Numerička rješenja Burgersove jednadžbe u jednoj dimenziji

Pavičić Donkić, Mihaela

Sveučilište u Splitu
Prirodoslovno – matematički fakultet

Numerička rješenja Burgersove jednadžbe u jednoj dimenziji

Završni rad

Mihaela Pavičić Donkić

Split, rujan 2019.

Zahvaljujem docentici Larisi Zoranić na pomoći pri odabiru teme, prenesenom znanju, motivaciji, strpljenju te pomoći pri pisanju i korekcijama rada.

Hvala mojoj obitelji, prvenstveno roditeljima, na podršci i posebno na strpljenju, bez njih ovo ne bi bilo moguće.

Hvala mom cimeru na bezuvjetnoj podršci, motivaciji i razumijevanju za vrijeme polaganja ispita i pisanja rada.

Temeljna dokumentacijska kartica

Sveučilište u Splitu
Prirodoslovno – matematički fakultet
Odjel za fiziku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Završni rad

Numerička rješenja Burgersove jednadžbe u jednoj dimenziji

Mihaela Pavičić Donkić

Sveučilišni preddiplomski studij Fizika

Sažetak:
Burgersova jednadžba nerijetko se susreće u pojednostavljivanju složenih modela u fizici. Pritom opisuje idealni sustav čije je razumijevanje temeljno za kasnije razumijevanje složenijih sustava. U ovom radu je kao primjer prikazano pojednostavljenje Navier-Stokes jednadžbi. Glavni cilj rada je numeričko rješavanje Burgersove jednadžbe u programskom jeziku C. Pritom se koriste dvije metode: prva metoda je kombinacija Fourierovog razvoja u red te Hopf-Cole transformacije koja Burgersovu jednadžbu svodi na jednadžbu topline, dok se druga metoda temelji na Eulerovoj metodi prema naprijed. Rješenja dobivena prvom metodom nazivaju se točna rješenja te nam time koriste za usporedbu rješenja dobivenih Eulerovom metodom.

Ključne riječi: Burgersova jednadžba, Navier-Stokes jednadžbe, Fourierov razvoj, Hopf-Cole transformacija, Eulerova metoda

Rad sadrži: 40 stranica, 13 slika, 19 tablica, 10 literaturnih navoda. Izvornik je na hrvatskom jeziku

Mentor: doc. dr. sc. Larisa Zoranić

Ocjenjivači: doc. dr. sc. Larisa Zoranić
doc. dr. sc. Petar Stipanović
dr. sc. Ivana Weber

Rad prihvaćen: 23. rujna 2019.

Rad je pohranjen u knjižnici Prirodoslovno – matematičkog fakulteta, Sveučilišta u Splitu.

Basic documentation card

University of Split
Faculty of Science
Department of Physics
Ruđera Boškovića 33, 21000 Split, Croatia

Bachelor thesis

Some of the numerical solutions of one-dimensional Burgers' equation

Mihaela Pavičić Donkić

University undergraduate study programme Physics

Abstract:
Burgers' equation is often encountered in simplifying complex cases in physics. In doing so, it is describing an ideal system whose understanding is fundamental to later understanding of more complex systems. In this paper is presented, as an example, a simplification of the Navier-Stokes equations. The main objective of the paper is to numerically solve the Burgers equation in programming language C. Two methods are used: the first method is a combination of Fourier series and the Hopf-Cole transformation that reduces the Burgers equation to the heat equation, while the second method is based on the Euler forward method. The solutions obtained by the first method are called exact solutions and are thus used to compare the solutions obtained by the Euler method.

Keywords: Burgers' equation, Navier-Stokes equations, Fourier series, Hopf-Cole transformation, Euler forward

Thesis consists of: 40 pages, 13 figures, 19 tables, 10 references. Original language: Croatian

Supervisor: Assist. Prof. Dr. Larisa Zoranić

Reviewers: Assist. Prof. Dr. Larisa Zoranić
Assist. Prof. Dr. Petar Stipanović
Dr. Ivana Weber

Thesis accepted: September 23rd, 2019.

Thesis is deposited in the library of the Faculty of Science, University of Split.

Sadržaj

1. UVOD

U ovom se radu izlaže Burgersova jednadžba, njena primjena te numeričko rješenje. Jednadžbu prvi put spominje H. Bateman u svom radu [1]. Burgersove jednadžbe se često pojavljuju kao rješenja složenijih modela. U tim slučajevima imaju svrhu razumijevanja nekog unutarnjeg ponašanja općeg problema. Najčešće se javljaju u teoriji udarnih valova, problemima turbulencija i kontinuiranim stohastičkim procesima, a primjenjuju se u različitim područjima kao što su dinamika plinova, toplinska vodljivost, elastičnost, pa čak i u toku prometa [9].

U radu je opisan jedan od primjera primjene jednadžbe u fizici, pojednostavljenje Navier-Stokesovih jednadžbi. Naime, idealizacijom fluida kojeg Navier-Stokes jednadžbe opisuju te promatranjem istog u jednoj dimenziji, dobije se upravo Burgersova jednadžba koja stoga opisuje temeljno, idealno ponašanje fluida. Numeričko rješenje se nalazi pomoću dvije metode: prva metoda koristi Hopf-Cole transformaciju koja Burgersovu jednadžbu svodi na jednadžbu topline zajedno s Fourierovim razvojem, dok se druga metoda temelji na Eulerovoj metodi prema naprijed (eng. *Euler forward*). Dobivene vrijednosti međusobno se uspoređuju tablično i grafički. S obzirom da je promatrana jednadžba diferencijalna jednadžba, a poznato je da uvjet egzistencije i jedinstvenosti rješenja diferencijalne jednadžbe zahtjeva da je diferencijalnoj jednadžbi pridružen i početni uvjet, promotriti će se dva primjera s različitim početnim uvjetima. Također, za svaki će se primjer promotriti 6 slučajeva koji se međusobno razlikuju u ulaznim konstantama.

2. BURGERSOVA JEDNADŽBA

Promotrimo pojednostavljenje Navier-Stokes jednadžbi na oblik Burgersove jednadžbe. U općenitom obliku Navier-Stokes jednadžbe za sustav na koji ne djeluje vanjska sila su:

$$\begin{cases} \nabla \cdot \vec{v} = 0 & \text{(2.1a)} \\ \frac{\partial}{\partial t}(\rho\vec{v}) + \nabla \cdot (\rho\vec{v}\vec{v}) + \nabla p - \mu\nabla^2\vec{v} = 0 & \text{(2.1b)} \end{cases} \qquad (2.1)$$

Ako je ρ gustoća, p tlak, $\vec{v}$ brzina i μ viskoznost nekog fluida, jednadžbe (2.1) opisuju dinamiku divergencijski slobodnog, nestlačivog ($\rho_t = 0$) toka gdje su gravitacijski učinci zanemarivi [9]. Uvjet za divergenciju brzine dolazi iz zakona očuvanja mase i uvjeta da se radi o nestlačivom fluidu za koji je gustoća mase konstanta.

Ukoliko su komponente brzine $\vec{v} = (v_x, v_y, v_z)$, izraz (2.1b) može se pojednostaviti u smislu promatranja samo *x* komponente vektora brzine, v_x:

$$\rho\frac{\partial v_x}{\partial t} + \rho v_x\frac{\partial v_x}{\partial x} + \rho v_y\frac{\partial v_x}{\partial y} + \rho v_z\frac{\partial v_x}{\partial z} + \frac{\partial p}{\partial x} - \mu\left(\frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial z^2}\right) = 0 \qquad (2.2)$$

Jednadžba se može dodatno pojednostavniti promatranjem jednodimenzionalnog problema bez gradijenta tlaka. Tada se (2.2) reducira na

$$\rho\frac{\partial v_x}{\partial t} + \rho v_x\frac{\partial v_x}{\partial x} - \mu\frac{\partial^2 v_x}{\partial x^2} = 0 \qquad (2.3)$$

Uvođenjem varijable *u* umjesto v_x te označavanjem kinematičke viskoznosti s ν, tj. $\nu = \frac{\mu}{\rho}$, (2.3) postaje:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \qquad (2.4)$$

Na ovaj način je dobivena takozvana Burgersova jednadžba. Time Burgersova jednadžba opisuje idealan i najosnovniji slučaj dinamike fluida, što je temelj za daljnje, kompleksnije razumijevanje ponašanja istog.

