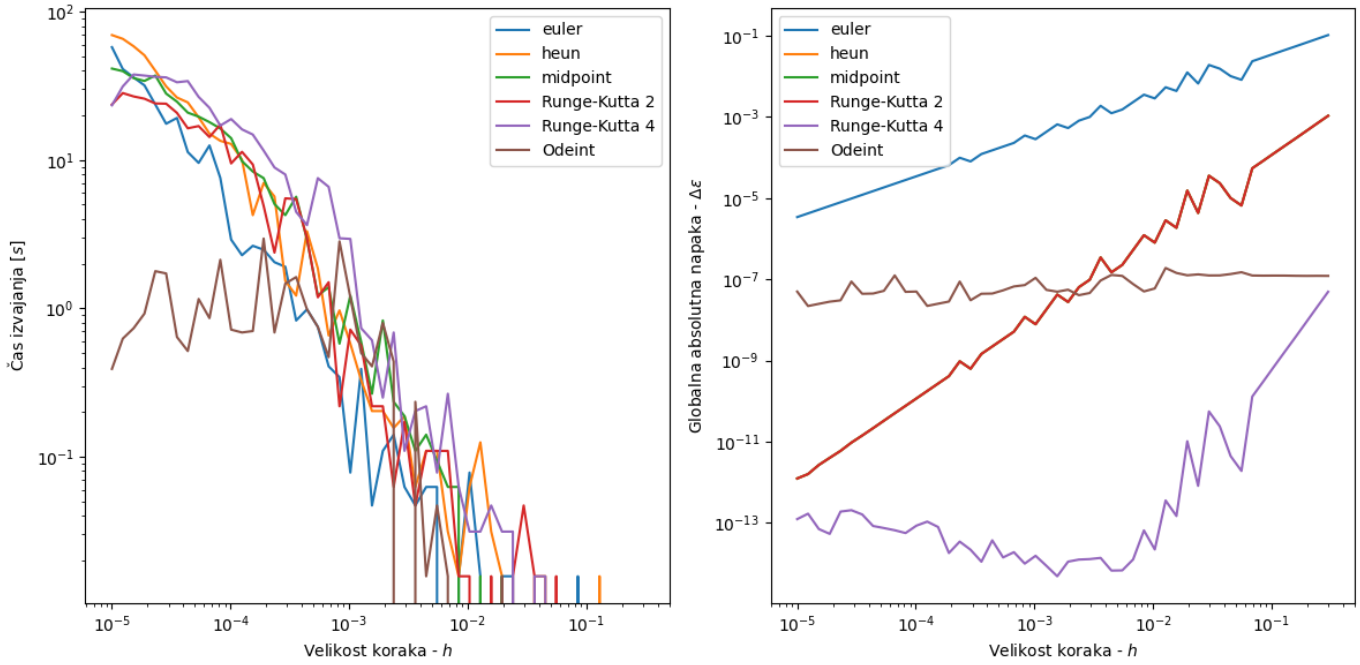


Slika 6: Čas izvajanja za vse metode.

