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

Matematično-fizikalni praktikum

Tretja naloga: *Enačbe hoda*

Simon Bukovšek, 28211067

Mainz, 15. november 2023

Profesor: prof. dr. Borut Paul Kerševan

Naloga: Enačbe hoda

Preizkusi 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 $T(0) = 21\,^\circ\mathrm{C}$ ali $T(0) = -15\,^\circ\mathrm{C}$, zunanjo temperaturo $T_{\mathrm{zun}} = -5\,^\circ\mathrm{C}$ in parametrom $k = 0{,}1\,\mathrm{K/h}$. Kako velik (ali majhen) korak h je potreben? Izberi metodo (in korak) za izračun družine rešitev pri različnih vrednostih parametra k.

1 Uvod

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

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

z analitično rešitvijo

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

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, $y' \approx (y(x+h) - y(x-h))/2h$. Želeli bi si pravzaprav nekaj takega

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

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

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

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:

$$\hat{y}_{n+1} = y_n + h \cdot f(x_n, y_n) \tag{4}$$

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, \hat{y}_{n+1})\right] \tag{5}$$

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

$$k_1 = f(x_n, y_n) \tag{6}$$

$$k_2 = f(x_n + h/2, y_n + 1/2\,h\,k_1) \tag{7}$$

$$y_{n+1} = y_n + h\,k_2 \tag{8}$$

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

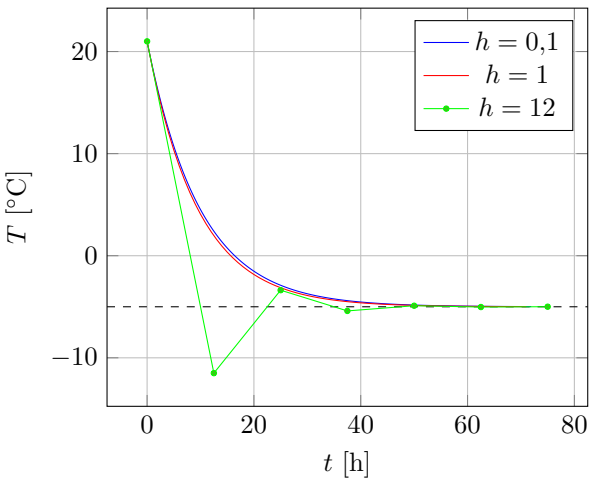
V praksi zahtevamo natančnost in numerično učinkovitost, ki sta neprimerno boljši kot pri opisanih preprostih metodah. Uporabimo metode, zasnovane na algoritmih prediktor-korektor, metode višjih redov iz družine Runge-Kutta (z adaptivnimi koraki), ali ekstrapolacijske metode. Brez dvoma ena najbolj priljubljenih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