Burgersova jednadžba je nelinearna parabolična diferencijalna jednadžba koja za dano polje $u(x,t)$ ima oblik

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \qquad (2.5)$$

gdje je $\nu > 0$ konstanta viskoznosti. Ovaj oblik Burgersove jednadžbe naziva se i viskozna Burgersova jednadžba. Karakteristično svojstvo jednadžbe je kombinacija difuzijskog $\nu \frac{\partial^2 u}{\partial x^2}$ i konvekcijskog člana $u \frac{\partial u}{\partial x}$. Korisno je primijetiti i da se jednadžba (2.5) može zapisati u obliku

$$\frac{\partial u}{\partial t} + \frac{1}{2} \frac{\partial^2 u}{\partial x^2} = \nu \frac{\partial^2 u}{\partial x^2} \tag{2.6}$$

Često se jednadžbi (2.5) pridružuje i početni uvjet

$$u(x, 0) = f(x), 0 < x < 1 \tag{2.7}$$

te rubni uvjeti

$$u(0, t) = g_1(t), \;\; 0 \leq t \leq T \tag{2.8}$$

$$u(1, t) = g_2(t), \;\; 0 \leq t \leq T \tag{2.9}$$

U slučajevima kada nema difuzijskog člana, tj. kada je $\nu = 0$, jednadžba (2.5) poprima oblik

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0 \tag{2.10}$$

Ovakav oblik jednadžbe naziva se i neviskozna Burgersova jednadžba.

3. HOPF-COLE TRANSFORMACIJA

Hopf-Cole transformacija ima iznimnu važnost jer svodi nelinearnu parcijalnu jednadžbu (2.5) na linearnu jednadžbu topline. Objašnjenje transformacije slijedi u teoremu i njegovu dokazu:

Teorem 1. *U kontekstu s početnim problemom* (2.5) – (2.9).

Ako je $\phi(x,t)$ *bilo koje rješenje jednadžbe topline*

$$\frac{\partial \phi}{\partial t} = \nu \frac{\partial^2 \phi}{\partial x^2} \tag{3.1}$$

onda je nelinearna Hopf-Cole transformacija

$$u = -2\nu \frac{1}{\phi}\frac{\partial \phi}{\partial x} \tag{3.2}$$

rješenje za (2.5) – (2.9).

Dokaz: Neka je

$$u = \frac{\partial \psi}{\partial x}\,, \quad \psi = \psi(x,t) \tag{3.3}$$

Uvrštavanjem u jednadžbu (2.5) i integriranjem s obzirom na *x*, dobije se

$$\frac{\partial \psi}{\partial t} + \frac{1}{2}\left(\frac{\partial \psi}{\partial x}\right)^2 = \nu \frac{\partial^2 \psi}{\partial x^2} \tag{3.4}$$

Uvođenjem $\psi = -2\nu \ln \phi$ slijedi

$$\frac{\partial \phi}{\partial t} = \nu \frac{\partial^2 \phi}{\partial x^2} \tag{3.5}$$

s početnim uvjetom

$$\phi(x,0) = \exp\left(-\frac{1}{2\nu}\int_0^x u_0(x')\mathrm{d}x'\right)\,, \quad 0 < x < 1 \tag{3.6}$$

i rubnim uvjetima

$$\frac{\partial \phi}{\partial x}(0,t) = 0 = \frac{\partial \phi}{\partial x}(1,t)\,, \quad t \geq 0 \tag{3.7}$$

Time je pokazano da se Burgersova jednadžba (2.5) može svesti na linearnu jednadžbu topline (3.1) s Neumannovim rubnim uvjetima (3.7).

Ovisnost konačnog rješenja o početnim uvjetima može se definirati rješavanjem jednadžbe topline (3.1) uz primjenu Fourierovog razvoja:

$$\phi(x,t) = A_0 + \sum_{n=1}^{\infty} A_n \exp(-n^2\pi^2\nu t)\cos(n\pi x) \tag{3.8}$$

Fourierovi koeficijenti u $t = 0$ su

$$A_0 = \int_0^1 \exp\left[-\frac{1}{2\nu}\int_0^x u_0(x')\mathrm{d}x'\right]\mathrm{d}x \tag{3.9}$$

$$A_n = 2\int_0^1 \exp\left[-\frac{1}{2\nu}\int_0^x u_0(x')\mathrm{d}x'\right]\cos(n\pi x)\,\mathrm{d}x \tag{3.10}$$

Na ovaj se način dobije rješenje jednadžbe (2.5):

$$u(x,t) = 2\pi\nu\frac{\sum_{n=1}^{\infty} A_n \exp(-n^2\pi^2\nu t)\, n\sin(n\pi x)}{A_0 + \sum_{n=1}^{\infty} A_n \exp(-n^2\pi^2\nu t)\cos(n\pi x)} \tag{3.11}$$

U nastavku ćemo se na rješenje dobiveno ovom metodom referirati kao na „točno rješenje“.

Transformacija (3.2) pojavila se prvi put u tehničkom izvješću [2], a objavio ju je J.D. Cole [3]. Otprilike u isto vrijeme objavio ju je samostalno E. Hopf [4]. Stoga je transformacija (3.2) poznata kao Hopf-Cole transformacija.

4. NUMERIČKI PRISTUP

Poznata metoda za numeričko rješavanje parcijalnih diferencijalnih jednadžbi je Eulerova metoda prema naprijed (eng. *Euler forward*). Ideja metode temelji se na tome da je domena rješenja jednadžbe (2.5) diskretizirana mrežom $[0,1] \times [0,T]$ [9]. Interval $[0,1]$ podijeli se na N jednakih podintervala s čvorovima $x_0, x_1, \ldots x_N$ međusobno udaljenima za Δx, tj. $\Delta x = 1/N$, dok se vrijeme promatra od početnog trenutka do nekog vremena T u koracima Δt. Kreće se od početnog uvjeta i rubnih uvjeta te se pomoću njih „grade“ ostale vrijednosti rješenja. Postupak dobivanja izraza za uvrštavanje u program kreće od numeričke derivacije.

4.1 Numerička derivacija

Poznato je da za derivaciju funkcije vrijedi:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \lim_{h\to 0} \frac{f(x+h) - f(x)}{\Delta x} \tag{4.1}$$

gdje je Δx korak. Međutim, ukoliko se primjeni Taylorov razvoj:

$$f(x_0 + \Delta x) = f(x_0) + \Delta x \frac{\mathrm{d}f}{\mathrm{d}x}(x)\Big|_{x_0} + \frac{(\Delta x)^2}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x)\Big|_{x_0} + \frac{(\Delta x)^3}{3}\frac{\mathrm{d}^3 f}{\mathrm{d}x^3}(x)\Big|_{x_0} + \sigma((\Delta x)^n) \tag{4.2}$$

te se zanemare članovi višeg reda, dobije se sljedeće:

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x)\Big|_{x_0} \approx \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} \tag{4.3}$$

Lako je uočiti sličnost između (4.1) i (4.3).

