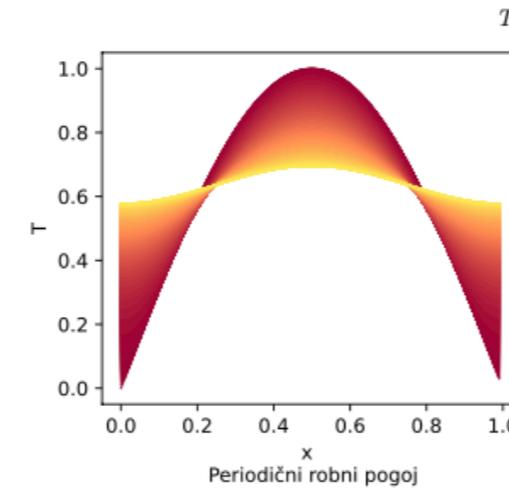
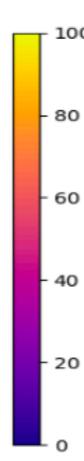
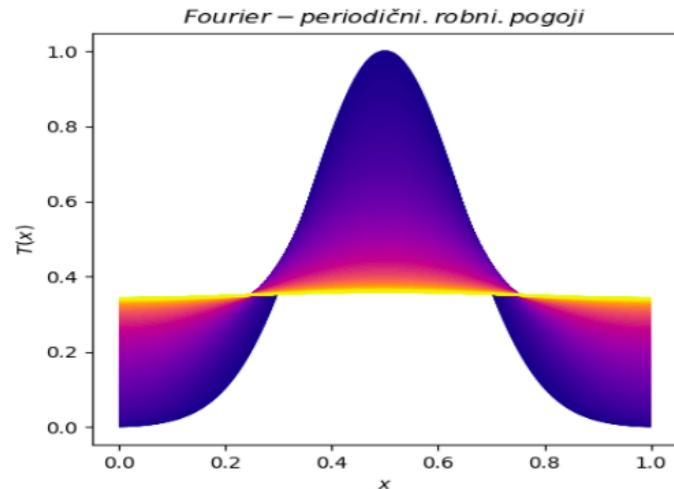


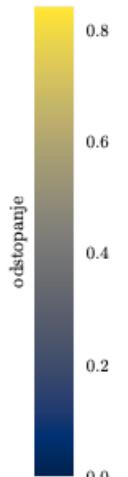
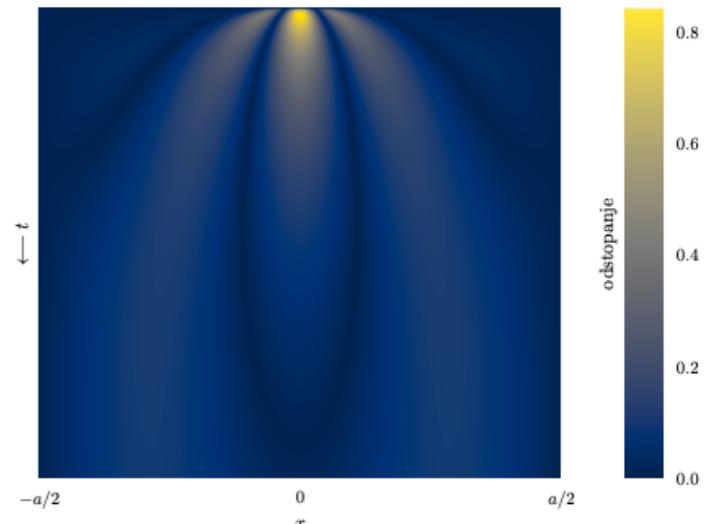
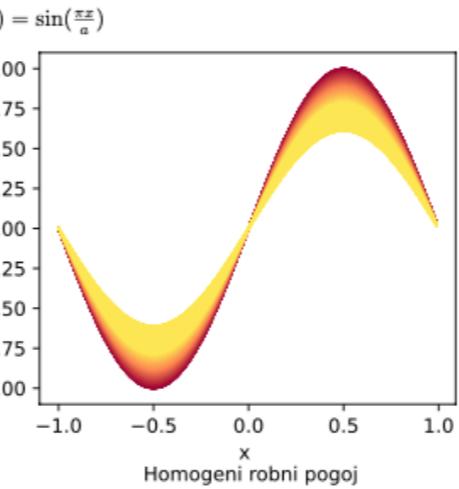


9. naloga

2022/23



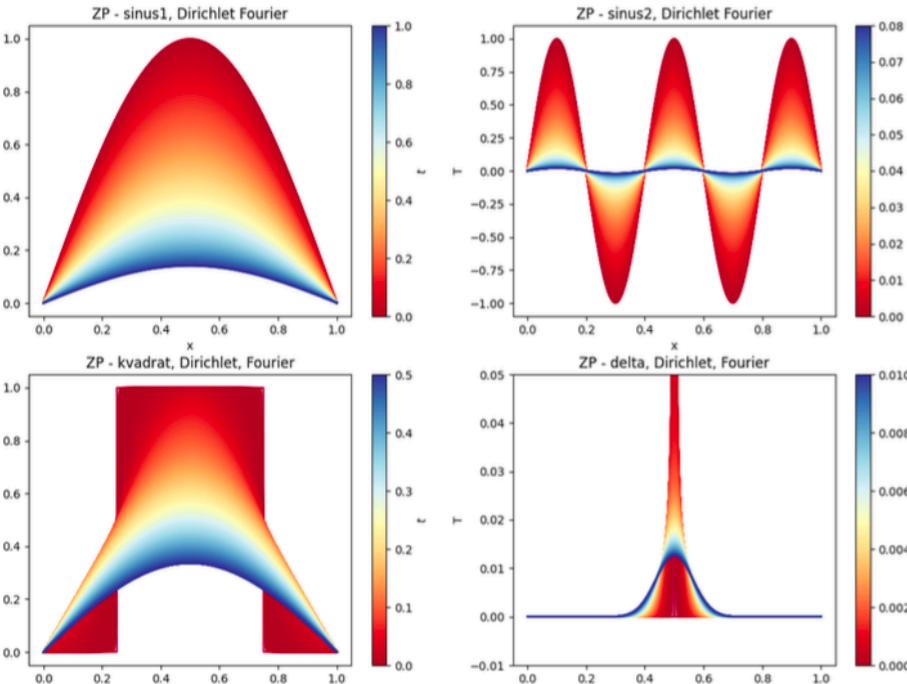
$$T(x, 0) = \sin\left(\frac{\pi x}{a}\right)$$



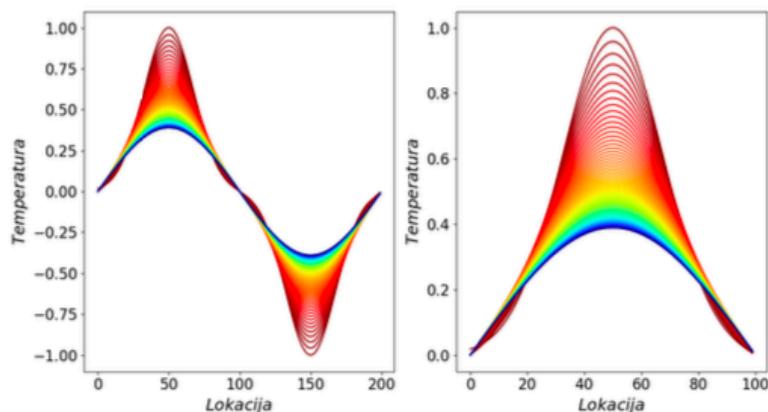
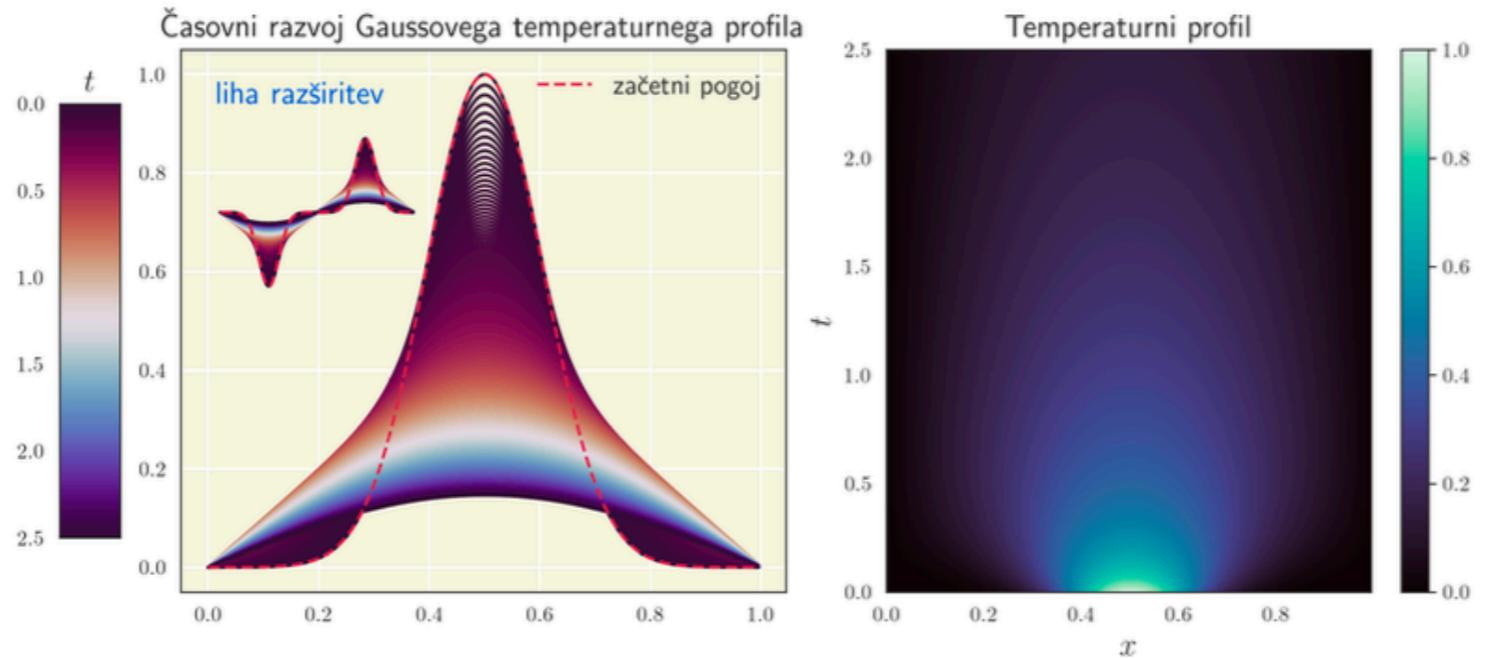


Neanimirane predstavitve

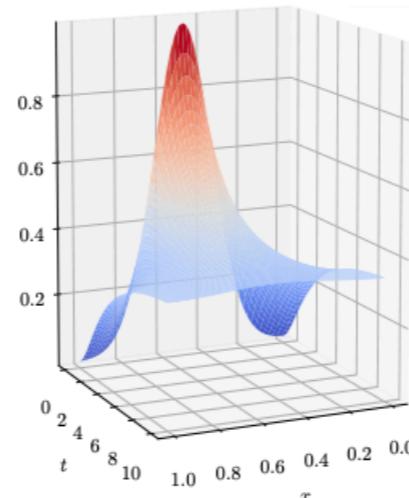
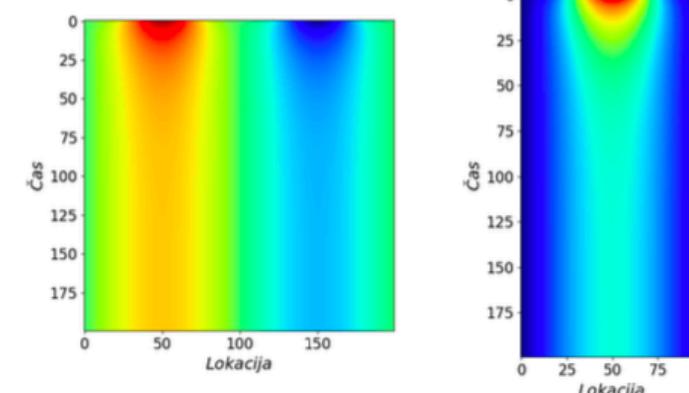
- Lahko 1D, 2D, 3D...