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

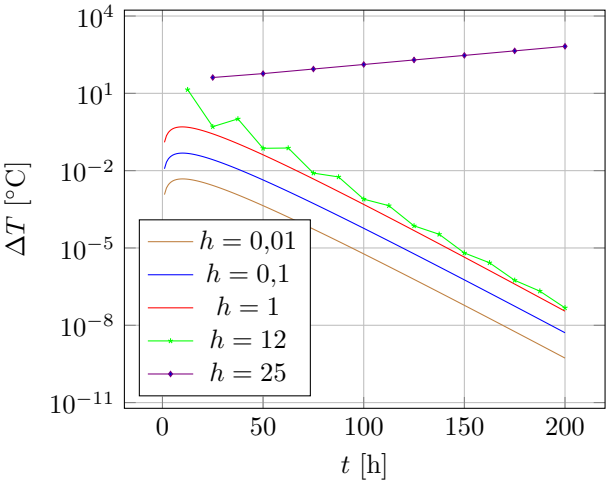
2 Reševanje enačbe hoda

2.1 Eulerjeva metoda

Če ni rečeno drugače, vedno velja, da je $T(0) = 21\,°\mathrm{C}$, $T_{\mathrm{zun}} = -5\,°\mathrm{C}$ in $k = 0{,}1\,\mathrm{K/h}$. Najprej sem se spoprijel z Eulerjevo metodo. Zanimalo me je, kako se spreminja končna napaka po 75 urah pri različnih korakih. Na Sliki 1a so prikazane rešitve za korake $\Delta t = h \in [0{,}1\,\mathrm{h}, 1\,\mathrm{h}, 12\,\mathrm{h}]$. Popravek iz 1 h na 0,1 h je že tako majhen, da se na videz skoraj ne pozna več. Sprva sem hotel narisati rešitev s korakom velikosti 10 ur, vendar je za 12 ur bolj zanimiv. Že v prvem koraku preskoči asimptoto, vendar se vseeno "ujame" v pravi trend in konvergira h pravi vrednosti. Kakšnih pretirano pametnih stvari pa nam 1a ne pove, zato si raje poglejmo razlike med simuliranimi in analitičnimi rešitvami za različne vrednosti koraka. Te so prikazane na Sliki 1b. V splošnem velja, da z manjšim korakom dosežemo boljšo natančnost, zaradi narave same enačbe pa se tudi s časom vedno bolj približujemo pravi vrednosti. Za kakšno bolj zapleteno enačbo bi sicer moralo veljati ravno obratno, z vsakim korakom bi se morali od prave rešitve nekoliko oddaljiti. Približevanje pravi rešitvi pa tudi v našem primeru ni vedno nujno, saj to velja samo za korake, ki so manjši od 20 ur. Z vijolično vidimo prikazan tudi primer za $h = 25\,\mathrm{ur}$, ki pa se s časom oddaljuje od prave vrednosti. Taka rešitev torej ni stabilna.



(a) Nekaj prikazov rešitev enačbe hoda z Eulerjevo metodo za različno velike korake. Vidimo, da je zelena krivulja kljub velikemu koraku še vedno stabilna.

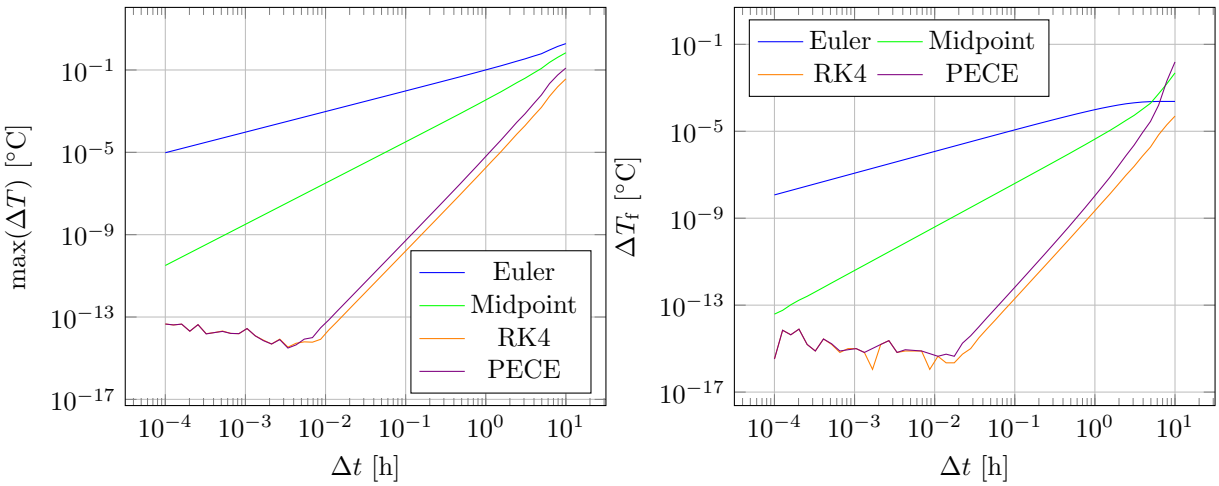


(b) Razlike med analitično in numerično rešitvijo za različne korake v odvisnosti od časa. Za manjše korake je napaka manjša, pri $h > 20\,\mathrm{ur}$ pa rešitve postanejo nestabilne.