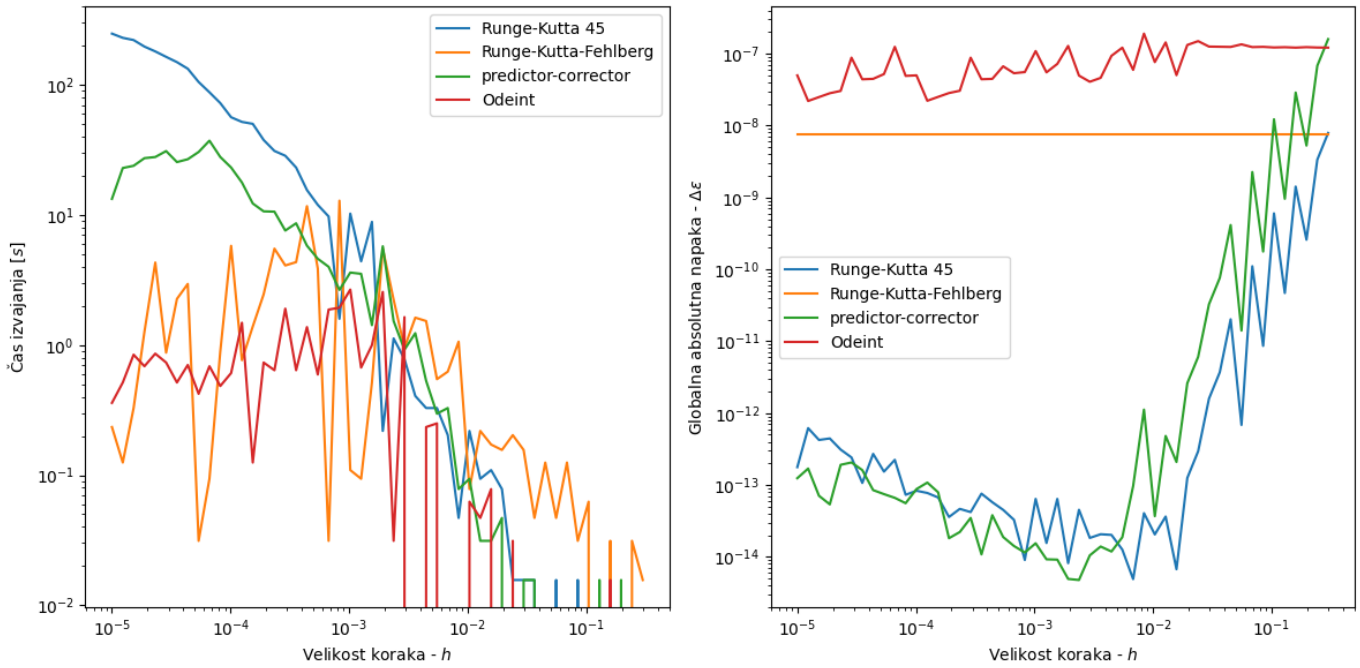
Za lepšo preglednost, sem na ločil nekaj metod. Na sliki 7 je prikazan čas izvajanja za Eulerjevo metodo, Heun, Midpoint, Runge-Kutta drugega reda ter Runge-Kutta četrtega reda. Razvidna je večanje časa izvajanja pri manjšem koraku, ter večanje absolutne napake pri večjem koraku.

Isto sem storil še za Runge-Kutta metodo četrtega reda z oceno napake, Runge-Kutta-Fehlberg, Adams-Bashforth-Moulton predictor-corrector metodo ter za *scipy.integrate.odeint* funkcijo iz knjižnice *scipy* 8.

Metoda Runge-Kutta-Fehlberg temelji na adaptivnem koraku, zato ima le ta metoda konstantno napako. Optimalno velikost koraka pri integraciji si izberemo glede na željeno natančnost, metodo ki jo želimo uporabiti ter z upoštevanjem konstrikcij z časovnim izvajanjem.



Slika 7: Čas izvajanja za preprostejše metode.