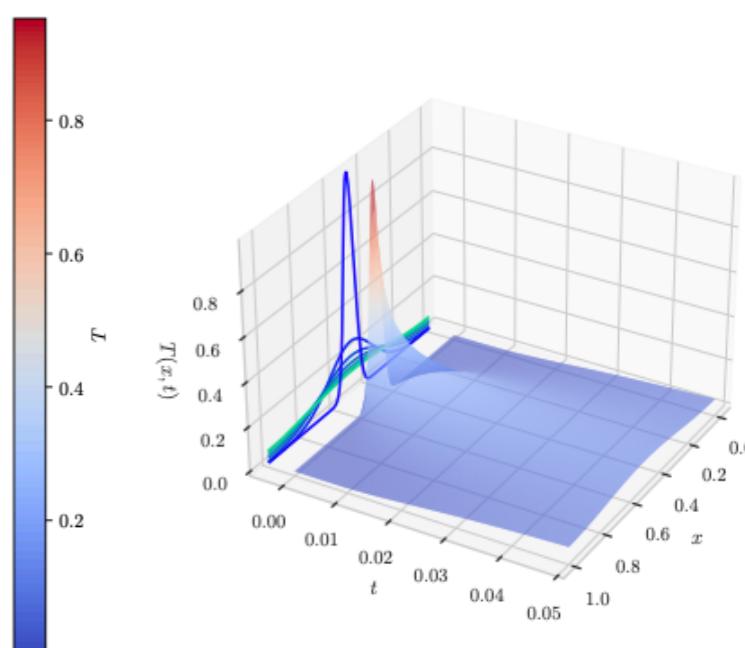
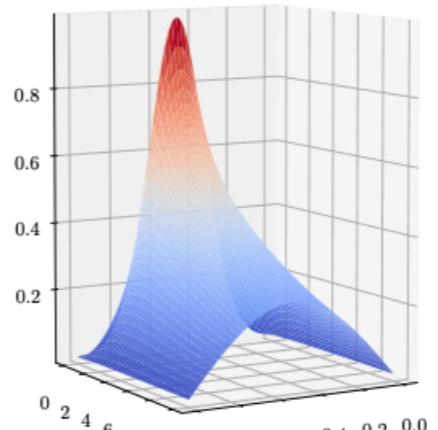
Homogen robni pogoj, $T(0, t) = T(1, t) = 0$



Periodični RP



Dirichletovi RP

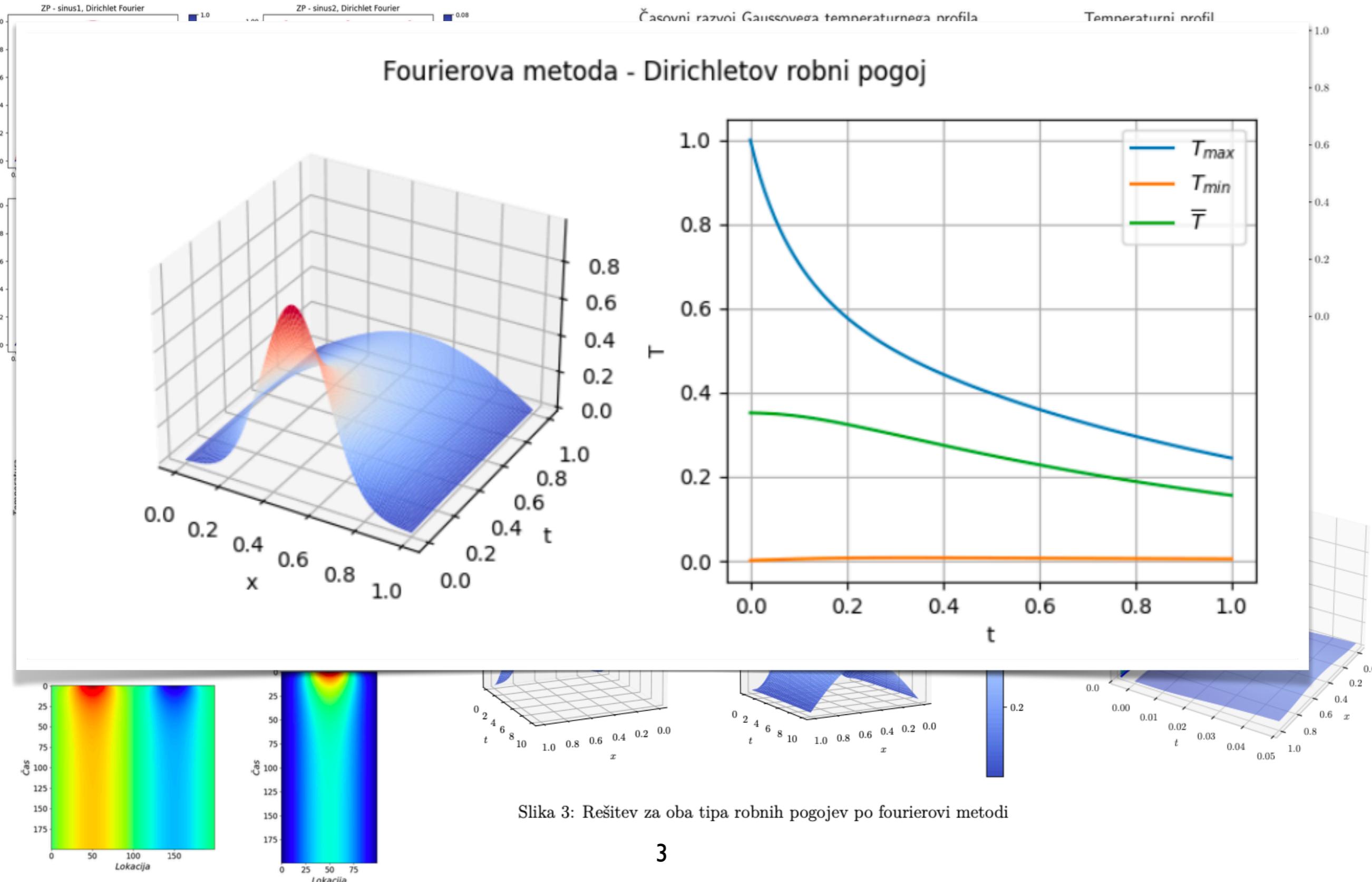


Slika 3: Rešitev za oba tipa robnih pogojev po fourierovi metodi



Neanimirane predstavitve

- Lahko 1D, 2D, 3D...

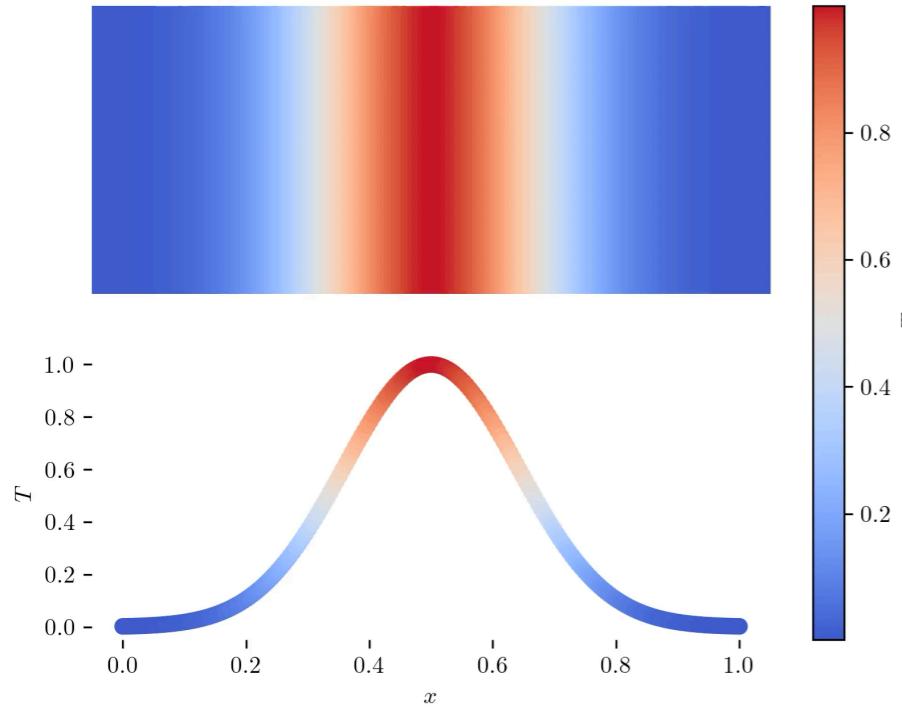




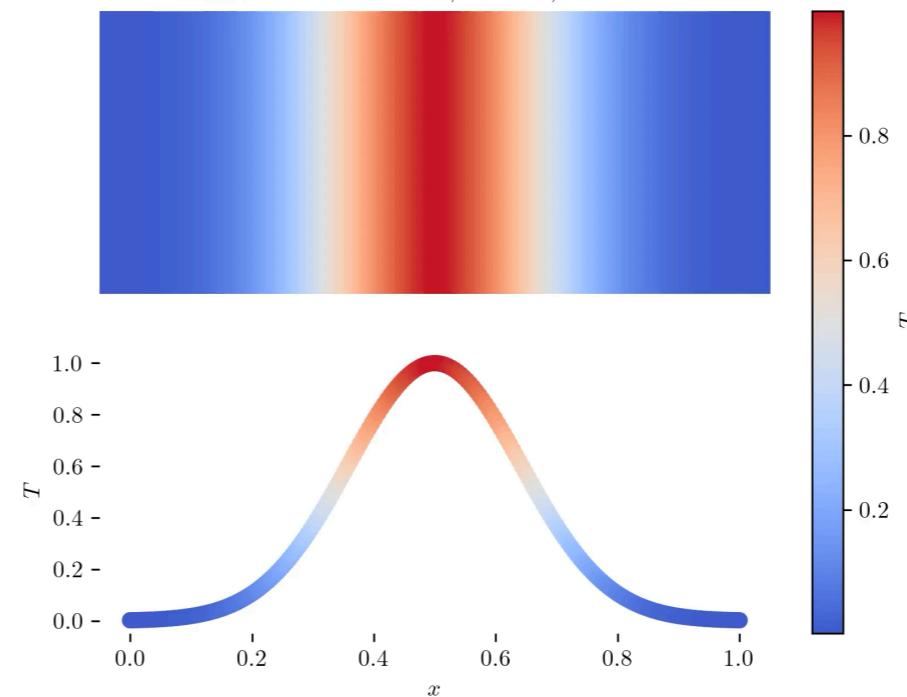
Animirane predstavitev

Lepo!

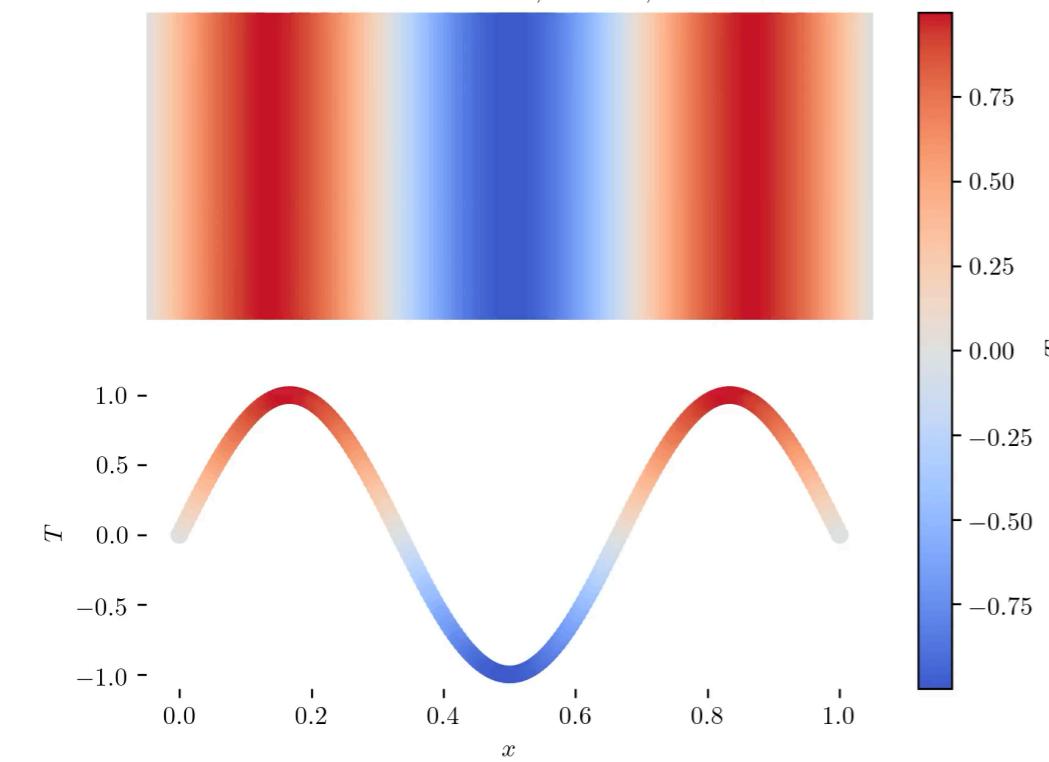
Dirichletovi : $N = 200, \sigma = 0.2, D = 0.01$