Slika 1: Reševanje enačbe hoda z osnovno Eulerjevo metodo.

2.2 Preostali živalski vrt metod

Eulerjeva metoda je bila samo za ogrevanje. Poleg nje sem implementiral še štiri druge metode: Heunovo metodo, Midpoint metodo, Runge-Kutta metodo četrtega reda (RK4) in Adams-Bashforth-Moultonov prediktor-korektor (predict-evaluate-correct-evaluate ali PECE). Vse metode sem najprej preizkusil na enak način kot Eulerjevo metodo, torej sem jih primerjal z analitično rešitvijo in preveril, kako se spreminja napaka s časom. Osredotočil sem se na dve stvari: oddaljenost numerično izračunane rešitve na koncu intervala ($t = 100\,\mathrm{h}$) od analitične rešitve ter maksimalno oddaljenost kjerkoli v rešitvi. Na Sliki 2a je maksimalna oddaljenost, na Sliki 2b pa končna oddaljenost. Pomembno je, da v našem primeru končna oddaljenost ni enaka največji, saj ima rešitev zelo močno asimptoto. Črte na Grafu 2a predstavljajo potenčne funkcije. Potenca pri Eulerjevi metodi je enaka $1{,}00 \pm 0{,}03$, potenca pri Heunovi in Midpoint metodi je $2{,}00 \pm 0{,}01$, pri PECE in RK4 metodi pa $4{,}0170 \pm 0{,}0001$. Vsi ti nakloni so v skladu s teoretičnimi napovedmi. Lepo pa se vidi še en pojav: natančnost pri RK4 in PECE metodi pada do približno $\Delta t = 7 \cdot 10^{-3}\,\mathrm{h}$, nato pa se z manjšanjem koraka celo začne večati. Obrat se zgodi, ko natančnost metode doseže natančnost float64, manjšanje koraka pa le še veča število seštevanj in zato je nakopičena napaka zaokrožanja večja.

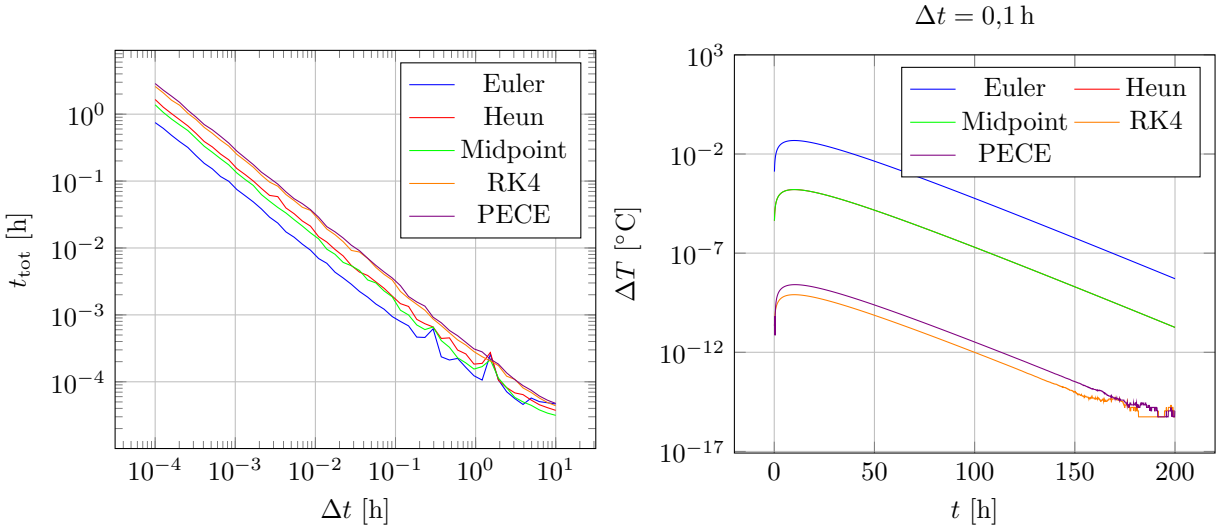


(a) Maksimalno odstopanje od analitične rešitve v odvisnosti od velikosti koraka.