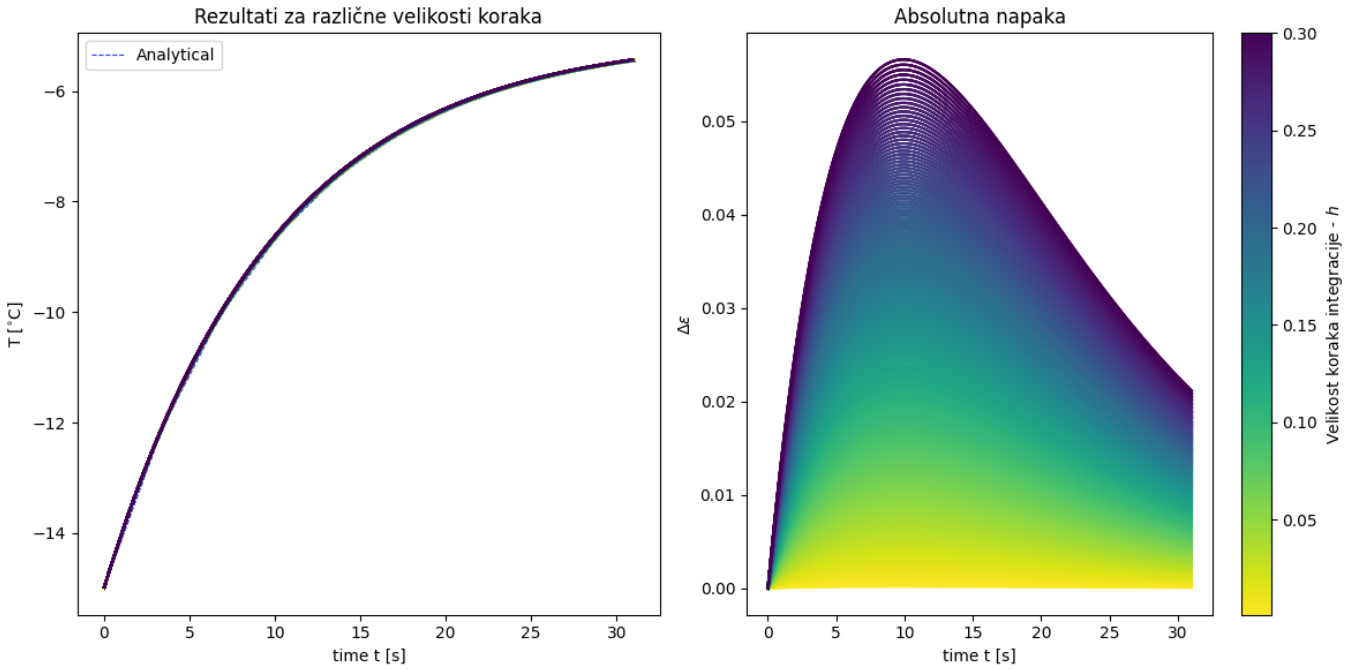


Slika 8: Čas izvajanja za naprednejše metode.

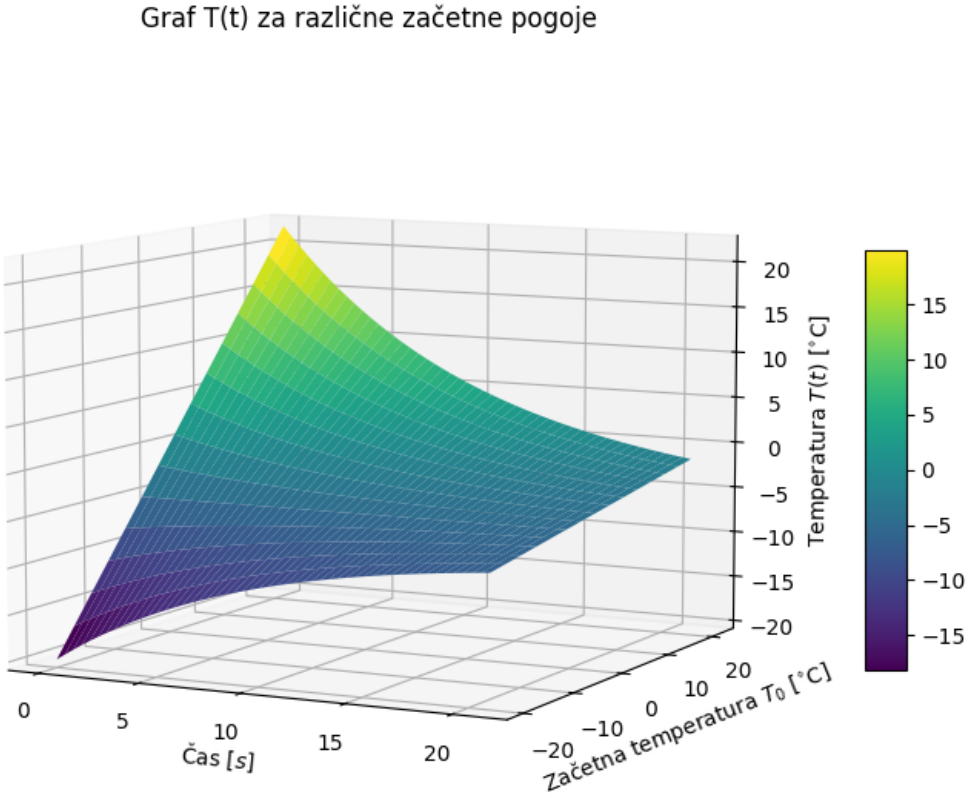
6 Rešitev za poljubno začetno temperaturo