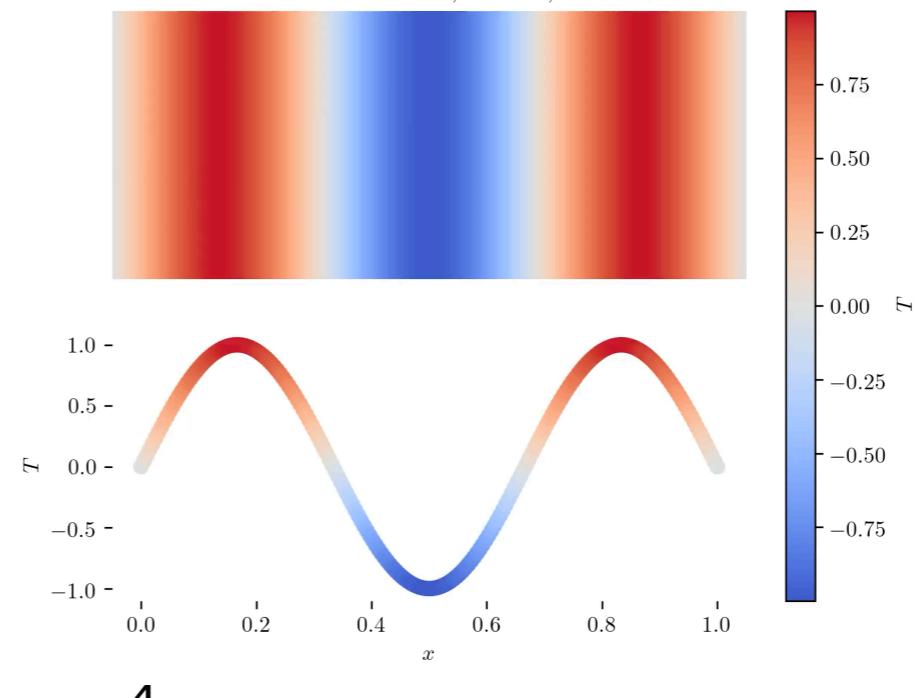
Periodični : $N = 200, \sigma = 0.2, D = 0.01$



Dirichletovi : $N = 200, \sigma = 0.2, D = 0.01$



Dirichletovi : $N = 200, \sigma = 0.2, D = 0.01$

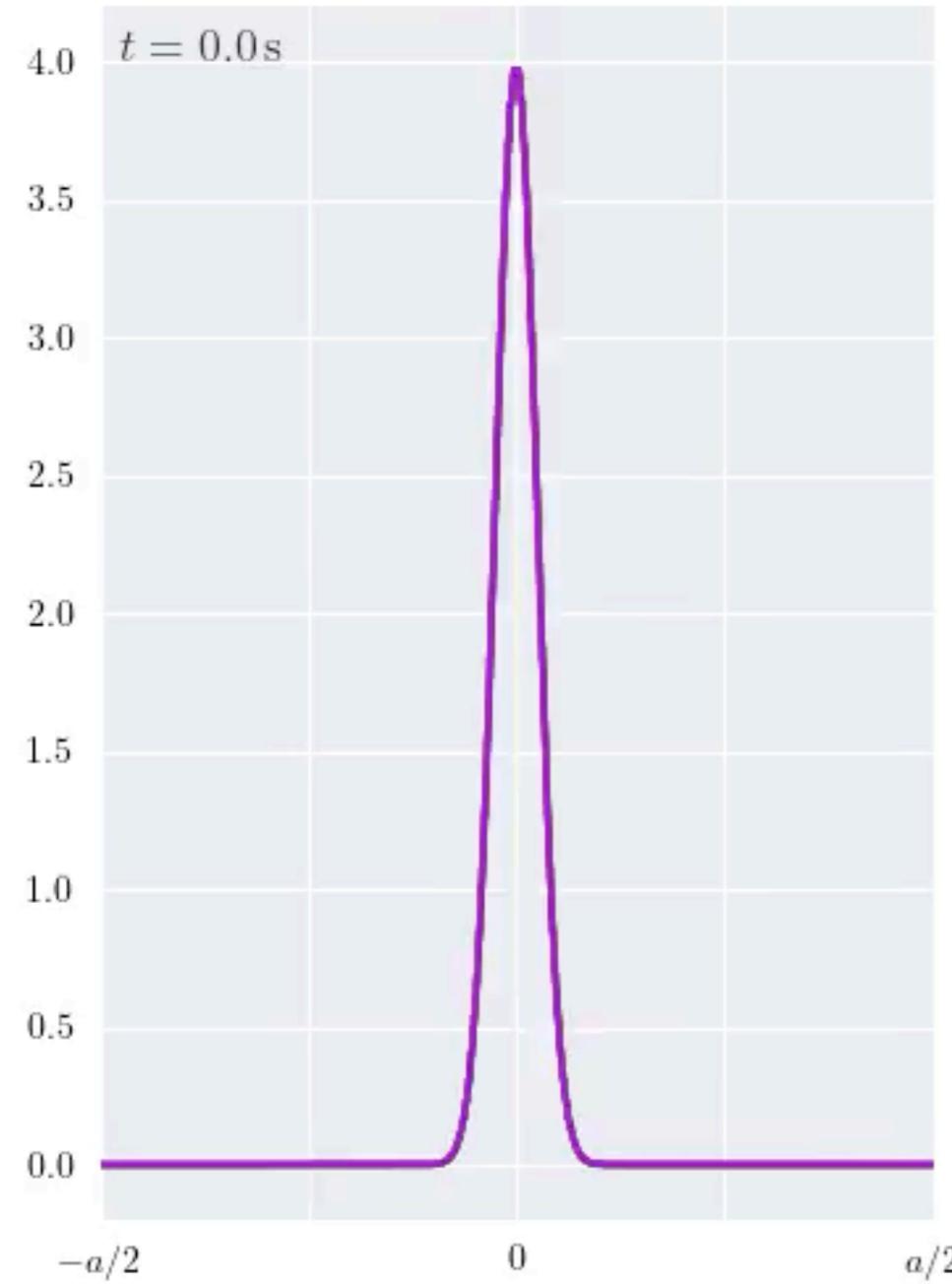




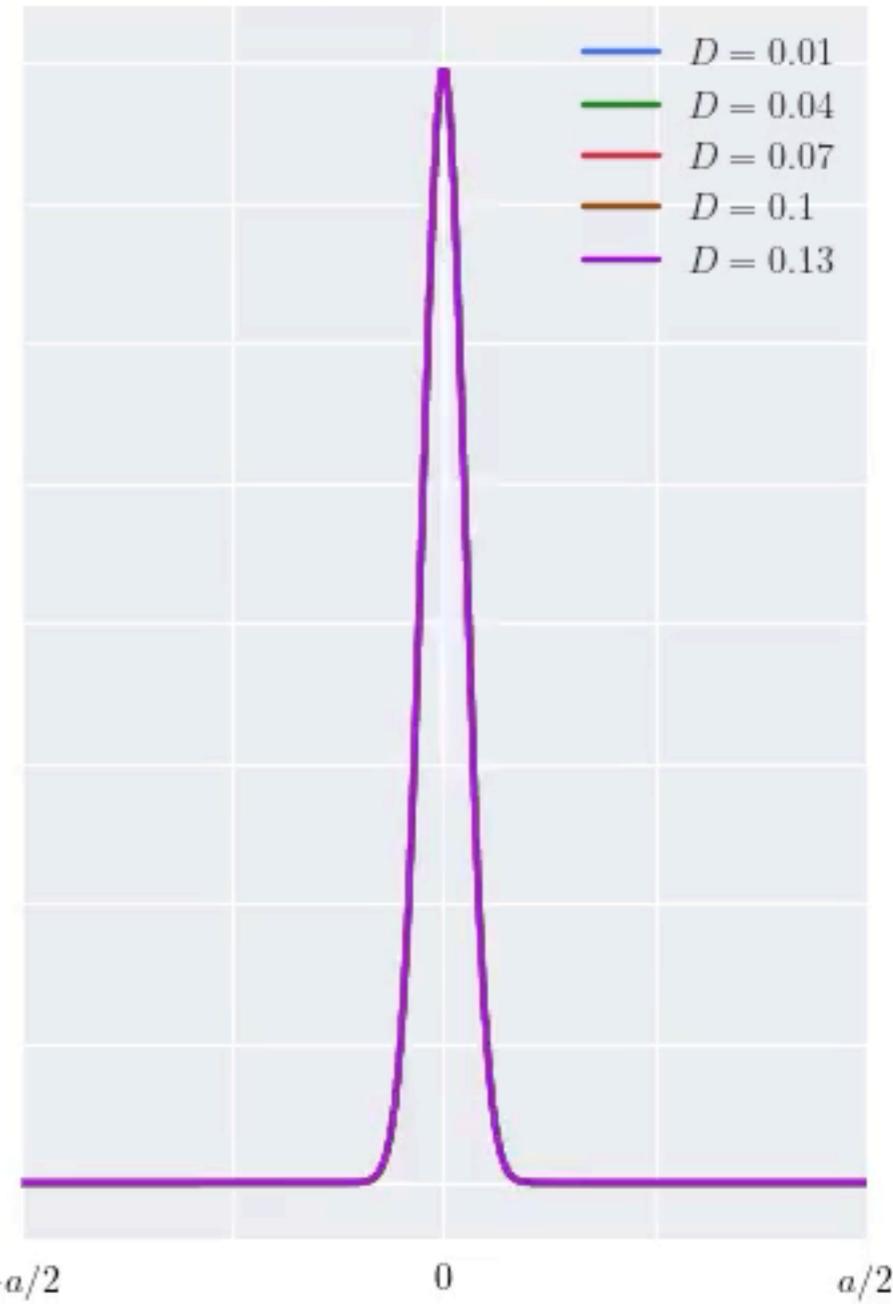
Animirane predstavitev

Lepo!