(b) Končno odstopanje od analitične rešitve za različne velikosti koraka.

Slika 2: Odstopanja petih različnih metod: Eulerjeve, Heunove, Midpoint, Runge-Kutta 4. reda (RK4) in Adams-Bashforth-Moultonov prediktor-korektor (PECE). Heunova metoda ni prikazana, saj, kot kaže, vrne za našo diferencialno enačbo popolnoma isto rešitev kot Midpoint metoda.

Prej sem omenil, da maksimalno odstopanje ne nastopi na koncu intervala, temveč bolj proti začetku. Slika 3b nam prikazuje časovno odvisnost napake pri koraku $\Delta t = 0{,}1\,\mathrm{h}$. Pri tem koraku so vse metode stabilne, vendar se vseeno opazi hierarhija natančnosti. Najnatančnejša je Runge-Kutta, sledi prediktor-korektor in kmalu Midpoint ter Heunova metoda, čisto na koncu pa za več kot red velikosti odstopa Eulerjeva metoda. Pri RK4 se celo že opazi, da je natančnost rešitve pri $t = 200$ h presegla številsko natančnost `float64`.



(a) Časovna zahtevnost izbranih metod v odvisnosti od velikosti koraka.

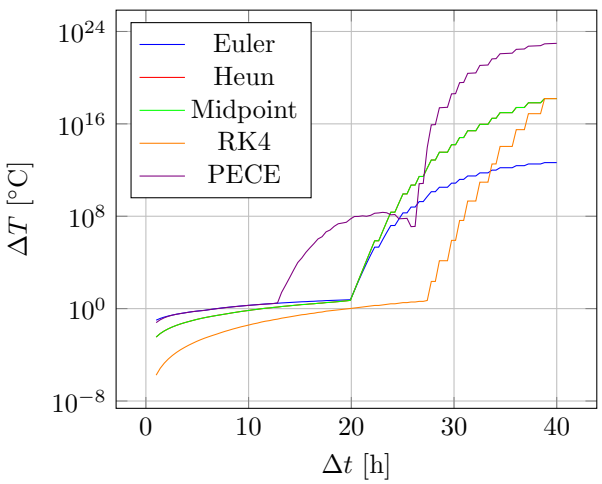
(b) Odvisnost odstopanja od analitične rešitve v odvisnosti od časa (v rešitvi) za izbrane metode pri velikosti koraka $\Delta t = 0{,}1\,\mathrm{h}$.

Slika 3: Dodatne raziskave glede natančnosti in časovne zahtevnosti metod.

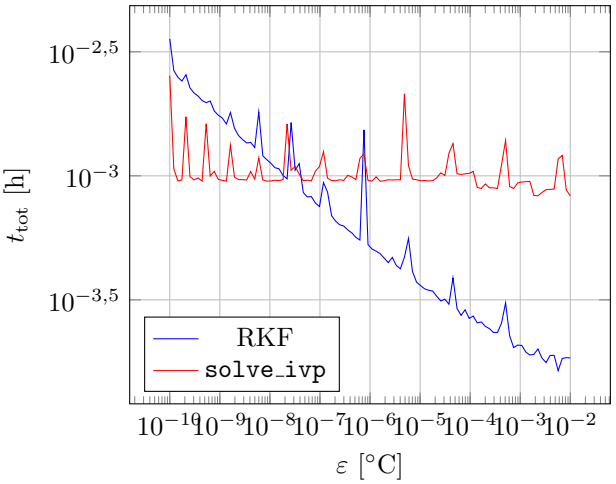
Enako pomembna kot natančnost metode pa je tudi časovna učinkovitost. Vsem omenjenim metodam sem meril čas izvajanja za različne dolžine korakov in rezultate prikazal na Sliki 3a. Eulerjeva metoda je najhitrejša, saj je tudi najbolj preprosta in ima najmanj vrednotenj funkcije (za ceno manjše natančnosti).