Stoga se za računanje numeričke derivacije ima pravo koristiti

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + \sigma((\Delta x)^3) \tag{4.4}$$

Za rješavanje jednadžbe (2.5), prikladno je derivacije zapisati koristeći izraz (4.4):

$$\frac{\partial u}{\partial t}(x,t) \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} \tag{4.5}$$

$$\frac{\partial u}{\partial x}(x,t) \approx \frac{u(x+\Delta x,t) - u(x,t)}{\Delta x} \tag{4.6}$$

$$\frac{\partial^2 u}{\partial x^2}(x,t) \approx \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2} \tag{4.7}$$

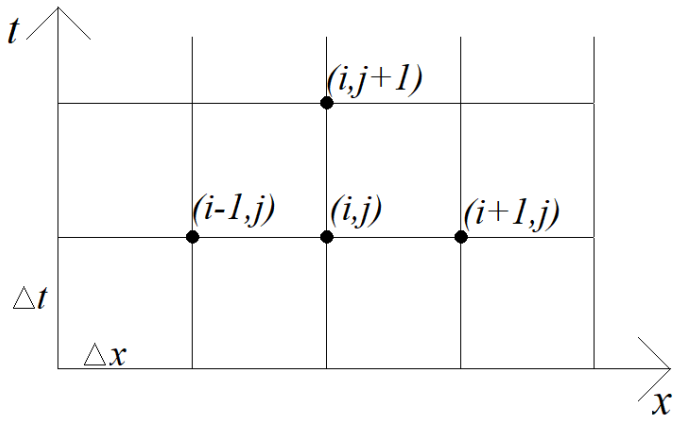
4.2 Primjena Eulerove metode

Shema u kojoj se primjenjuje diskretizacija prostora za Eulerovu metodu može se zamisliti kao na slici 1. Ukoliko se izrazi (4.5), (4.6) i (4.7) prilagode šabloni na način da je točka (x, t) točka (i, j) sa sheme i označi se s (x_i, t_j), dobiju se izrazi:

$$\frac{\partial u}{\partial t}(x,t) = \frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} \tag{4.8}$$

$$\frac{\partial u}{\partial x}(x,t) = \frac{u(x_{i+1}, t_j) - u(x_i, t_j)}{\Delta x} \tag{4.9}$$

$$\frac{\partial^2 u}{\partial x^2}(x,t) = \frac{u(x_{i+1}, t_j) - 2u(x_i, t_j) + u(x_{i-1}, t_j)}{(\Delta x)^2} \tag{4.10}$$



Slika 1 Shema mreže za Eulerovu metodu

Uvrštavanjem (4.8), (4.9) i (4.10) u jednadžbu (2.5), dobije se

$$\frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} + u(x_i, t_j)\frac{u(x_{i+1}, t_j) - u(x_i, t_j)}{\Delta x} = \nu \frac{u(x_{i+1}, t_j) - 2u(x_i, t_j) + u(x_{i-1}, t_j)}{(\Delta x)^2} \tag{4.11}$$

Izraz se pomnoži s Δt, uvedu se konstante

$$p = \frac{\Delta t}{\Delta x} \qquad r = \frac{\nu \Delta t}{(\Delta x)^2} \tag{4.12}$$

te se kratkim sređivanjem dobije izraz pogodan za korištenje u izradi programa:

$$u(x_i, t_{j+1}) = u(x_i, t_j)\left[1 - p \cdot u(x_{i+1}, t_j) + p \cdot u(x_i, t_j) - 2r\right] + r\left(u(x_{i+1}, t_j) + u(x_{i-1}, t_j)\right) \tag{4.13}$$

Može se primijetiti da se vrijednosti dobivaju iz prijašnjih podataka. Točnije, za prvi korak se vrijednosti dobivaju iz vrijednosti koje su zadane početnim uvjetom. Stoga ova metoda ima naziv metoda prema naprijed.

4.3 Numerička integracija

Kako bi se usporedili rezultati dobiveni Eulerovom metodom, korisno je za dani problem pronaći točno rješenje spomenuto na kraju poglavlja 3. Kako bi se izračunali Fourierovi koeficijenti iz (3.9) i (3.10), potrebno je koristiti neku od metoda numeričkog integriranja. U ovom slučaju koristiti će se Simpsonova formula. Simpsonova formula spada u Newton-Cotes kvadraturu te je proširenje trapezne formule. Pritom se površina koja je jednaka iznosu integrala dijeli na jednake intervale te se bira korak $h = \frac{b-a}{M}$ gdje a i b predstavljaju granice integrala, a M predstavlja broj jednakih podintervala. Ideja kreće od Taylorovog razvoja funkcije $f(x)$ u točki $x_0 = x \pm h$:

$$f(x = x_0 \pm h) \approx f(x_0) \pm (x - x_0)\frac{\mathrm{d}f}{\mathrm{d}x}(x)\Big|_{x_0} + \frac{1}{2}(x - x_0)^2 \frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x)\Big|_{x_0} \tag{4.14}$$

gdje su zanemareni članovi reda $(x - x_0)^3$ i više.

Korištenjem izraza za numeričku derivaciju iz poglavlja 4.1 i prilagodbom istih za ovaj slučaj, dobije se:

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x)\Big|_{x_0} \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} \tag{4.15}$$

$$\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x)\Big|_{x_0} \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} \tag{4.16}$$

Korištenjem (4.14)-(4.16), općeniti integral poprima oblik:

$$\begin{aligned}
\int_{x_0-h}^{x_0+h} f(x)\mathrm{d}x \approx \int_{x_0-h}^{x_0+h} \Bigg[& f(x_0) + \frac{f(x_0 + h) - f(x_0 - h)}{2h}(x - x_0) \\
& + \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}\frac{1}{2}(x - x_0)^2 \Bigg] \mathrm{d}(x - x_0) \\
& = f(x_0)(x - x_0)\Big|_{x_0-h}^{x_0+h} \\
& + \frac{f(x_0 + h) - f(x_0 - h)}{2h}\frac{1}{2}(x - x_0)^2 \Bigg|_{x_0-h}^{x_0+h} \\
& + \frac{1}{6}\frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}(x - x_0)^3 \Bigg|_{x_0-h}^{x_0+h} = \cdots = \\
& = \frac{1}{3}h[f(x_0 + h) + hf(x_0) + f(x_0 - h)]
\end{aligned} \tag{4.17}$$

Integral s granicama *a* i *b* se stoga razdvaja na više integrala čije su granice udaljene za *2h*, kao u integralu pod (4.17) te se potom primjenjuje sličan postupak:

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x = \int_a^{a+2h} f(x)\mathrm{d}x + \cdots + \int_{b-2h}^{b} f(x)\mathrm{d}x = \cdots \\ = \frac{h}{3}[f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) \\ + 2f(a+4h) + \cdots + 2f(b-2h) + 4f(b-h) + f(b)]\end{aligned} \tag{4.18}$$

Dobiveni izraz naziva se Simpsonova formula.

5. PROGRAM U C JEZIKU

Kako bi se numerički riješila jednadžba (2.5), bilo primjenom Eulerove metode, bilo primjenom Hopf-Cole transformacije u kombinaciji s Fourierovim razvojem, potrebno je poznavati početne i rubne uvjete. Stoga će se promotriti nekoliko jednostavnih testnih primjera s različitim početnim i rubnim uvjetima kako bi se provjerila učinkovitost metoda [7].