Dirichletov robni pogoj

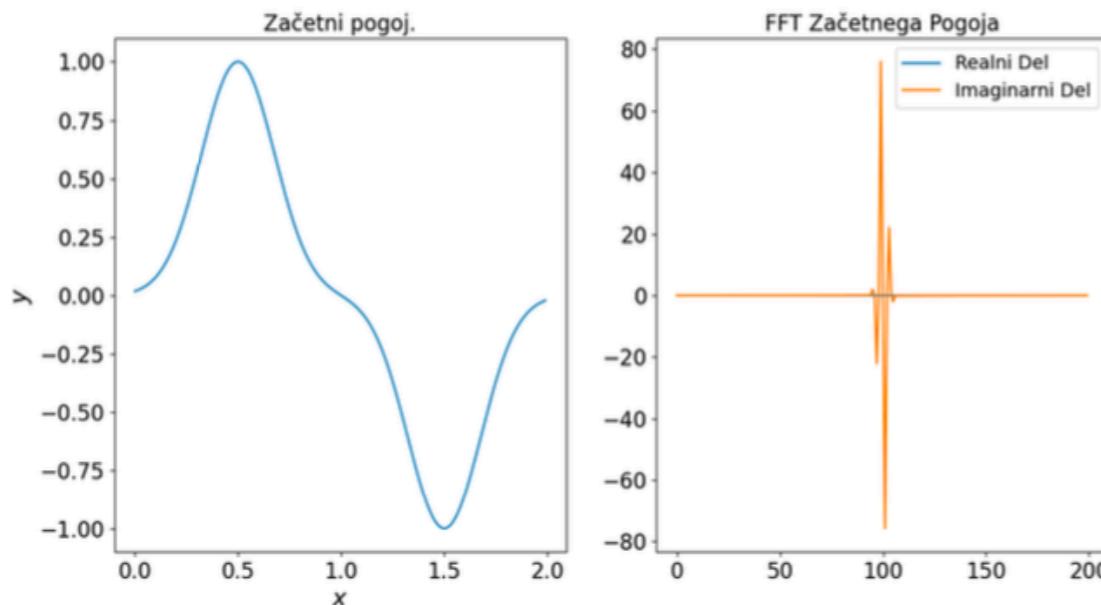


Periodični robni pogoj

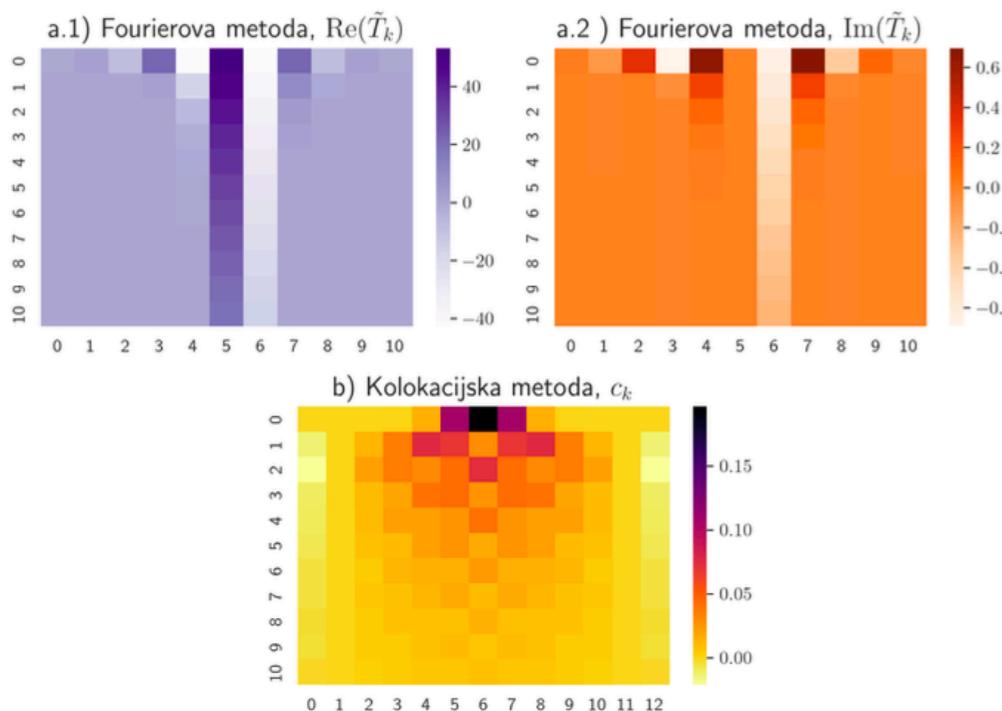




Vpogled v teorijo



Koeficienti pri spektralni metodi na mreži 11×11



Slika 8: Časovni razvoj koeficientov v razvoju po baznih funkcijah pri a) Fourierovi in b) kolokacijski metodi.

Lani - eden redkih, ki si je vzel čas za hitri račun...

Če to vstavimo v difuzijsko enačbo, dobimo diferencialne enačbe za koeficiente \tilde{T}_k , ki jih lahko analitično rešimo:

$$\frac{\partial \tilde{T}_k}{\partial t} = -4\pi^2 f_k^2 D T_k(t) \implies \tilde{T}_k(t) = \tilde{T}_k(0) e^{-4\pi^2 f_k^2 D t}, \quad (3)$$

lahko pa jih ne rešimo analitično in opazujemo, kako dobro se pravim koeficientom približamo s kakšno numerično integracijsko shemo, na primer kar po Eulerju

$$\tilde{T}_k(t+h) = (1 - 4hD\pi^2 f_k^2) \tilde{T}_k(t). \quad (4)$$

Eulerjeva metoda seveda ni stabilna za kar vsak časovni korak. V pogoj za stabilnost $|1 - 4h\pi^2 f_k^2 D| < 1$ vstavimo $h = t_{\max}/N_t$ in $f_{\max} = N_x/2a$, ker hočemo, da bo metoda stabilno delovala na vseh koeficientih \tilde{T}_k . Z N_t in N_x smo označili število točk, s katerimi smo diskretizirali časovni in krajevni interval. Dobimo, da mora biti

$$N_t > \frac{2D\pi^2 t_{\max}}{a^2} N_x^2,$$

torej moramo za stabilno rešitev izbrati časovni korak, ki se manjša s kvadratom N_x . To kvadratno odvisnost zelo hitro občutimo, ko razdelimo krajevni interval na okrog $N_x = 100$ točk, kar je povsem razumna številka, časovni interval pa moramo potem razbiti na $N_t \approx 10^4$ točk (predfaktor pred N_x^2 pride približno 1, če pustimo simulacijo teči približno tako dolgo, da pride temperaturni profil ravno blizu ravnovesja), za kar pa se tudi računalnik že nekoliko prepoti. Temu se lahko na nek

Lani ...

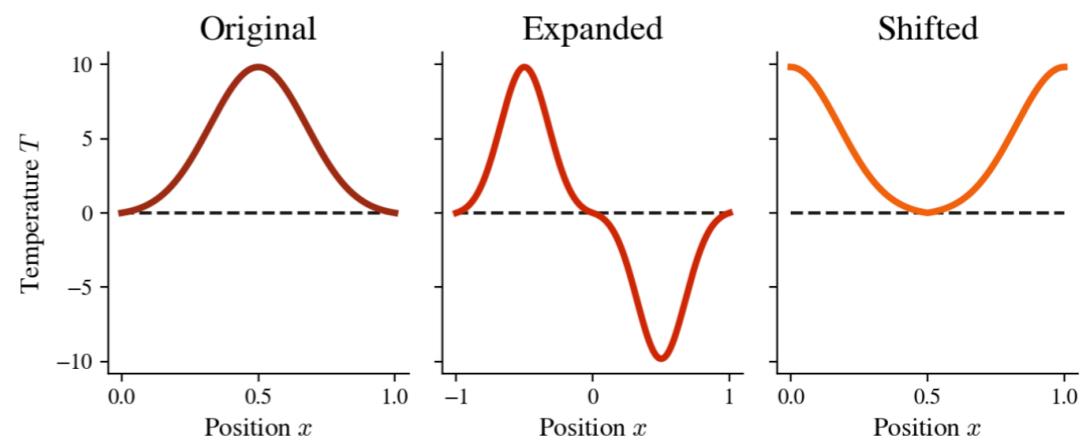


Figure 2: Modifications to the initial temperature distribution (left), needed to satisfy the homogeneous (middle) and periodic (right) boundary conditions used with the Fourier method.



Vpogled v teorijo

1.1.1 Dirichletovi robni pogoji

Najprej si poglejmo rešitev za homogene robne pogoje

$$T(0, t) = T(a, t) = 0.$$

Funkcije $e^{-2\pi i f_k x}$, po katerih razvijamo, ne izpolnjujejo robnih pogojev za vsako izbiro c_k . Če želimo zagotoviti, da bojo koeficienti c_k vedno taki, da bodo izpolnjevali robne pogoje, moramo začetne pogoje liho razširiti na mrežo $2N - 2$ točk

$$f_{2N-2-n} \equiv -f_n.$$

Izberemo $2N - 2$ točk, ker robni pogoji ne smejo biti pri razširitvi podvjeni. Zakaj za homogene robne pogoje izberemo ravno liho razširitev bomo pokazali v naslednjih nekaj vrsticah. Po definiciji je člen H_k DFT enak

$$H_k = \sum_{n=0}^{2N-3} f_n e^{-2\pi i \frac{kn}{2N-2}},$$

kar lahko razpišemo na originalni in razširjeni del

$$H_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i \frac{kn}{2N-2}} + \sum_{n=N}^{2N-3} f_n e^{-2\pi i \frac{kn}{2N-2}}$$

in upoštevamo definicijo naše razširitve

$$H_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i \frac{kn}{2N-2}} - \sum_{n=N}^{2N-3} f_{2N-2-n} e^{-2\pi i \frac{kn}{2N-2}}.$$

V drugem členu preimenujemo $n \leftrightarrow 2N - 2 - n$

$$H_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i \frac{kn}{2N-2}} - \sum_{n=1}^{N-2} f_n e^{-2\pi i \frac{k(2N-2-n)}{2N-2}},$$

kjer v eksponentu drugega člena upoštevamo, da je $k \in \mathbb{Z}$

$$H_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i \frac{kn}{2N-2}} - \sum_{n=1}^{N-2} f_n e^{2\pi i \frac{kn}{2N-2}}.$$

Ker velja $y_0 = y_{N-1} = 0$, se nič ne spremeni če ju v drugi vsoti dodamo in obe vsoti združimo, saj seštevamo po enakih indeksih

$$H_k = \sum_{n=0}^{N-1} f_n \left(e^{-2\pi i \frac{kn}{2N-2}} - e^{2\pi i \frac{kn}{2N-2}} \right). \quad (1.1.1)$$

Tu hitro vidimo da velja $H_k = -H_k^* \implies H$ je imaginaren. Hitro lahko preverimo, da velja tudi $H_k = -H_{2N-2-k}$. Inverzna transformacija je torej

$$f_n = \frac{1}{2N-2} \sum_{k=0}^{2N-3} H_k e^{2\pi i \frac{kn}{2N-2}} = \frac{1}{2N-2} \sum_{k=1}^{N-2} H_k \left[e^{2\pi i \frac{kn}{2N-2}} - e^{-2\pi i \frac{kn}{2N-2}} \right],$$

kjer lahko izrazimo

$$f_n = \frac{1}{N-1} \sum_{k=1}^{N-2} H_k i \sin \left(2\pi i \frac{kn}{2N-2} \right).$$

Lani - res temeljit razmislek ...

Od tu hitro vidimo, da je $f_0 = f_{N-1} = 0$, kar sicer že vemo. Pomembna ugotovitev te izpeljave je, da če izpolnimo pogoja $H_k = -H_k^*$ in $H_k = -H_{2N-2-k}$, bo rešitev ustrezala robnim pogojem (iz imaginarnosti H_k lahko sklepamo, da je rešitev f_k realen). Tu pa naletimo na prvo težavo z DFT. Transformiramo le točke, diferencialna enačba pa vsebuje odvode. Ker skozi izbor točk lahko napeljemo neskončno funkcij različnimi odvodi, morami poižkusiti izbrati najbolj ustrezne. Če je izbor točk vrednosti začetnega pogoja dovolj dober - funkcija ne divja med dvema vzorčenjema (to je težko natančno definirati, nek poižkus definicije bi bil, da se mora funkcija med vzorčenji dovolj dobro ujemati s Taylorjevim razvojem do drugega člena) je najbolj ustrezan izbor z najmanjšimi frekvencami. Če funkcija ni dobro opisana pride seveda lahko do velikih napak v izračunu, npr. vzorec zgreši špico funkcije, kar povsem spremeni obnašanje. Za izračun časovnega razvoja izberemo torej set frekvenc

$$\{f_k\} = \begin{cases} \{0, 1, \dots, \frac{N}{2}-1, -\frac{N}{2}, \dots, -1\}/(\Delta x \cdot N), & N \text{ sod}, \\ \{0, 1, \dots, \frac{N-1}{2}, -\frac{N-1}{2}, \dots, -1\}/(\Delta x \cdot N), & N \text{ lih}, \end{cases}$$

kjer je N število točk vzorca. Tak izbor je zelo pomemben tudi, da se pri propagiranju razvoja skozi čas ohranja pogoj $H_k = -H_{2N-2-k}$, saj ob taki izbiri velja $f_k = -f_{2N-2-k}$, kjer je minus na časovni razvoj ne vpliva, saj nastopa le kvadrat frekvence. Člen časovnega razvoja je seveda realen za vsako izbiro frekvence, kar nam ohranja tudi drug pogoj $H_k = -H_k^*$. Ker je število vzorčnih točk pri lihi razširitvi vedno sodo, moramo preveriti še ali lahko povzročata probleme frekvenci f_0 in f_{N-1} , ki nimata para. Če vstavimo $k = 0$ in $k = N - 1$ v enačbo 1.1.1 dobimo

$$H_0 = \sum_{n=0}^{N-1} f_n (1 - 1) = 0,$$

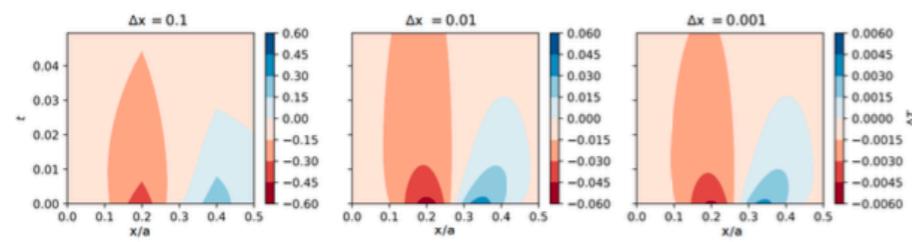
$$H_{N-1} = \sum_{n=0}^{N-1} f_n ((-1)^n - (-1)^{-n}) = 0.$$

Ker sta oba člena ob začetnem času enaka 0, bosta to tudi ob vsakem drugem času in z njima ne bo težav. Ko smo našli postopek za reševanje pri homogenih robnih pogojih, pa lahko problem posplošimo na poljubne Dirichletove robne pogoje

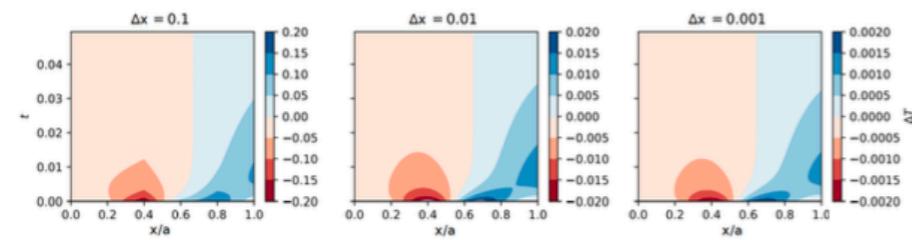


Študije napak, stabilnost...