Vrstni red je po časovni zahtevnosti praktično obraten vrstnemu redu po natančnosti, le razlike so nekoliko drugačne. Najpočasnejši sta RK4 in PECE metodi, pri čemer je RK4 bistveno natančnejša. Midpoint pa za odtenek prehiti Heunovo metodo, čeprav dasta na koncu isti rezultat. Pomembno je poudariti, da so to rezultati metod, ki sem jih napisal sam. To pomeni, da je lahko zaradi kakšne nepazljivosti določena metoda po nepotrebnem počasnejša, kot bi bila v optimalnem primeru. Za vse metode sem preveril naklon grafa in ugotovil, povsod velja $t_{\text{tot}} \propto h^{-1{,}00\pm0{,}01}$, kar je v skladu s teoretičnimi napovedmi.

Zadnja pomembna lastnost metod reševanja diferencialnih enačb je stabilnost. Hkrati pa je to lastnost najlažje preveriti. Namesto da bi korak manjšal, sem ga večal in opazoval, kje se zgodi preskok v natančnosti. Vzel sem časovni interval do 1000 ur in povečeval korake po eno uro. Pri Eulerjevi metodi smo že na Grafu 1b opazili, da pri $\Delta t = 25\,\text{h}$, stabilnosti ni več. Na Sliki 4a se čudovito vidi meja stabilnosti za vse obravnavane metode. Euler, Heun in Midpoint imajo mejo pri $\Delta t = 20\,\text{h}$, prediktor-korektor pri $\Delta t = 12{,}5\,\text{h}$ in RK4 pri $\Delta t = 28\,\text{h}$. Heunova metoda se ponovno skriva za Midpoint. Zanimivo je opaziti neke vrste otoček semi-stabilnosti pri PECE metodi okoli $\Delta t = 60\,\text{h}$. Na žalost se na to temo ne spoznam zadosti, da bi lahko komentiral ta pojav. Poleg vseh že omenjenih metod pa sem implementiral še eno:



(a) Maksimalno odstopanje od analitične rešitve v odvisnosti od velikosti koraka. Prelom na grafu označuje mejo stabilnosti. Zna kdo razložiti dva preloma na grafu PECE metode? (Jaz ne.)



(b) Primerjava moje implementacije Runge-Kutta-Fehlbergove metode (RKF) z vgrajeno metodo `solve_ivp` iz knjižnice `scipy.integrate`. Graf prikazuje skupen čas za izračun do $t_{\max} = 50\,\text{h}$ pri dani zahtevani natančnosti ε.

Slika 4: Še več eksperimentiranja z različnimi metodami.

Runge-Kutta-Fehlbergovo metodo (RKF oz RK45), ki pa je zaradi posebnega načina delovanja nisem mogel vključiti v isto analizo kot ostale. Metodi se namreč ne predpiše velikosti koraka, temveč natančnost. Metoda nato sama izbere primerno (in ne nujno enakomerno) dolge korake. Funkcija `solve_ivp` iz knjižnice `scipy.integrate` uporablja isto metodo, zato sem se odločil, da preverim še te dve. Rezultati so na Sliki 4b. Rezultati so presenetljivi. Pri velikih vrednostih zahtevane natančnosti je moj program hitrejši od `scipy`-ja, najbrž zaradi zahtevane najmanjše velikosti koraka pri slednjem. Sam v svoj program nisem vgradil nobene zgornje meje za velikost koraka, zato sklepam, da v primerih nizke zahtevane natančnosti program uporabi gromozanske korake in s tem pridobi prednost pred `scipy`-jem. Pri visokih zahtevanih natančnostih pa je `scipy` hitrejši, zaradi vseh že velikokrat omenjenih prednosti te knjižnice.