5.1 Primjer 1

Neka je početni problem:

$$u(x,0) = \sin(\pi x) \ , \qquad 0 < x < 1 \tag{5.1}$$

$$u(0,t) = u(1,t) = 0 \ , \ \ 0 \le t \le T \tag{5.2}$$

Za izračun točnog rješenja, potrebni su Fourierovi koeficijenti. Prema (3.9) i (3.10), oni za ovaj primjer imaju oblik:

$$A_0 = \int_0^1 \exp\left[-\frac{1}{2\nu}\int_0^x \sin(\pi x')\,\mathrm{d}x'\right]dx = \int_0^1 \exp\left[\frac{\cos(\pi x)-1}{2\nu\pi}\right]\mathrm{d}x \tag{5.3}$$

$$\begin{aligned} A_n &= 2\int_0^1 e^{-\frac{1}{2\nu}\int_0^x \sin(\pi x')dx'}\cos(n\pi x)\,\mathrm{d}x \\ &= 2\int_0^1 \exp\left[\frac{\cos(\pi x)-1}{2\nu\pi}\right]\cos(n\pi x)\,\mathrm{d}x \end{aligned} \tag{5.4}$$

Dok je točno rješenje jednadžbe (2.5) s uvjetima (5.1) i (5.2) dano s (3.11).

Za izračun rješenja pomoću Eulerove metode, potreban je početni uvjet i rubni uvjeti za funkciju $u(x,t)$. Oni su jednostavno definirani izrazima (5.1) i (5.2), a za dobivanje rješenja koristi se izraz (4.13).

Zgodno je promotriti ponašanje rješenja za različite kombinacije vrijednosti veličina ν, Δx i Δt i to u različitim vremenima *T*. Stoga će se tu promotriti sljedeći slučajevi navedeni u nastavku (Tablica 1).

Tablica 1 Slučajevi za različite vrijednosti ulaznih konstanti

SLUČAJ	ν	Δx	Δt	T
1)	10	0,0125	0,0001	0,01; 0,02; 0,04
2)	20	0,0125	0,0001	0,01; 0,02; 0,04
3)	1	0,0125	0,001	0,1; 0,2; 0,3; 0,4
4)	2	0,0125	0,001	0,1; 0,2; 0,3; 0,4
5)	0,1	0,0125	0,01	0.4; 0,6; 1; 2; 3; 4
6)	0,2	0,0125	0,01	0.4; 0,6; 1; 2; 3; 4

5.1.1 Program za računanje koeficijenta A_0

Koeficijent A_0 kao što je i prije spomenuto dobije se korištenjem Simpsonove metode. Kako A_0 ovisi samo o konstanti *v*, a time i o promatranom slučaju. Korisno je u svrhu ubrzanja izvršavanja programa napraviti poseban program za izračunavanje koeficijenta te kasnije vrijednost istog definirati u programu za rješenje. U programu se prvo uključuju potrebne biblioteke te definiraju potrebne konstante:

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define a 0
#define b 1
#define h 0.000000125
#define M (b-a)/h
```

S obzirom da se u Simpsonovoj formuli više puta koristi vrijednost podintegralne funkcije, korisno je uvesti funkciju koja računa navedenu vrijednost. Parametri koji „ulaze“ u funkciju su varijabla *x* te vrijednost konstante v:

```
double fja(double x, double v) {
	double vrij;
	vrij = exp((cos(PI*x) - 1) / (2 * v * PI));
	return vrij;
}
```

Kako je rečeno da će se promatrati više vrijednosti konstante *v*, a time i više vrijednosti koeficijenta A_0, korisno je te varijable u programu uzeti u obliku polja. Dimenziju polja definira broj primjera koji će se računati. Stoga početak glavnog programa izgleda:

```
int main(void) {
	double A0[6], v[6], x;
	int i, j;
```

Pritom *i* i *j* imaju uloge brojača u petljama; *i* se pojavljuje u petljama za prolazak kroz polja, a *j* u petlji za računanje integrala.

Kako će se koeficijent A_0 koristiti kasnije u računanju točnog rješenja, prikladno je izračunate vrijednosti zapisati u datoteci kako bi se kasnije lakše pronašle:

```
	FILE *tok;
	tok = fopen("Primjer_1_Vrijednosti_A0.txt", "w");
	fprintf(tok, "v\tA0\t\n");
```

Time je stvorena datoteka pod nazivom „Primjer_1_Vrijednosti_A0.txt“ te je pripremljena za unos dobivenih vrijednosti. Također, u prvom retku datoteke je upisan redak koji označava koje se veličine nalaze u kojem stupcu.

Potom se inicijaliziraju vrijednosti konstante ν:

```
v[0] = 10;
v[1] = 20;
v[2] = 1.0;
v[3] = 2.0;
v[4] = 0.1;
v[5] = 0.2;
```

te se konačno kreće s izračunom koeficijenta A_0 korištenjem Simpsonove formule:

```
x = a;
for (i = 0; i < 6; i++){
	A0[i] = fja(a, v[i]) + fja(b, v[i]);
	for (j = 1; j <= M; j++) {
		x += h;
		if (j % 2 == 0)
			A0[i] += 2 * fja(x, v[i]);
		else
			A0[i] += 4 * fja(x, v[i]);
	}
	A0[i] *= h / 3;
	fprintf(tok, "%.2f\t%.6f\t\n", v[i], A0[i]);
}
fclose(tok);
return 0;
}
```

Nakon izvršavanja koda, dobivena je tablica s podacima prikazanima u nastavku (Tablica 2).

Tablica 2 Izračunate vrijednosti za koeficijent A_0 ovisno o konstanti ν za primjer 1

ν	A_0
10.00	0.984273
20.00	0.992090
1.00	0.858274
2.00	0.924969
0.10	0.354455
0.20	0.525546

5.1.2 Program za računanje koeficijenta A_n i točnog rješenja

U nastavku je prikazan program za računanje rješenja u slučaju 1. Rješenja za ostale slučajeve dobiju se izmjenom prethodno navedenih konstanti (Tablica 1). u samom zaglavlju koda.

Na početku programa se uključuju potrebne biblioteke te definiraju potrebne konstante

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define v 10
#define A0 0.984273
#define a 0.0
#define b 1.0
```

```
#define dy 0.00000125
#define K (b-a)/dy
#define dx 0.0125
#define M (b-a)/dx
#define dt 0.0001
#define T 0.04
#define N T/dt
```

Pritom je definirana i konstanta A_0 dobivena programom iz potpoglavlja 5.1.1 za $\nu = 10$. dy predstavlja korak za računanje integrala potrebnog za dobivanje koeficijenta A_n, K je broj koraka dy, dx predstavlja prostorni korak potreban za računanje točnog rješenja, M je broj koraka dx, dt predstavlja vremenski korak za računanje rješenja, T je ukupno vrijeme, tj. definira interval $[0, T]$ na kojem se traži rješenje, a N je broj koraka dt.

Potom se uvodi funkcija za računanje koeficijenta A_n Simpsonovom formulom, na sličan način kao i za računanje koeficijenta A_0. Treba pripaziti na bitnu razliku između koeficijenata A_0 i A_n, a to je da A_0 ovisi samo o konstanti ν, dok A_n ovisi i o vrijednosti indeksa n. Zbog toga se vrijednost koeficijenta A_n računa u posebno definiranoj funkciji u kodu na sljedeći način:

```
long double fjaAn(long double x, int n) {
	long double vrijAn;
	vrijAn = 2 * exp((cos(PI*x) - 1) / (2 * v*PI))*cos(n*PI*x);
	return vrijAn;
}

long double fAn(int n) {
	long double y, intAn;
	int i;
	intAn = fjaAn(a, n) + fjaAn(b, n);
	y = a;
	for (i = 1; i <= K; i++) {
		y += dy;
		if (i % 2 == 0)
			intAn += 2 * fjaAn(y, n);
		else
			intAn += 4 * fjaAn(y, n);
	}
	intAn *= dy / 3.0;
	return intAn;
}
```

Prva funkcija računa vrijednost podintegralne funkcije, dok druga funkcija računa vrijednost koeficijenta za dani n.