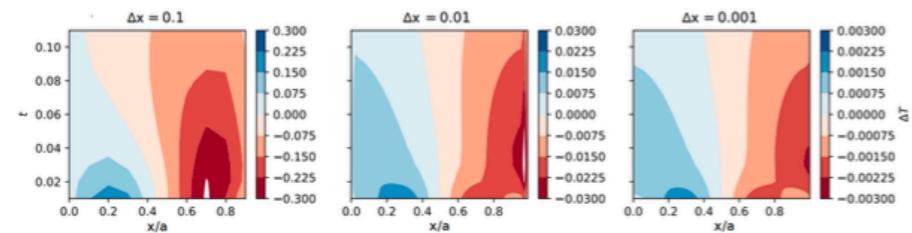
Lepo!



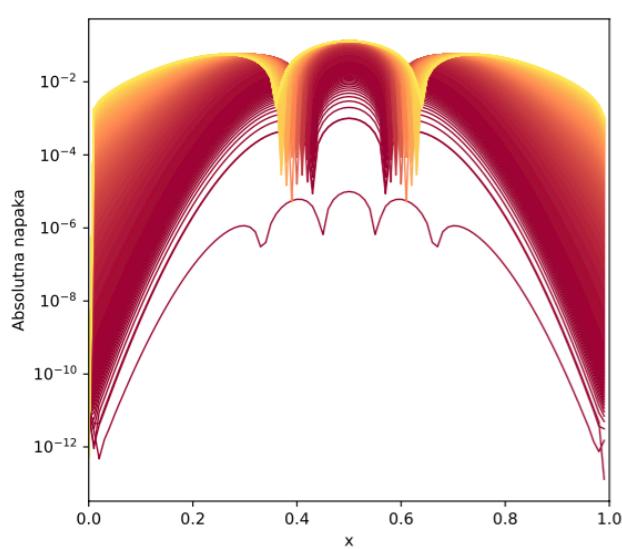
Slika 3: Lokalno odstopanje difuzije Gaussovega profila pri periodičnem robnem pogoju.



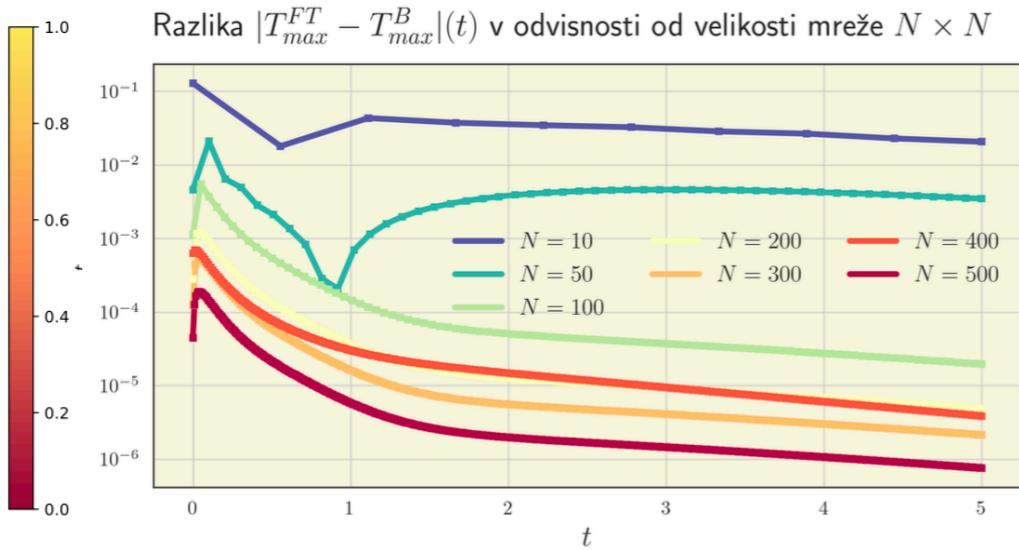
Slika 4: Lokalno odstopanje difuzije Gaussovega profila pri homogenem robnem pogoju.



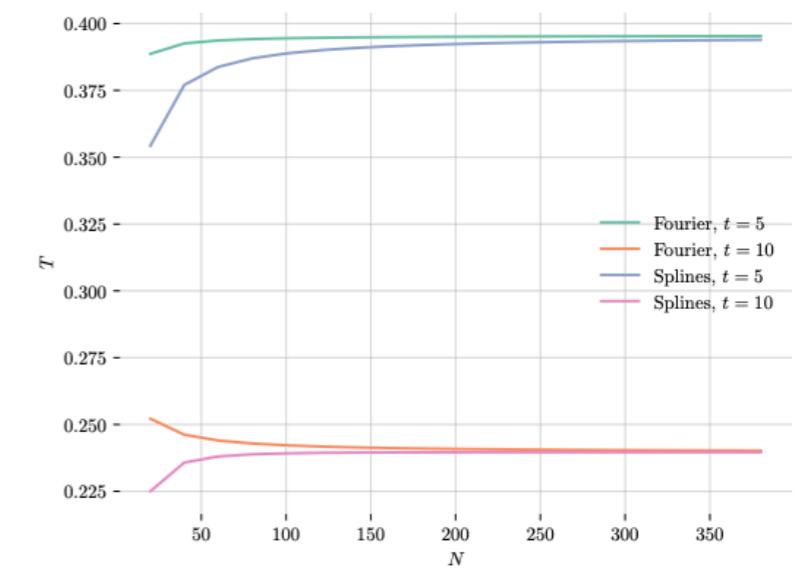
Slika 5: Lokalna napaka profilov po kolokacijski metodi.



Slika 3: Absolutna napaka kolokacijske metode.



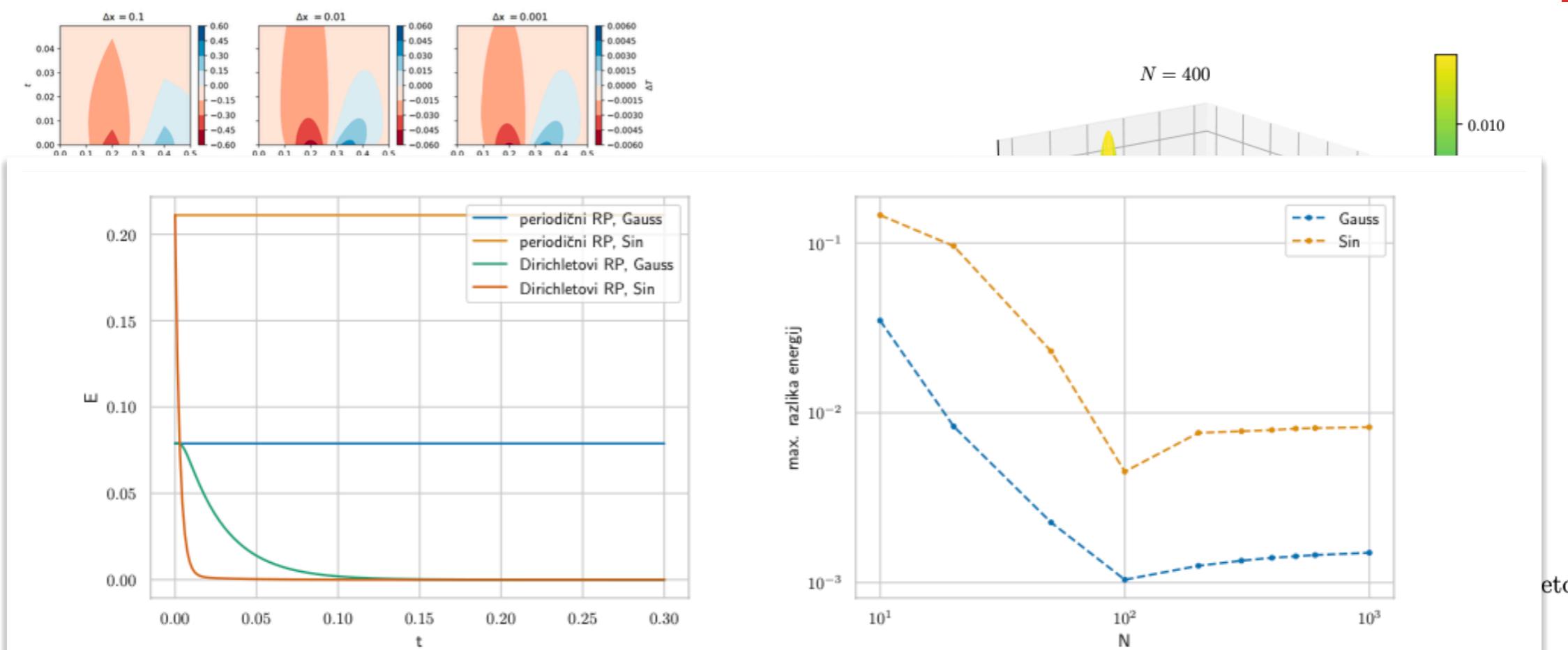
Slika 6: Razlika med rešitvijo po Fourierovi in kolokacijski metodi



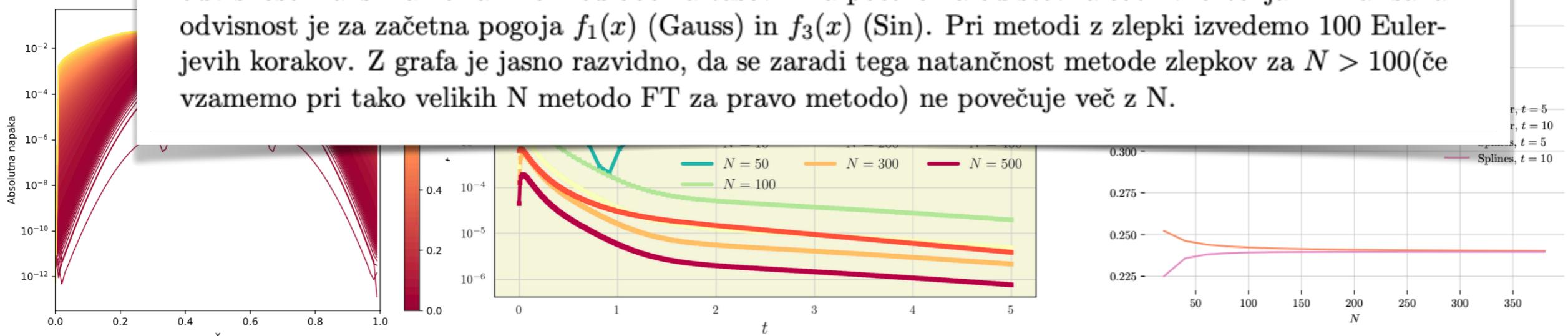
Slika 8: Stabilnost temperature sredinske točke pri dveh različnih časih, za obe metodi

Študije napak, stabilnost...

Lepo!



Slika 3: Časovni potek energije v odvisnosti od časa (levo). Časovni potek energije pri Dirichletovih robnih pogojih izračunamo tako z metodo zlepkov kot s Fourierovo transformacijo. Na desni prikažemo odvisnost maksimalne razlike med obema časovnima potekoma od števila točk vzorčenja N . Narisana odvisnost je za začetna pogoja $f_1(x)$ (Gauss) in $f_3(x)$ (Sin). Pri metodi z zlepki izvedemo 100 Eulerjevih korakov. Z grafa je jasno razvidno, da se zaradi tega natančnost metode zlepkov za $N > 100$ (če vzamemo pri tako velikih N metodo FT za pravo metodo) ne povečuje več z N .



Slika 3: Absolutna napaka kolokacijske metode.