Recimo, da želimo odstopanje največ $\varepsilon = 10^{-5}\,\text{K}$ na celotnem izračunanem intervalu med začetkom in končnim časom $t_{\max} = 200\,\text{h}$. V Tabeli 1 so prikazani časi, ki jih posamezne metode potrebujejo za izračun rešitve pri zahtevani natančnosti. Komentirajmo nekaj opazk. Pri vseh metodah sem se potrudil, da sem korak izbral, tako da sem se čim bolj približal željeni natančnosti. Pri RKF metodi sem preprosto vstavil izbrano natančnost, dobljen rezultat pa je štirikrat bolj natančen. Zatorej sem vstavil štirikrat manjšo natančnost, da je končna natančnost primerljiva z ostalimi. Razlike pa so v resnici zelo majhne, tako da nam za to ni potrebno tako skrbeti. Prediktor-korektor se omenjenima metodama tudi zelo približa, Midpoint in Heunova metoda sta za red velikosti počasnejši, Eulerjeva pa je na seznamu le za opomnik, da moramo biti hvaležni vsem ljudem, ki so sestavili boljše algoritme. Tu je tudi odgovor na vprašanje od

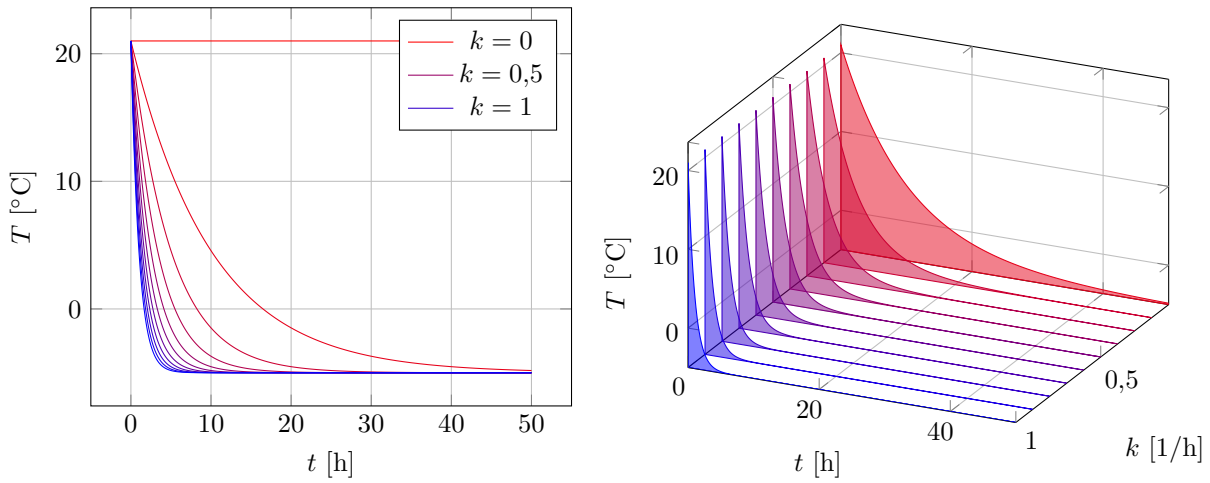
metoda	korak [h]	čas [s]	max(ΔT) [°C]
RKF	/	$3{,}80 \cdot 10^{-4}$	$9{,}28 \cdot 10^{-6}$
Euler	10^{-4}	1,53	$9{,}56 \cdot 10^{-6}$
Heun	0,055	$5{,}84 \cdot 10^{-3}$	$9{,}69 \cdot 10^{-6}$
Midpoint	0,055	$4{,}93 \cdot 10^{-3}$	$9{,}69 \cdot 10^{-6}$
PECE	1,1	$5{,}39 \cdot 10^{-4}$	$9{,}22 \cdot 10^{-6}$
RK4	1,5	$3{,}54 \cdot 10^{-4}$	$9{,}23 \cdot 10^{-6}$

Tabela 1: Čas, ki so ga različne metode porabile, da so dosegle globalno natančnost $\varepsilon = 10^{-5}\,\mathrm{K}$. Najhitrejša je metoda Runge-Kutta 4. reda, zato ni čudno, da je tako priljubljena.

prej: četudi je Eulerjeva metoda pri fiksni dolžini koraka hitrejša od ostalih, je toliko manj natančna, da je res ni vredno uporabljati.