Sedaj ko imamo metode in dovolj veliko natančnost sem, kot je naloga zahtevala, izračunal še rezultat integralov pri začetnem pogoju $T(0) = -15°\mathrm{C}$ prikazano na grafu 9

Dodatno sem še želel prikazati rezultat integracije za vse vrednosti temperatur vmes. To sem storil z tridimenzionalnim grafom prikazanim na sliki 10



Slika 9: Rezultati integracije pri začetni temperaturi $T(0) = -15°\text{C}$

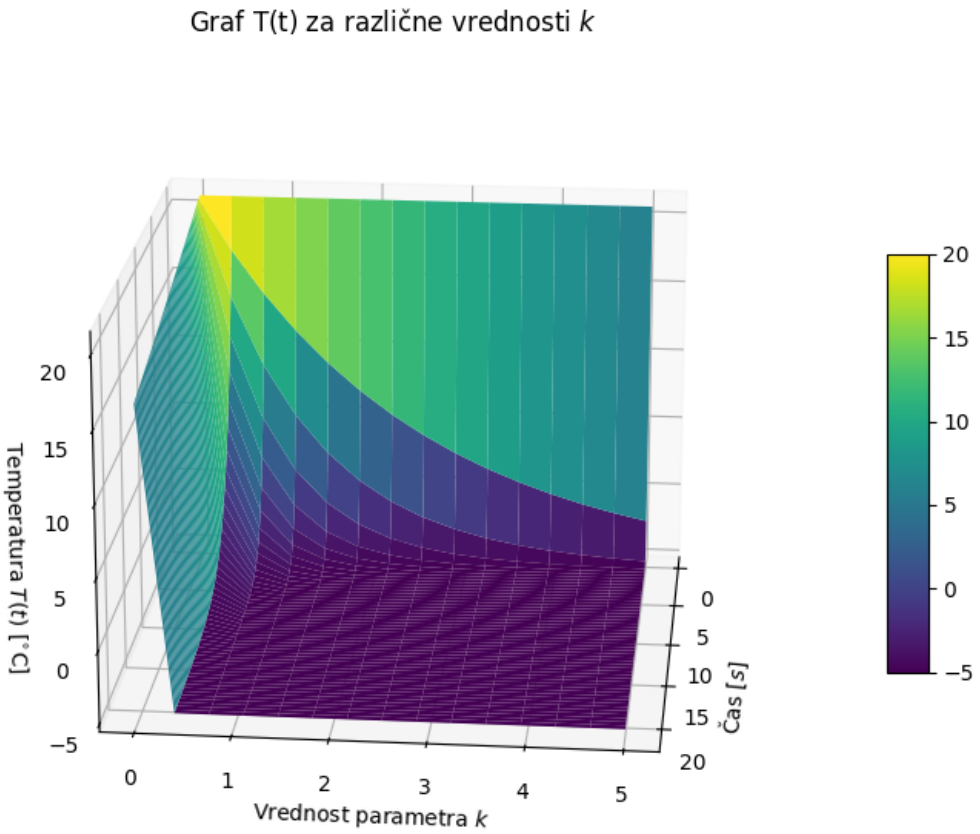


Slika 10: Rezultati T(t) pri različnih začetnih temperatura T_0.