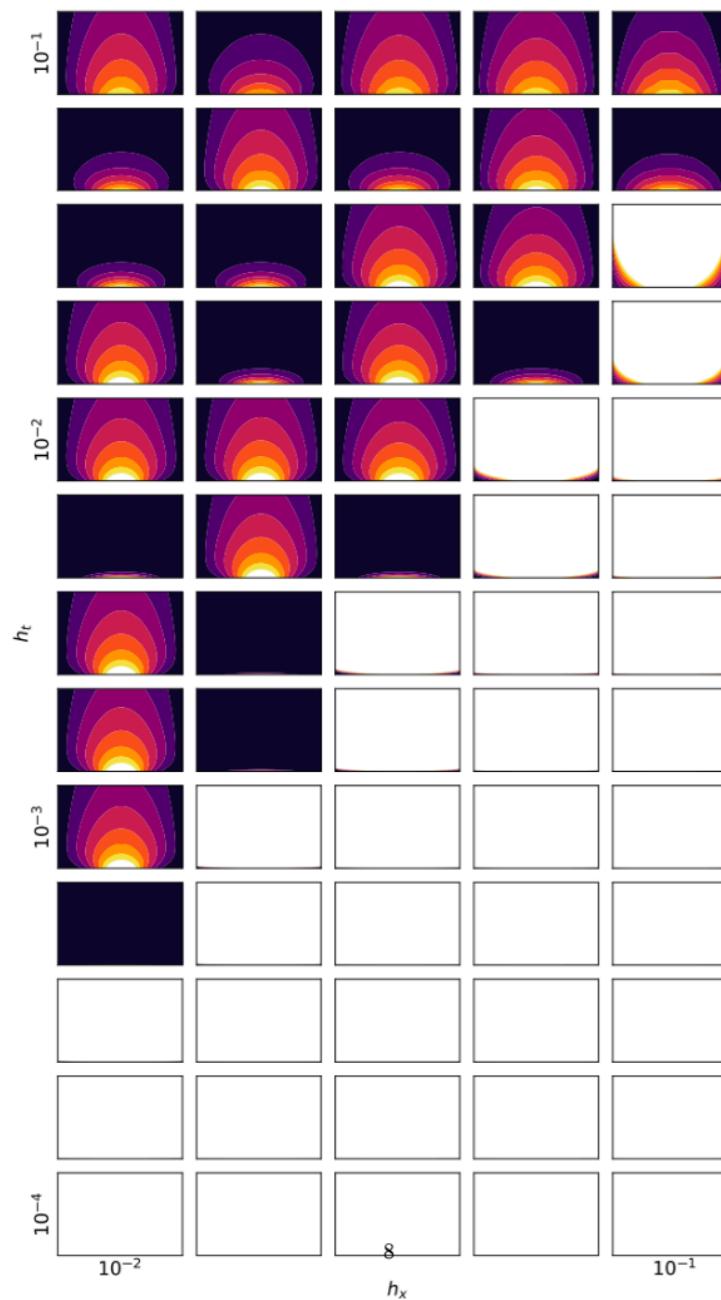
Slika 8: Stabilnost temperature sredinske točke pri dveh različnih časih, za obe metodi



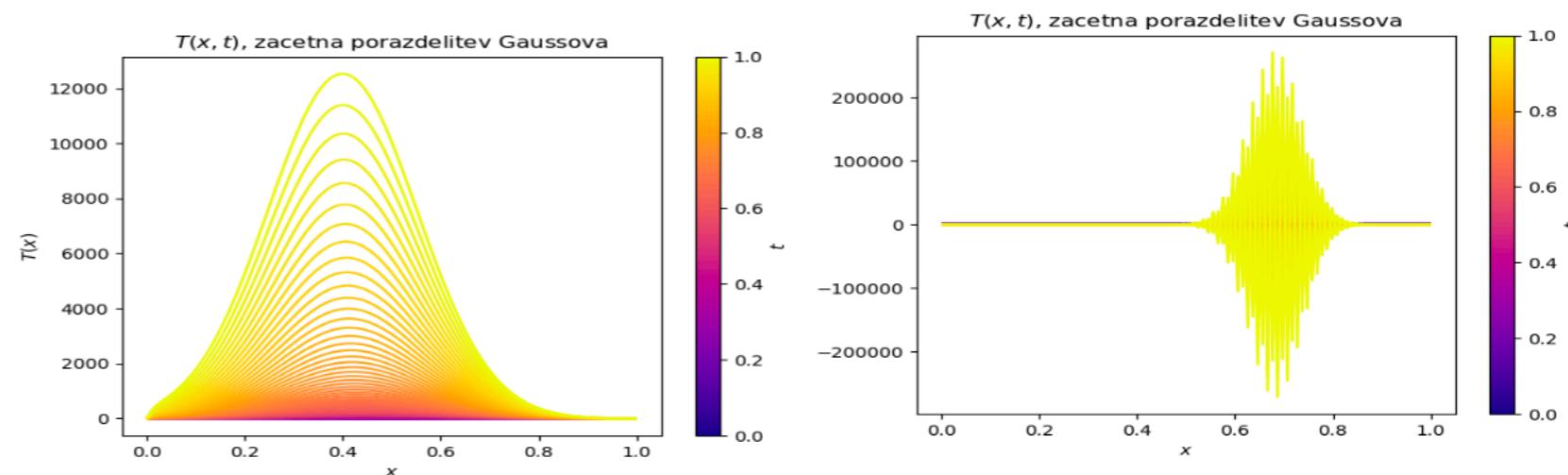
Študije napak, stabilnost...

Enaka analiza stabilnosti kot v prejšnjem poglavju praznično vzdušje žal hitro pokvari (slika 6). Oblike rešitve se sicer zdijo pravilne, a je hitrost difuzije navidez povsem kaotično odvisna od dolžine obeh korakov. Vsi poskusi določitve napake so se izkazali za jalove, iskanje pa je nedvomno oteževalo pomajkanje vsakršnega trenda ali vsaj vzorca. Poglobljena primerjava metod je tako nesmiselna, vseeno pa lahko zberemo nekaj opažanj. Tudi tu ima stabilno-

Pri metodi končnih elementov s kubičnimi B-zlepki sem naletel na nekoliko več težav. Ob nerodni izbiri parametrov so bili rezultati precej drugačni od pričakovanih. V nadaljevanju prikazujem nekaj tipičnih rezultatov, ki sem jih dobival ob nerodni izbiri parametrov. Na sliki 5 vidimo, da oscilacije pri rezultatih podivljajo. Na sliki 6 opazimo da se naša palica v resnici segreva, temperturni maksimum pa se pri tem pomika v levo. V obeh primerih gre za očitno nefizikalno obnašanje.

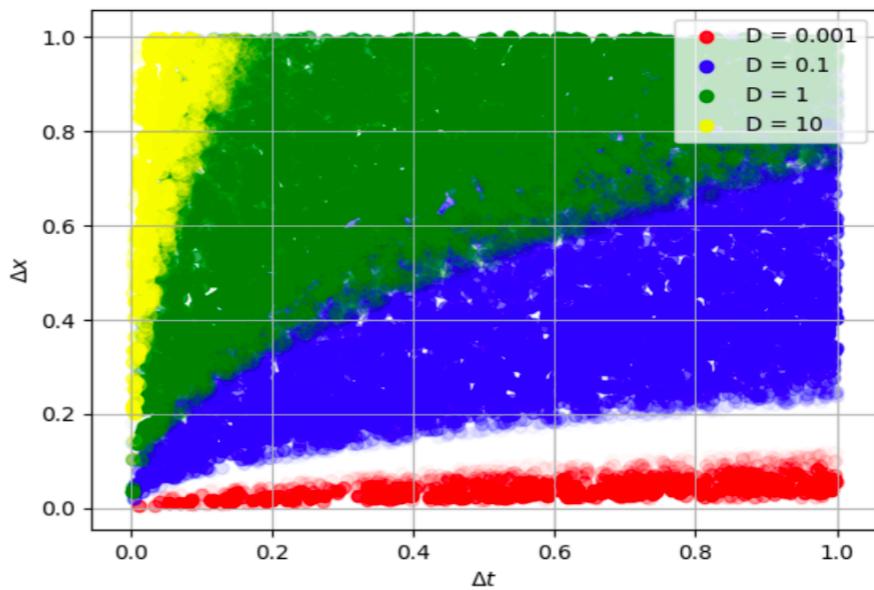


Slika 6: Difuzija Gaussovega profila izračunana po kolokacijski metodi.



Slika 6: B - zlepki . Težave 2

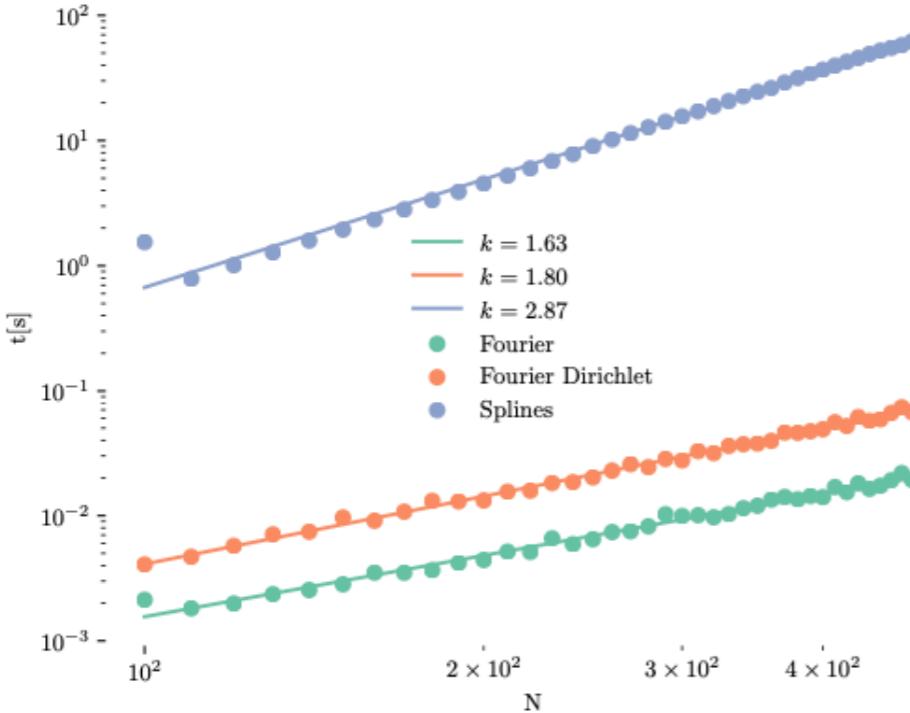
Lani ...



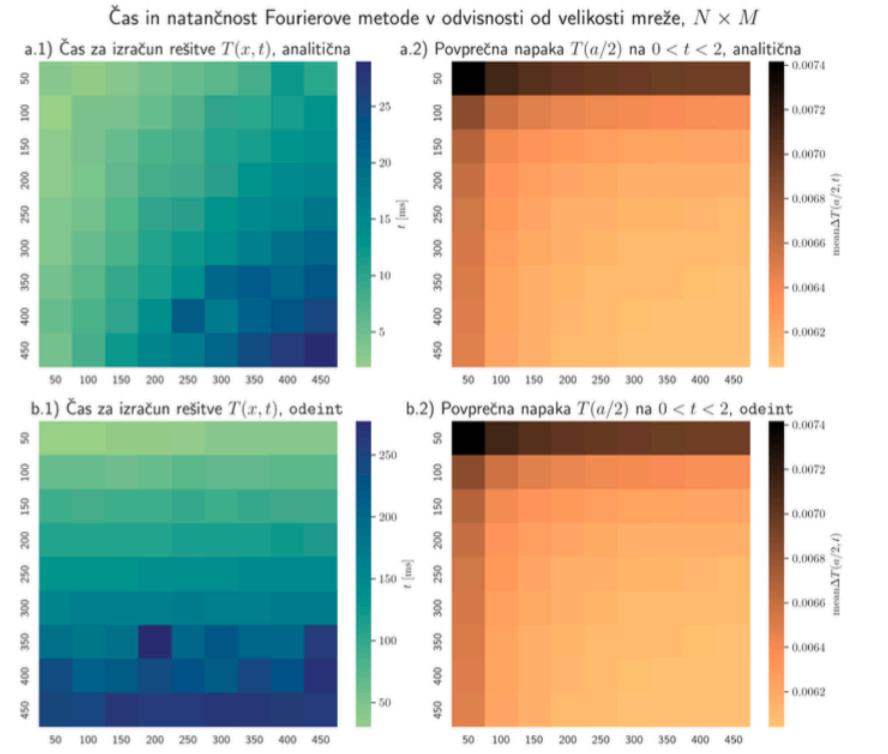
Slika 7: Stabilnostna območja za metodo B-zlepkov pri različnih vrednostih parametra D

Časovna zahtevnost..

Lepo!