2.3 Družina rešitev

Poglejmo si sedaj celo družino rešitev začetne enačbe pri različnih vrednostih parametra k. Za iskanje rešitev sem uporabil metodo RK4, saj je najnatančnejša in nisem časovno omejen. Za korak sem vzel $\Delta t = 0{,}1\,\mathrm{h}$. Na Sliki 5a so prikazane rešitve za $k \in [0, 1]$. Pri $k = 0$ je rešitev konstantna, z večanje k pa graf vedno hitreje pade na končno vrednost. Na Sliki 5b pa je popolnoma ista zadeva le v 3d, če se komu tako zdi lepše. Osebno mi je 2D prikaz bolj pri srcu, ampak sem v trenutku risanja ravno imel voljo prebiti nekaj ur zakopan v petsto-stranski dokumentaciji paketa Pgfplots in spreminjanju vseh možnih parametrov, tako da je nastal tudi 3D prikaz. Tipično se ravno zaradi zahtevnosti risanja 3D grafov izogibam takim prikazom, četudi občasno dodajo vsaj novo perspektivo.



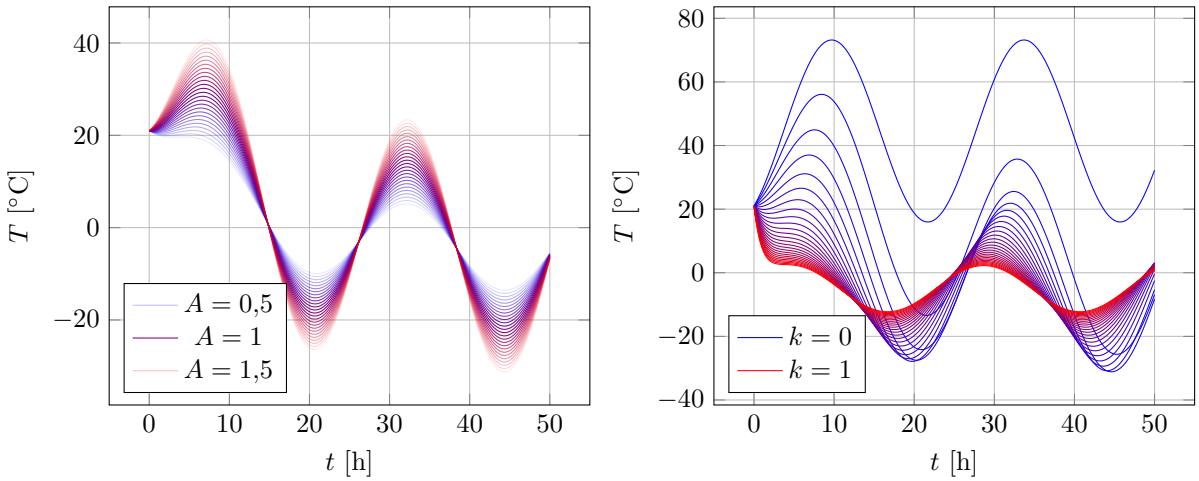
(a) Nekaj rešitev diferencialne enačbe za različne vrednosti parametra k.

(b) Malo drugačna perspektiva levega grafa.

Slika 5: Dva različna načina prikaza družine rešitev enačbe hoda pri različnih vrednostih parametra k. Pick your poison.