Zbog osjetljivosti podataka, varijable (osim brojača) se definiraju kao tip „long double“ kako bi se smanjio gubitak podataka. Koeficijent A_n definiramo u obliku polja te ga pri početku glavnog programa odmah računamo kako bi kasnije smanjili broj iteracija te time znatno ubrzali izvršavanje programa. Početak glavnog programa je:

```
int main(void) {
	long double x = a, t, ut, brojnik, nazivnik, An[50];
	int n, i, j;
```

Pritom je *n* brojač za sume potrebne za izračunavanje točnog rješenja iz izraza (3.11) te time i indeks za prethodno navedeni koeficijent A_n, *i* je brojač za prostorni korak, a *j* za vremenski korak.

Kako bi se kasnije mogli usporediti podaci dobiveni ovom metodom s podacima dobivenim Eulerovom metodom, te iste podatke prikazati grafički, korisno je iste zabilježiti u datoteku:

```
	FILE *tok;
	tok = fopen("Primjer_1_Euler_slucaj_1.txt", "w");
	fprintf(tok, "x\tt\tu(x,t)\n");
```

Time je stvorena datoteka s upisanim prvim retkom i spremna je za upisivanje podataka.

Potom se, kao što je prethodno rečeno, uvodi petlja za postavljanje vrijednosti koeficijenta A_n. Pri računu se poziva funkcija definirana na početku koda:

```
	for (n = 0; n < 50; n++) {
		An[n] = fAn(n + 1);
	}
```

Sada konačno kreće izračun točnog rješenja te upisivanje dobivenih podataka u datoteku:

```
	for (i = 0; i <= M; i++) {
		x = i * dx;
			for (j = 0; j <= N; j++) {
				t = j * dt;
				if (t == 0.01 || t == 0.02 || t == 0.04) {
					brojnik = 0;
					nazivnik = A0;
					for (n = 1; n <= 50; n++) {
					 brojnik+=An[n-1]*exp(-n*n*PI*PI*v*t)*n*sin(n*PI*x);
					 nazivnik+=An[n-1]*exp(-n*n*PI*PI*v*t)*cos(n*PI*x);
					}
					ut = (2 * PI*v)*brojnik / nazivnik;
					fprintf(tok, "%.4lf\t%.2lf\t%.5lf\t\n", x, t, ut);
				}
			}
	}
	fclose(tok);
	return 0;
}
```

Nakon izvršavanja koda, dobivena je tablica s podacima dijelom prikazanima u nastavku (Tablica 3).

Tablica 3 Izračunate vrijednosti točnog rješenja za primjer 1, slučaj 1

x	t	$u(x,t)$
0.0000	0.01	0.00000
0.0000	0.02	0.00000
0.0000	0.04	0.00000
0.0125	0.01	0.01456
0.0125	0.02	0.00544
0.0125	0.04	0.00076
0.0250	0.01	0.02909
0.0250	0.02	0.01087
0.0250	0.04	0.00151
0.0375	0.01	0.04358
0.0375	0.02	0.01629
0.0375	0.04	0.00227

Kao što je prethodno rečeno, točno rješenje za ostale slučajeve dobije se promjenom definiranih vrijednosti u zaglavlju programa. Dobiveni podaci za ostale slučajeve biti će prikazani kasnije, zajedno s podacima dobivenima Eulerovom metodom.

5.1.3 Program za Eulerovu metodu

U nastavku je, kao i u prethodnom potpoglavlju, prikazan program za slučaj 1, dok se rješenja za ostale slučajeve dobiju izmjenom konstanti.

Kao i uvijek, na početku se uključuju potrebne biblioteke i definiraju potrebne konstante

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define v 10.0
#define dt 0.0001
#define T 0.04
#define M 400
#define dx 0.1
#define N 10
#define r (v*dt)/(dx*dx)
#define p dt/dx
```

dt predstavlja vremenski korak, T ukupno vrijeme, a M je broj koraka, tj. $M = T/dt$. Analogno, dx predstavlja prostorni korak, a N broj koraka, tj. $N = 1/dx$. p i r su konstante (4.12).

Funkciju za računanje vrijednosti iz početnog uvjeta, kao i rubnih uvjeta, u ovom slučaju ne treba izdvajati jer su uvjeti jednostavne funkcije. Stoga odmah kreće početak glavnog dijela programa te definiranje tipova varijabli:

```
int main(void) {
	long double x, t, u[N+2][M+1];
	int i, j;
```

Kao i u prethodnom programu, *i* je brojač za prostorni korak, a *j* je brojač za vremenski korak.

Podatci se ponovno zapisuju u datoteku:

```
FILE *tok;
tok = fopen("Primjer_1_Euler_slucaj_1_N=10.txt", "w");
fprintf(tok, " x\t\tu(x,t=0.01)\tu(x,t=0.02)\tu(x,t=0.04)\n ");
```

Kako bi se mogla primijeniti Eulerova metoda, potrebno je postaviti rubne vrijednosti sukladno početnom i rubnim uvjetima:

```
for (i = 1; i <= N; i++) {
	x = i * dx;
	u[i][0] = sin(PI*x);
}

for (j = 0; j <= M; j++) {
	u[0][j] = 0.0;
	u[N][j] = 0.0;
}
```

Prva petlja postavlja vrijednosti sukladno početnom uvjetu, a druga sukladno rubnim uvjetima.

Sada može krenuti glavni dio programa, a to je računanje rješenja primjenom Eulerove metode:

```
for (j = 0; j < M; j++) {
	for (i = 1; i < N; i++) {
	 u[i][j+1]=u[i][j]*(1-p*u[i+1][j]+p*u[i][j]-2*r)+r*(u[i+1][j]+u[i-1][j]);
	}
}
```

Na kraju izračuna, dobivene vrijednosti se upisuju u datoteku:

```
x = 0.0;
for (i = 0; i <= N; i++) {
	fprintf(tok, "%lf\t", x);
	x = x + dx;
	for (j = 0; j <= M; j++) {
		t = j * dt;
		if (t == 0.01 || t == 0.02 || t == 0.04)
			fprintf(tok, "%lf\t", u[i][j]);

	}
	fprintf(tok, "\n");
}
fclose(tok);
return 0;
}
```

Time je program stvorio datoteku s podacima navedenima u nastavku (Tablica 4).

Tablica 4 Izračunate vrijednosti za u(x,t=0.01) za N=10 za primjer 1, slučaj 1 korištenjem Eulerove metode

x	$u(x, t = 0.01)$	$u(x, t = 0.02)$	$u(x, t = 0.04)$
0.000000	0.000000	0.000000	0.000000
0.100000	0.115261	0.043270	0.006065
0.200000	0.219417	0.082333	0.011536
0.300000	0.302373	0.113382	0.15880
0.400000	0.356001	0.133375	0.018669
0.500000	0.374941	0.140338	0.019632
0.600000	0.357169	0.133562	0.018673
0.700000	0.304262	0.113685	0.015886
0.800000	0.221306	0.082636	0.011542
0.900000	0.116428	0.043458	0.006069
1.000000	0.000000	0.000000	0.000000

Ostale dobivene vrijednosti, za različite vrijednosti *N*, kao i za ostale slučajeve će se iznijeti kasnije, u poglavlju usporedbi metoda.

5.2 Primjer 2

Početni problem je:

$$u(x,0) = 4x(1-x)\ , \qquad 0 < x < 1 \tag{5.5}$$

$$u(0,t) = u(1,t) = 0\ , \qquad 0 \le t \le T \tag{5.6}$$

Kao i u prošlom primjeru, za izračun rješenja, ponovno su potrebni Fourierovi koeficijenti. Prema (3.9) i (3.10), za ovaj slučaj imaju oblik:

$$A_0 = \int_0^1 \exp\left[-\frac{1}{2\nu}\int_0^x 4x'(1-x')\mathrm{d}x'\right]\mathrm{d}x = \int_0^1 \exp\left[\frac{x^2(2x-3)}{3\nu}\right]\mathrm{d}x \tag{5.7}$$

$$\begin{aligned} A_n &= 2\int_0^1 e^{-\frac{1}{2\nu}\int_0^x \sin(\pi x')\mathrm{d}x'}\cos(n\pi x)\,\mathrm{d}x \\ &= 2\int_0^1 \exp\left[\frac{x^2(2x-3)}{3\nu}\right]\cos(n\pi x)\,\mathrm{d}x \end{aligned} \tag{5.8}$$

Dok je točno rješenje jednadžbe (2.5) s uvjetima (5.5) i (5.6) ponovno (3.11).

Jednako kao i za problem 1, za izračun rješenja pomoću Eulerove metode, potreban je početni uvjet i rubni uvjeti za funkciju $u(x,t)$. Oni su jednostavno definirani izrazima (5.5) i (5.6), a za dobivanje rješenja koristi se izraz (4.13).

Ponovno će se promotriti ponašanje rješenja za različite kombinacije vrijednosti veličina ν, Δx i Δt i to u različitim vremenima T. Koristiti će se isti slučajevi kao i u primjeru 1 (Tablica 1).

5.2.1 Program za računanje koeficijenta A_0

Program za računanje koeficijenta analogan je programu za prethodni primjer s jedinom razlikom u obliku podintegralne funkcije koja se definira prije glavnog dijela programa:

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define a 0
#define b 1
#define h 0.000000125
#define M (b-a)/h

double fja(double x, double v) {
	double vrij;
	vrij = exp((x*x*(2*x-3))/(3*v));
	return vrij;
}
```