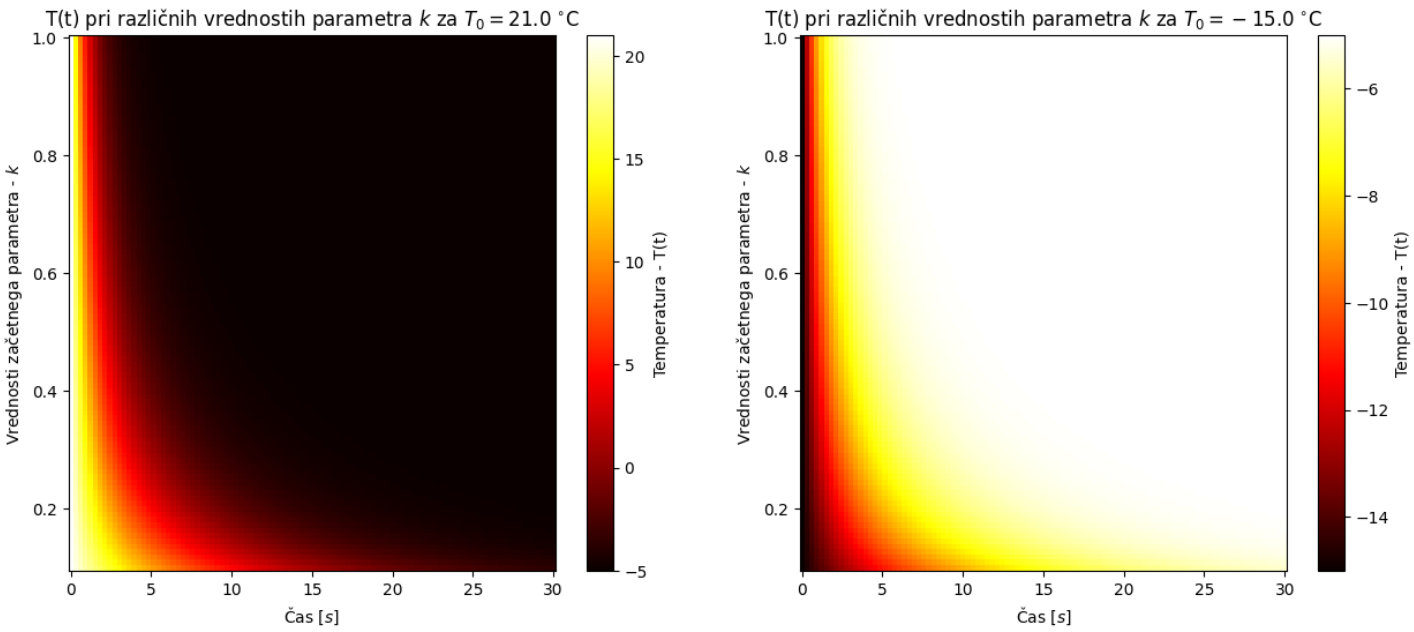
Žal je to le projekcija. Brez interaktivnosti 3D grafa, zato so malo težje razvidne oblike grafa.

7 Različne vrednosti parametra k

Za konec je naloga zahtevala še izračun integralov pri različnih vrednostih začetnega parametra k v formuli 1. To sem prikazal na 3D grafu 11 ter na toplotnem zemljevidu 12.



Slika 11: Rezultati T(t) pri različnih vrednostih parametra k.