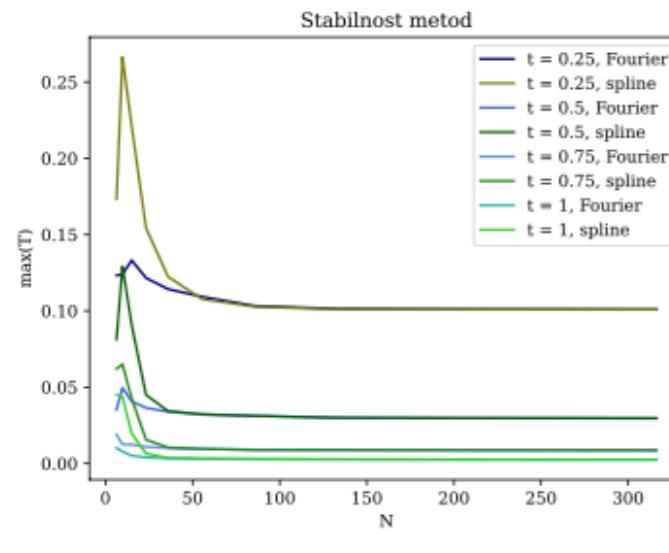


Slika 7: Časovna zahtevnost obeh metod

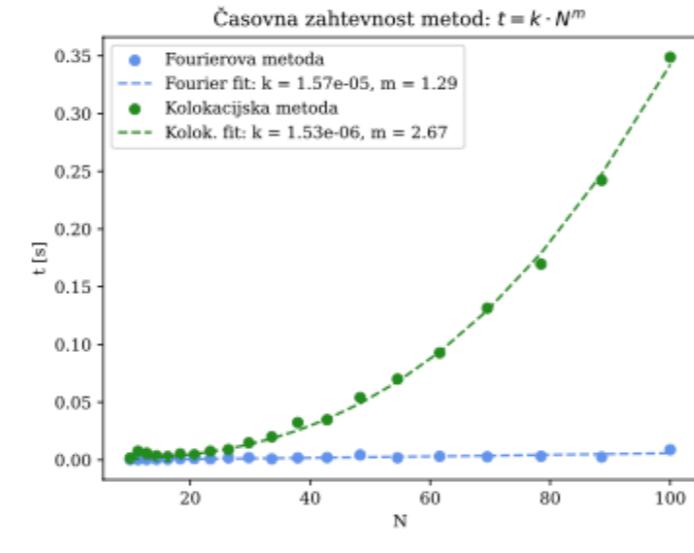


Slika 1: Časovna zahtevnost in natančnost Fourierove metode, če koeficiente izračunamo po analitični rešitvi ali pa z vgrajeno metodo `tt` scipy.odeint, v odvisnosti od velikosti mreže $N \times M$, kjer je M število časovnih točk in N število krajevnih točk.

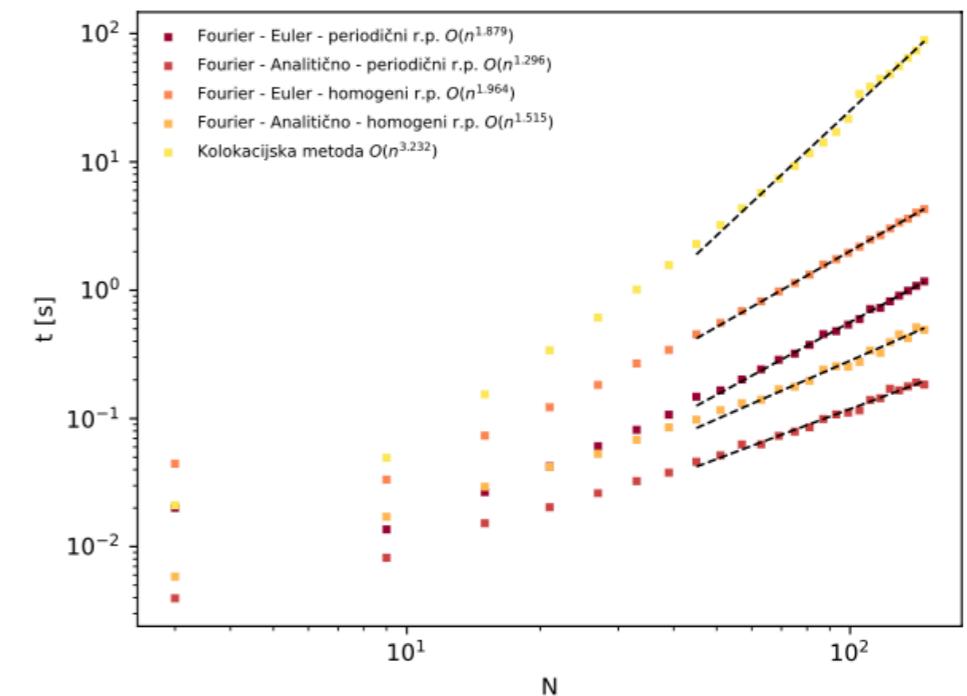
FFT zmaga - verjetno še kaj rezerve s 'sparse' algoritmi (?)



(a) Max. temp. v odvisnosti od št. točk N.



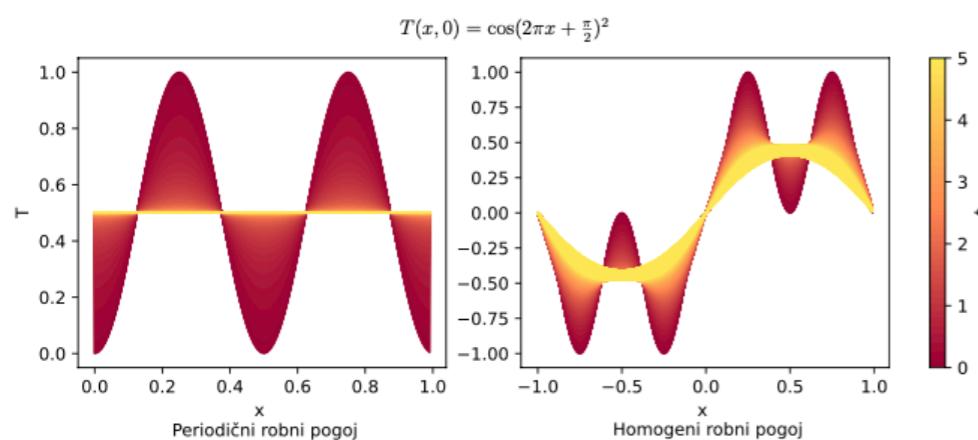
(b) Časovna zahtevnost.



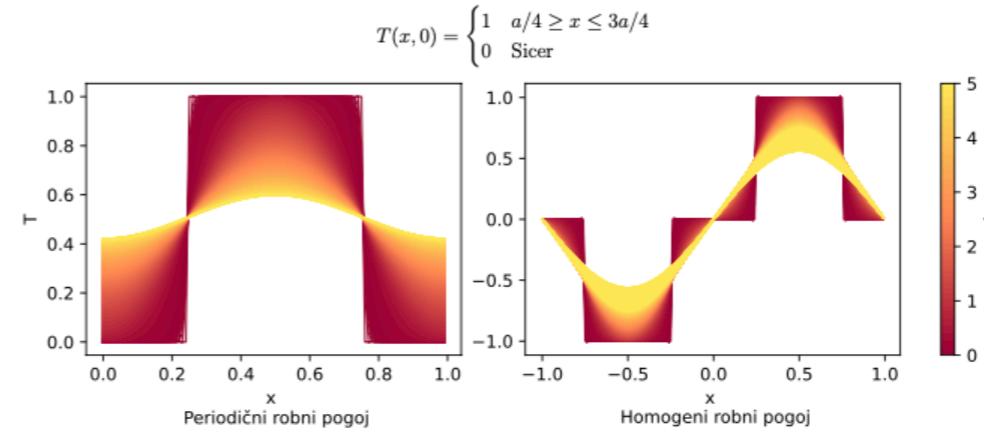
Slika 4: Časovne zahtevnosti metod z različnimi robnimi pogoji.



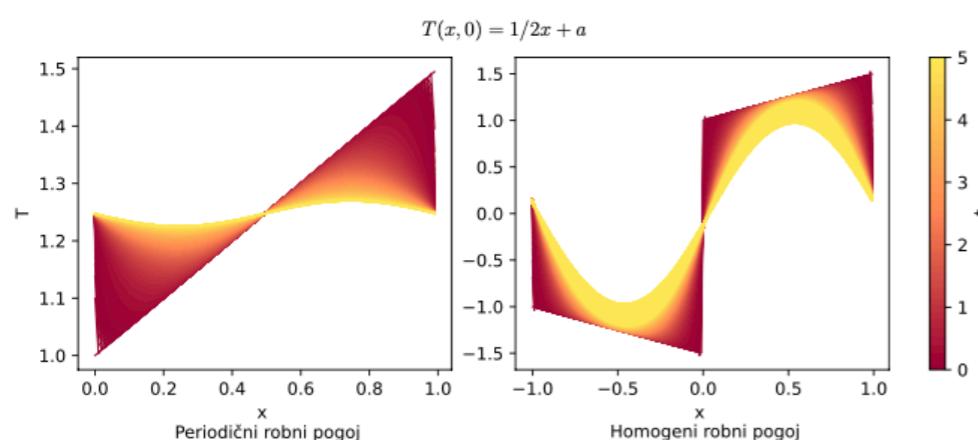
Dodatne študije



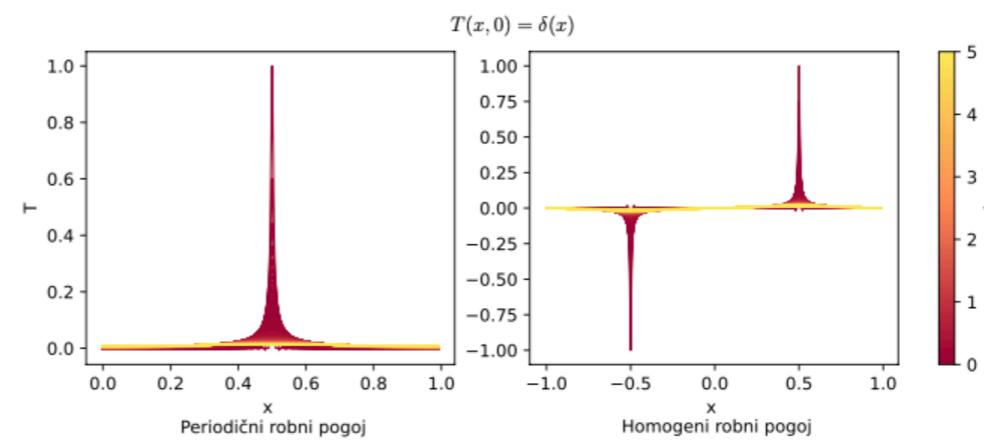
Slika 6: Kvadrirani kosinusni začetni pogoj.



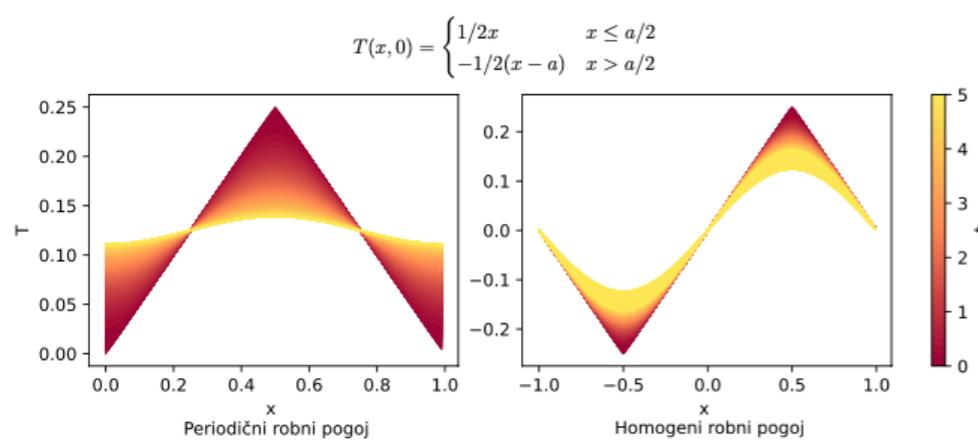
Slika 9: Kvadratni začetni pogoj.