2.4 Dodatna naloga

Pri dodatni nalogi smo morali rešiti enačbo hoda z dodatnim členom $A\sin(\omega(t + \delta))$, z $\omega = 2\pi/24$. Za reševanje sem ponovno uporabil metodo RK4. Če bi želeli natančneje določiti lege vrhov, bi priporočal uporabo RK45 (RKF) metode, saj prilagaja dolžino koraka. Natančneje, kjer je drugi odvod (absolutno) večji, je korak manjši. Ker je v bližini vrha tipično drugi odvod velik, je korak tam majhen, torej lahko z precejšnjo natančnostjo določimo vrh, ne da bi tratili čas na mestih, kjer je graf bolj raven. Nekaj prikazov družin rešitev obravnavane enačbe je na Sliki 6a in Sliki 6b.



(a) Družina rešitev pri $k = 0{,}1/\mathrm{h}$, $A \in [1/2, 3/2]$ in $\delta = 0\,\mathrm{h}$.

(b) Družina rešitev pri $A = 1\,\mathrm{K}$, $k \in [0, 1]$ in $\delta = 0\,\mathrm{h}$.

Slika 6: Prikazi rešitev enačbe hoda z dodatnim členom $A\sin(\omega(t+\delta))$.

Pripravil sem tudi animirane prikaze različnih rešitev pri spreminjanju različnih parametrov. Animacije so v priponkah tega dokumenta, dostopne tukaj (spreminjanje A), tukaj (spreminjanje k) in tukaj (spreminjanje δ). *Nasvet*: za prvinsko izkušnjo pri uživanju v animacijah priporočam uporabo Adobe Acrobat Readerja. Pri ostalih orodjih za morebitno slabšo izkušnjo pri ogledu ne odgovarjam.

3 Zaključek

Tokratna naloga se je na prvi pogled zdela zelo zahtevna, na drugi pogled zelo preprosta, na koncu pa mi je vseeno vzela več časa, kot sem pričakoval (does anybody notice a recurring theme?). S tem imam v mislih natanko eno vrstico, ki se glasi `h = t[i + 1] - t[i]`. Nepotrebne komplikacije so se začele s tem, ko sem nekako spregledal datoteko v spletni učilnici z napisanimi vsemi pomembnimi metodami. Zato sem se vseh implementacij lotil sam. Ta del niti ni bil tako časovno potraten in je bil v resnici izredno poučen. Na videz so vse metode delovale čudovito, dokler se nisem malo bolj poglobil v natančnost pri nižanju velikosti koraka. Vsem metodam se je natančnost zmanjševala bistveno prepočasi in bistveno preveč kaotično. Zgodilo se je, da je bila natančnost pri $\Delta t = 0{,}1\,\mathrm{h}$ enaka 10^{-10}, natančnost pri $\Delta t = 0{,}09\,\mathrm{h}$ pa 10^{-3}. Po več urah precej obupavajočega iskanja napake sem ugotovil, da me je `numpy` pustil na cedilu. Čase sem namreč porazdelil s pomočjo funkcije `linspace` na primerno število enakih kosov med začetno in končno točko. Ker so koraki enako veliki (sem si mislil), lahko vzamem velikost koraka $\mathrm{d}t$ in to številko uporabim v računih. Vendar pa `linspace` ne vrne natančno enakih korakov, sem se naučil. Za željeno natančnost je potrebno v vsakem koraku izračunati dejansko velikost časovnega premika kot razliko dveh zaporednih elementov v "enakomerno" porazdeljenem seznamu časovnih točk.

Če se vrnemo k pozitivnejšim izkušnjam: našel sem inovativni način za uporabo funkcij višjih redov v Pythonu, nekoliko sem se poglobil v dokumentacijo za risanje tridimenzionalnih grafov z orodjem Pgfplots (rezultat je Slika 5b ter še 3D verzija grafa 6b, ki pa mi ni uspela) in odkril sem paket `animate`, s katerim sem sestavil tri krasne animacije (v priponki dokumenta).