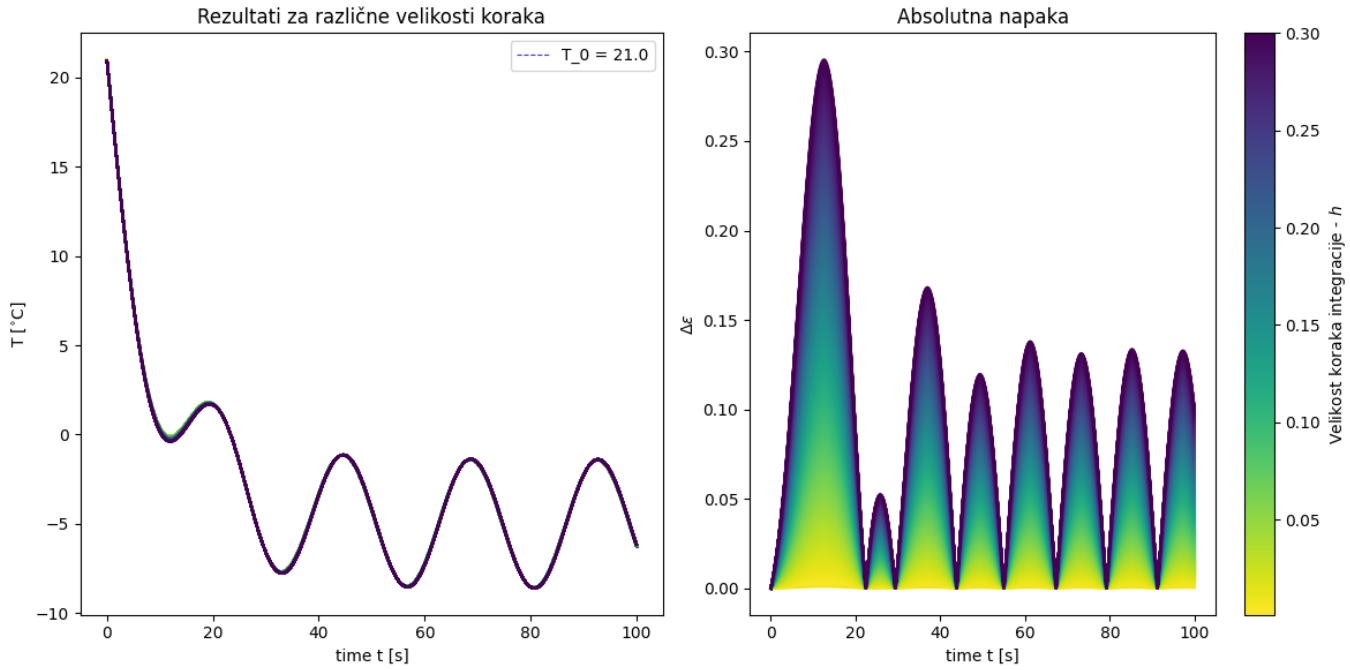
Slika 12: Rezultati T(t) pri različnih vrednostih parametra k.

8 Dodatna naloga

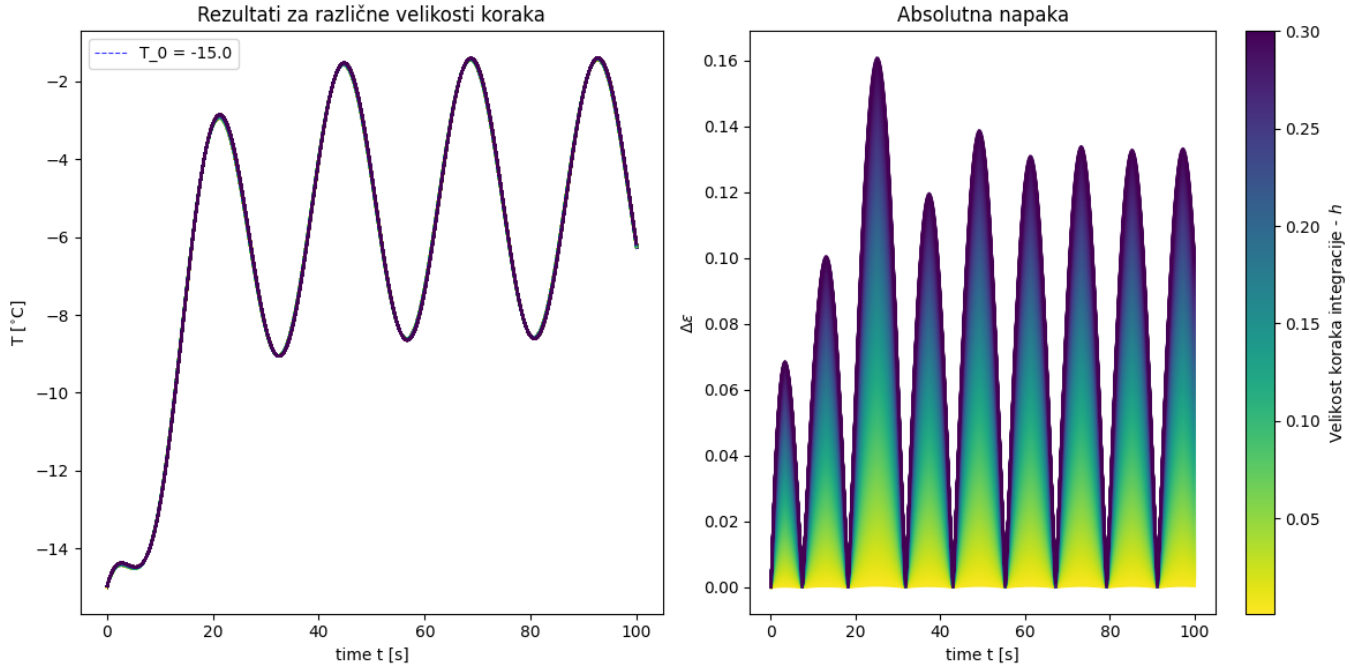
Dodatna naloga je od nas zahtevala izračunamo, kako se temperatura spreminja, če se lahko prostor dodatno segreva ali ohlaja zaradi recimo sončevega segrevanja skozi okna, s 24-urno periodo in nekim faznim zamikom δ, kar opišemo z dva- ali triparametrično enačbo:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -k(T - T_{zun}) + A \sin\left(\frac{2\pi}{24}(t - \delta)\right) \ . \tag{11}$$