Tu je uočljiva razlika između programa za primjer 1 (potpoglavlje 5.1.1). Nastavak koda jednak je onom za primjer 1:

```
int main(void) {
	double A0[6], v[6], x;
	int i, j;
	FILE *tok;
	tok = fopen("Primjer_2_Vrijednosti_A0.txt", "w");
	fprintf(tok, "v\tA0\t\n");
	v[0] = 10.0;
	v[1] = 20.0;
	v[2] = 1.0;
	v[3] = 2.0;
	v[4] = 0.1;
	v[5] = 0.2;
	x = a;
	for (i = 0; i < 6; i++) {
		A0[i] = fja(a, v[i]) + fja(b, v[i]);
		for (j = 1; j <= M; j++) {
			x = j*h;
			if (j % 2 == 0)
				A0[i] += 2 * fja(x, v[i]);
			else
				A0[i] += 4 * fja(x, v[i]);
		}
		A0[i] *= h / 3;
		fprintf(tok, "%.2f\t%.6f\t\n", v[i], A0[i]);
	}
	fclose(tok);
	return 0;
}
```

Nakon izvršavanja, stvorena je tablica s podacima prikazanima u nastavku (Tablica 5).

Tablica 5 Izračunate vrijednosti za koeficijent A_0 ovisno o konstanti ν za primjer 2

ν	A_0
10.00	0.983538
20.00	0.991718
1.00	0.852202
2.00	0.921597
0.10	0.339891
0.20	0.511097

5.2.2 Program za računanje koeficijenta A_n i točnog rješenja

Kao i za primjer 1, prikazati će se kod za slučaj 1, dok se rješenja za ostale slučajeve ponovno dobiju izmjenom konstanti. Razlike u kodu za primjer 1 i primjer 2 su minimalne: mijenja se vrijednost koeficijenta A_0 u zaglavlju sukladno prethodno navedenoj tablici (Tablica 5) te se mijenja oblik podintegralne funkcije za računanje koeficijenta A_n. Stoga je kod sljedeći:

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define v 10
#define A0 0.983538
```

```
#define a 0.0
#define b 1.0
#define dy 0.00000125
#define K (b-a)/dy
#define dx 0.0125
#define M (b-a)/dx
#define dt 0.0001
#define T 0.04
#define N T/dt

long double fjaAn(long double x, int n) {
	long double vrijAn;
	vrijAn = 2 * exp((x*x*(2 * x - 3)) / (3 * v))*cos(n*PI*x);
	return vrijAn;
}

long double fAn(int n) {
	long double y, intAn;
	int i;
	intAn = fjaAn(a, n) + fjaAn(b, n);
	y = a;
	for (i = 1; i <= K; i++) {
		y += dy;
		if (i % 2 == 0)
			intAn += 2 * fjaAn(y, n);
		else
			intAn += 4 * fjaAn(y, n);
	}
	intAn *= dy / 3.0;
	return intAn;
}

int main(void) {
	long double x = a, t, ut, brojnik, nazivnik, An[50];
	int n, i, j;
	FILE *tok;
	tok = fopen("Primjer_2_Euler_slucaj_1.txt", "w");
	fprintf(tok, "x\tt\tu(x,t)\n");
	for (n = 0; n < 50; n++) {
		An[n] = fAn(n + 1);
	}
	for (i = 0; i <= M; i++) {
		x = i * dx;
			for (j = 0; j <= N; j++) {
				t = j * dt;
				if (t == 0.01 || t == 0.02 || t == 0.04) {
					brojnik = 0;
					nazivnik = A0;
					for (n = 1; n <= 50; n++) {
					 brojnik+=An[n-1]*exp(-n*n*PI*PI*v*t)*n*sin(n*PI*x);
					 nazivnik+=An[n-1]*exp(-n*n*PI*PI*v*t)*cos(n*PI*x);
					}
					ut = (2 * PI*v)*brojnik / nazivnik;
					fprintf(tok, "%.4lf\t%.2lf\t%.5lf\t\n", x, t, ut);
				}
			}
	}
	fclose(tok);
	return 0;
}
```

Nakon izvršavanja koda, dobivena je tablica s podacima dijelom prikazanima u nastavku (Tablica 6).

Tablica 6 Izračunate vrijednosti točnog rješenja za primjer 2, slučaj 1

x	t	$u(x,t)$
0.0000	0.01	0.00000
0.0000	0.02	0.00000
0.0000	0.04	0.00000
0.0125	0.01	0.01502
0.0125	0.02	0.00562
0.0125	0.04	0.00078
0.0250	0.01	0.03002
0.0250	0.02	0.01122
0.0250	0.04	0.00156
0.0375	0.01	0.04498
0.0375	0.02	0.01681
0.0375	0.04	0.00234

5.2.3 Program za Eulerovu metodu

Kao i prethodni kod, i ovaj kod se minimalno razlikuje od koda za primjer 1. Jedina razlika je u funkciji koja postavlja početni uvjet, dok je sve ostalo jednako. Stoga je kod:

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define v 10.0
#define dt 0.0001
#define T 0.04
#define M 400
#define dx 0.1
#define N 10
#define r (v*dt)/(dx*dx)
#define p dt/dx


int main(void) {
	long double x, t, u[N+2][M+1];
	int i, j;
	FILE *tok;
	tok = fopen("Primjer_2_Euler_slucaj_1_N=10.txt", "w");
	fprintf(tok, "x\t\tu(x,t=0.01)\tu(x,t=0.02)\tu(x,t=0.04)\n");

	for (i = 1; i <= N; i++) {
		x = i * dx;
		u[i][0] = 4*x*(1-x);
	}

	for (j = 0; j <= M; j++) {
		u[0][j] = 0.0;
		u[N][j] = 0.0;
	}
```

```
	for (j = 0; j < M; j++) {
		for (i = 1; i < N; i++) {
		 u[i][j+1]=u[i][j]*(1-p*u[i+1][j]+p*u[i][j]-2*r)+r*(u[i+1][j]+u[i-1][j]);
		}
	}

	x = 0.0;
	for (i = 0; i <= N; i++) {
		fprintf(tok, "%lf\t", x);
		x = x + dx;
		for (j = 0; j <= M; j++) {
			t = j * dt;
			if (t == 0.01 || t == 0.02 || t == 0.04)
				fprintf(tok, "%lf\t", u[i][j]);

		}
		fprintf(tok, "\n");
	}
	fclose(tok);
	return 0;
}
```

Program je stvorio datoteku s podacima navedenima u nastavku (Tablica 7).