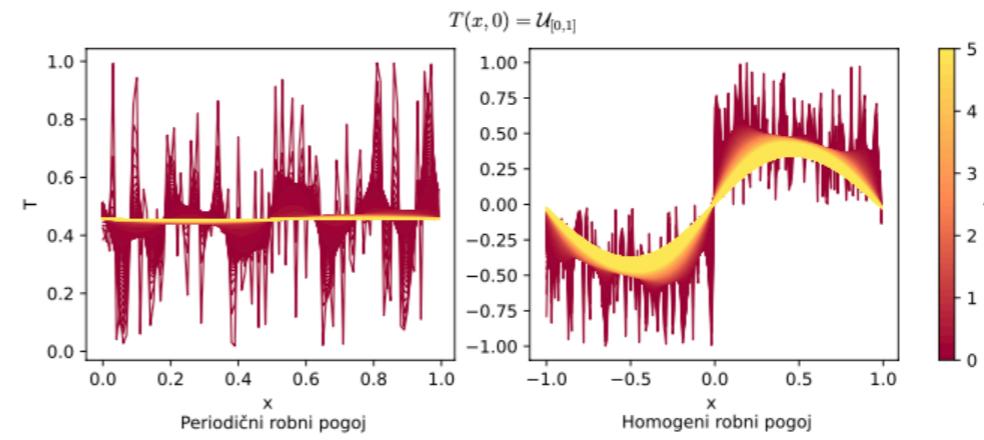
Slika 7: Linearni začetni pogoj.



Slika 10: Delta začetni pogoj.



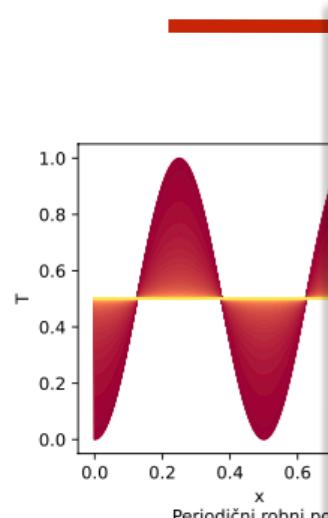
Slika 8: Trikotni začetni pogoj.



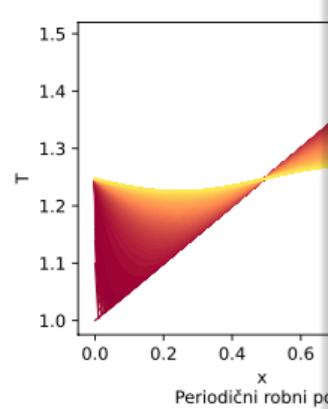
Slika 11: Naključno nenakomerno porazdeljene vrednosti kot začetni pogoj.



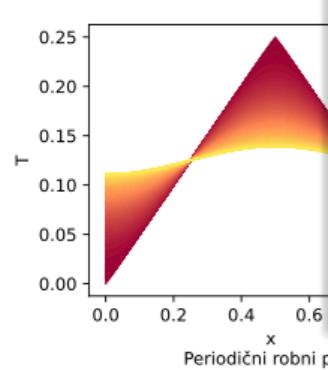
Dodatne študije



Slika 6:

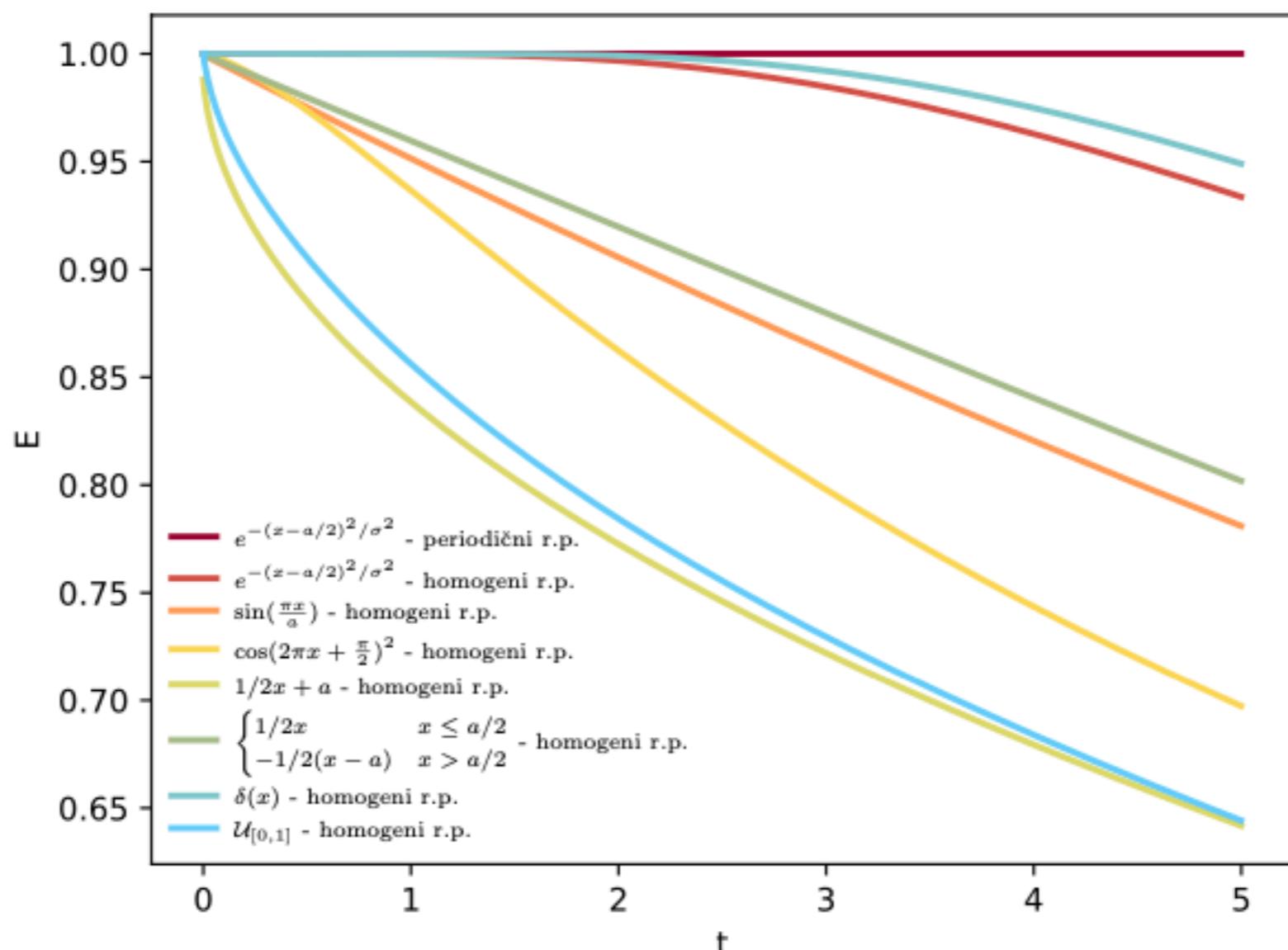


Slika 7:



Slika 8: Trikotni začetni pogoj.

Primerjava metod



Slika 12: Ohranitev energije.

Periodični robni pogoj

Homogeni robni pogoj

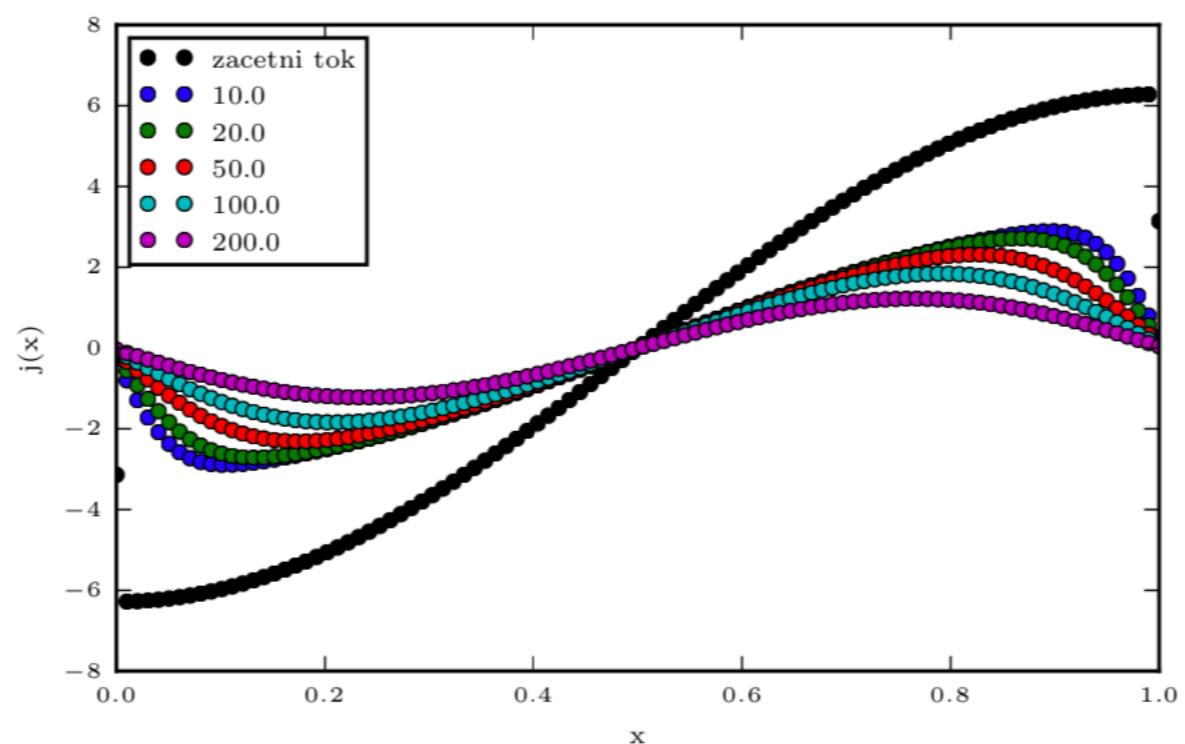
Slika 11: Naključno nenakomerno porazdeljene vrednosti kot začetni pogoj.



Fizikalni pogled

Lani...

Temperaturni profil palice se spreminja zaradi temperaturne nehomogenosti po palici. Ta povzroči nastanek topotnega toka. Ta teče iz predelov z višjo temperaturo proti predelom z nižjo. Energija prvih se torej zmanjšuje - se ohlajajo, drugih pa povečuje - ti se segrevajo. Velikost topotnega toka je odvisna od gradienca temperature, torej kako velika je temperaturna razlika v sosednjih predelih palice. Poglejmo si kako se spreminja topotni tok po palici s časom. Vrednost odvoda v posamezni točki izračunamo s pomočjo vrednosti v sosednjih dveh točkah. Pri robnih točkah za izračun uporabimo kar sami robni točki.





Razmišljanja

Lani...

V tej nalogi sem tudi ugotovil, da je Matlab veliko hitrejši od Python-a.

What language is MATLAB written in?

June 9th, 2012 | Categories: matlab | Tags:

As the resident MATLAB support guy (among other things) at [The University of Manchester](#), I often get asked which language MATLAB is written in. Many people seem to argue over whether or not it is Fortran or C with the prevailing wisdom being 'It was originally written in Fortran for the free version 1 and then was re-written in C when it went commercial'.

Well, I am not sure about the history but looking at modern MATLAB, the question should be which **languages**, rather than which **language!** The list I have so far is

- C (many built in compiled mex functions are written in C)
- C++ (MATLAB makes use of the Boost C++ libraries..thanks to M in the comments for this one)
- CIL ([Common Intermediate Language](#), used to be called MSIL, – The windows version of MATLAB uses this for various .NET stuff. Thanks to 'pmcs' in the comments for this one)
- NVidia CUDA (Routines in the GPU part of the parallel computing toolbox)
- Bash shell script (On Linux, the startup 'binary' is a shell script)
- Fortran (MATLAB uses the MKL and I'm fairly sure this is written in Fortran)
- Java (Many of the ticks in [Yair Altman's](#) excellent book, [Undocumented Secrets of MATLAB-Java Programming](#) make use of the Java elements in MATLAB)
- MATLAB (many MATLAB functions are written in .m code)
- Perl (Many mex-related scripts are written in Perl)
- Windows batch files (I've seen some simple .bat files as part of the mex machinery)

- **Res je, Python ni najhitrejši za numeriko:**

- Hitrejši je C++, ki je uporabljen v ozadju mnogih knjižnic (Boost),
- še hitrejši za numerične izračune je C,
- ... in najhitrejši je FORTRAN (skoraj vedno).

- **Tako da, če potrebujete res hitro kodo, nivo bolečine naraste...**

- Obstajajo tudi Python moduli, kot je Numba (<https://numba.pydata.org>)
- **Je pa zelo preprost za uporabo, in zato še vedno 'frontend' tudi za najzahtevnejša orodja (tudi ML - TensorFlow, PyTorch ...) - zato tudi moja izbira...**



Numba makes Python code fast

Numba is an open source JIT compiler that translates a subset of Python and NumPy code into fast machine code.



Zaključek

4 Zaključek

Vesel božič :)