Rezultat za začetne vrednosti parametrov $k = 0.1$, $\delta = 10$, $A = 1$, ter temperaturo $T(0) = 21.0°\mathrm{C}$ na grafu 13 ter za temperaturo $T(0) = -15.0°\mathrm{C}$ na grafu 14.



Slika 13: Rezultati T(t) s periodičnim segrevanje.

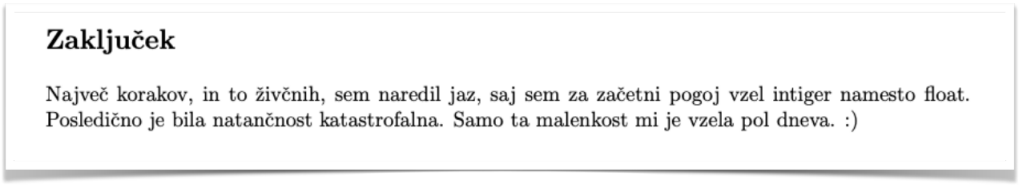


Slika 14: Rezultati T(t) s periodičnim segrevanje.

9 Zaključek

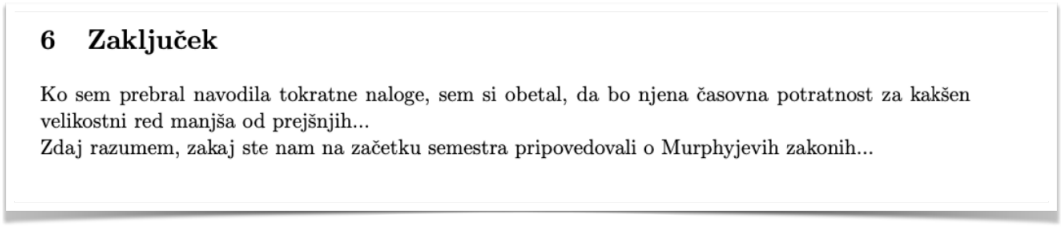
Cilj naloge je bil, da se spoznamo z uporabo in delovanjem različnim algoritmov za numerično integracijo. Najtežji del naloge je bil sama predstavitev podatkov na eleganten način. Verjamem, da je veliko boljših načinov za izvedbo le tega kot pa sem jaz implementiral. Žal mi je za globlje premisleke zmanjkalo časa, del razloga sem že opisal pri implementaciji, del pa je le bližujoči se kolokviji za katere je tudi potreben čas ki izginja v črno luknjo mafijskega praktikuma.

Moje težave z nalogo odlično povzameta dva komentarja lanskih mučenikov 9



Zaključek

Največ korakov, in to živčnih, sem naredil jaz, saj sem za začetni pogoj vzel intiger namesto float. Posledično je bila natančnost katastrofalna. Samo ta malenkost mi je vzela pol dneva. :)



6 Zaključek

Ko sem prebral navodila tokratne naloge, sem si obetal, da bo njena časovna potratnost za kakšen velikostni red manjša od prejšnjih...
Zdaj razumem, zakaj ste nam na začetku semestra pripovedovali o Murphyjevih zakonih...

Literatura

[1] Simon Bukovšek, *Skripta predavanj Matematika 4, Ljubljana 2023.*