Tablica 7 Izračunate vrijednosti za u(x,t=0.01) za N=10 za primjer 2, slučaj 1 korištenjem Eulerove metode

x	$u(x,t=0.01)$	$u(x,t=0.02)$	$u(x,t=0.04)$
0.000000	0.000000	0.000000	0.000000
0.100000	0.118945	0.044656	0.006259
0.200000	0.226430	0.084971	0.011907
0.300000	0.312041	0.117015	0.016389
0.400000	0.367393	0.137652	0.019269
0.500000	0.386954	0.144842	0.020263
0.600000	0.368632	0.137851	0.019273
0.700000	0.314046	0.117338	0.016396
0.800000	0.228435	0.085293	0.011913
0.900000	0.120183	0.044855	0.006263
1.000000	0.000000	0.000000	0.000000

6. USPOREDBA METODA

U ovom će se poglavlju, kao što je prethodno spominjano, dijelom prikazati dobiveni podaci za sve slučajeve te će se numerički i grafički usporediti Eulerova metoda i metoda točnog rješenja za oba primjera i sve slučajeve.

U nastavku su, u svrhu usporedbe, prikazane tablice s dijelom dobivenih podataka za oba primjera, za sve slučajeve (Tablica 8 -Tablica 19). Tekst „nan“ označava situacije u kojima program nije bio u mogućnosti provesti izračun te je tada, umjesto vrijednosti, ispisao „-nan(ind)“.

Ispod svake od tablica, nalazi se i graf za isti slučaj (**Slika 2 - Slika 13**), kako bi se vizualno lakše prikazale eventualne razlike u metodama.

Lako je uočljivo da se povećanjem vrijednosti *N*, odnosno smanjenjem koraka dx, rješenje korištenjem Eulerove metode približava točnom rješenju. Međutim, za veće vrijednosti *N*, metoda ne daje rješenja. Metoda ili ne konvergira ili su rješenja izvan mogućeg zapisan broja. Jedan od mogućih problema metode je i to što se u računanju numeričke derivacije u nazivniku pojavljuju oblici tipa $f(x + \Delta x) - f(x)$ što za jako male korake Δx, tj. za velike vrijednosti *N*, predstavlja dva skoro jednaka broja. Time je vrijednost nazivnika približno jednaka nuli, što u konačnici može programu predstavljati problem [10].

Također, uočljivo je da se za slučaj 4 javlja malo veće odstupanje rješenja dobivenog Eulerovom metodom od točnog rješenja nego za slučaj 3. Stoga se može zaključiti, s obzirom na to da se slučajevi 3 i 4 razlikuju samo za konstantu ν, da povećanje navedene konstante povećava grešku Eulerove metode.

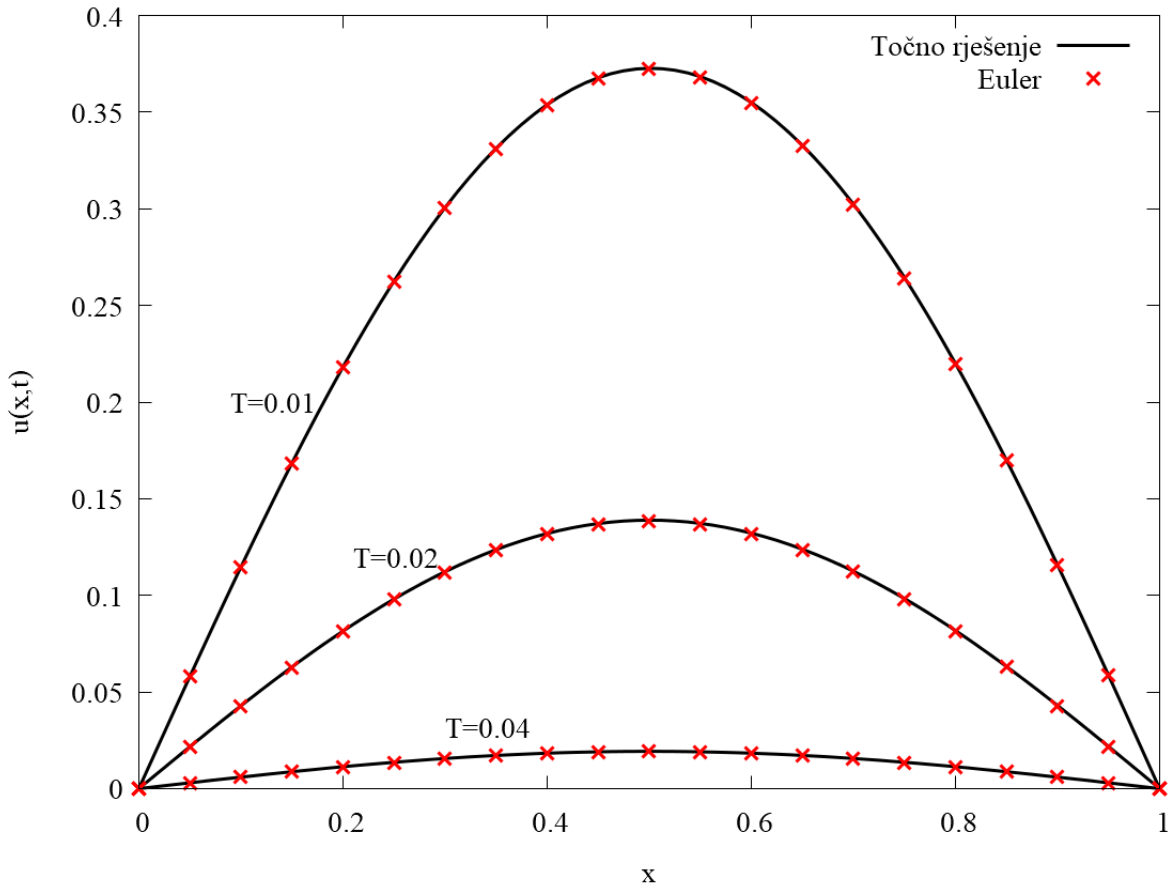
Lako je uočljivo i da se kod slučajeva 5 i 6 događa velika greška u Eulerovoj metodi. Grešku se dijelom može pripisati prethodno spomenutoj konstanti ν, ali i povećanom vremenskom koraku te ukupnom vremenskom intervalu u kojem se traži rješenje. Treba ponovno spomenuti i malu vrijednost broja koraka *N*, odnosno velik korak dx, čime se zasigurno desila pogreška.

Važno je spomenuti i uvjet stabilnosti Eulerove metode prema naprijed koji ograničava vrijednosti vremenskog i prostornog koraka, a dan je s:

$$\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{6.1}$$

Tablica 8 Dobivene vrijednosti rješenja za primjer 1, slučaj 1

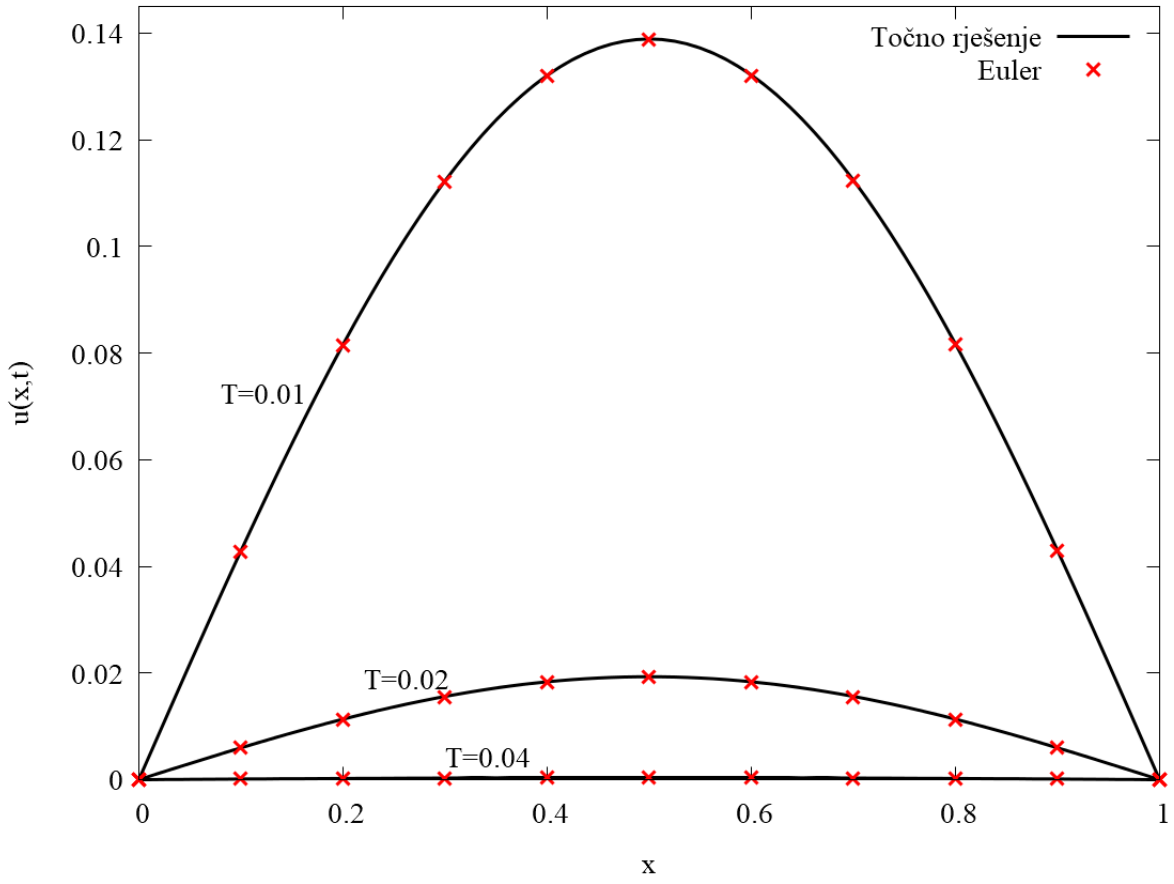
	T=0,01				**T=0,04**			
	Euler			Točno rješenje	Euler			Točno rješenje
x	N=10	N=20	N=40		N=10	N=20	N=40	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,11526	0,11443	nan	0,11461	0,00607	0,00591	nan	0,00596
0,2	0,21942	0,21782	nan	0,21816	0,01154	0,01123	nan	0,01134
0,3	0,30237	0,30015	nan	0,30062	0,01588	0,01546	nan	0,01561
0,4	0,35600	0,35336	nan	0,35390	0,01867	0,01818	nan	0,01835
0,5	0,37494	0,37214	nan	0,37270	0,01963	0,01912	nan	0,01930
0,6	0,35717	0,35449	nan	0,35502	0,01867	0,01818	nan	0,01835
0,7	0,30426	0,30198	nan	0,30243	0,01589	0,01547	nan	0,01561
0,8	0,22131	0,21965	nan	0,21997	0,01154	0,01124	nan	0,01134
0,9	0,11643	0,11556	nan	0,11573	0,00607	0,00591	nan	0,00596
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 2 Grafički prikaz dobivenih rješenja za primjer 1, slučaj 1, za N=20, $\Delta t = 0{,}0001$

Tablica 9 Dobivene vrijednosti rješenja za primjer 1, slučaj 2

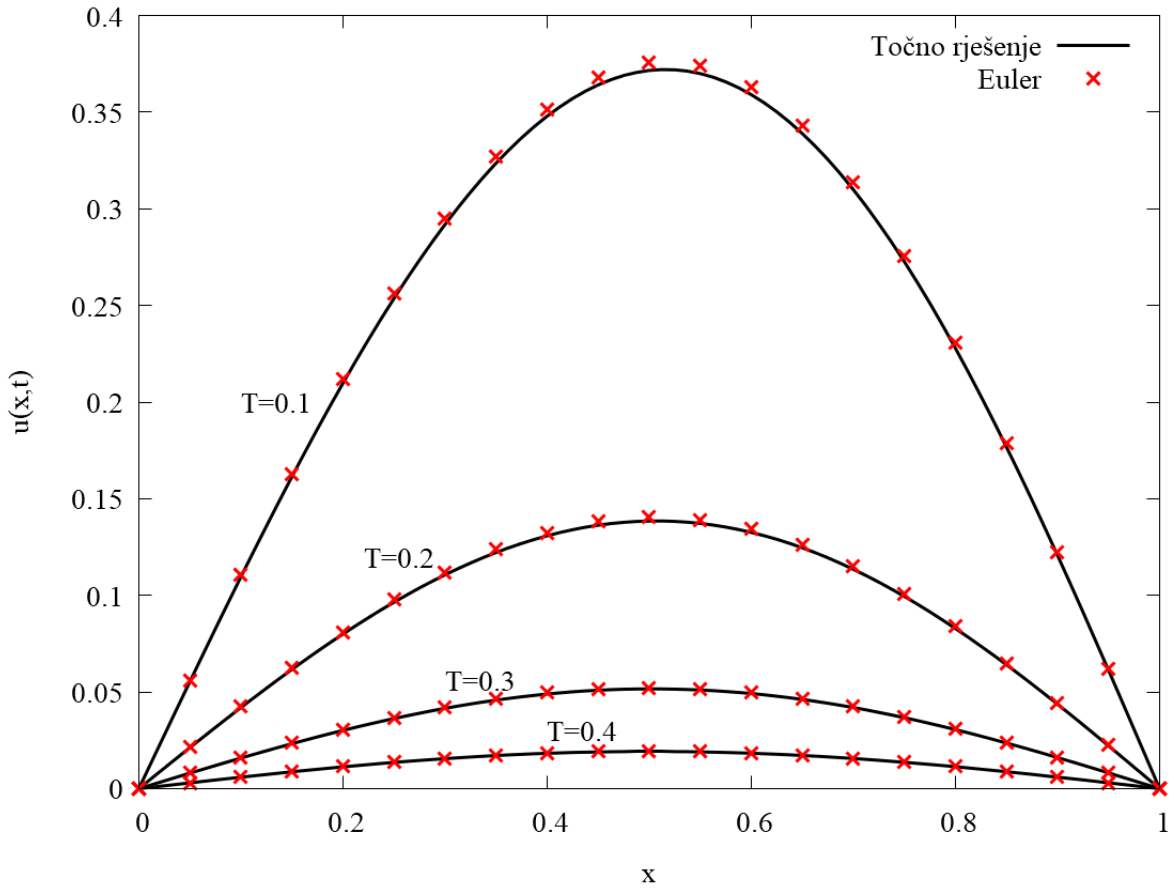
	T=0,01			**T=0,04**		
	Euler		Točno rješenje	Euler		Točno rješenje
x	N=10	N=20		N=10	N=20	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,04282	nan	0,04288	0,00011	nan	0,00012
0,2	0,08146	nan	0,08158	0,00022	nan	0,00022
0,3	0,11215	nan	0,11231	0,00030	nan	0,00030
0,4	0,13189	nan	0,13207	0,00035	nan	0,00035
0,5	0,13872	nan	0,13891	0,00037	nan	0,00037
0,6	0,13198	nan	0,13216	0,00035	nan	0,00035
0,7	0,11230	nan	0,11245	0,00030	nan	0,00030
0,8	0,08161	nan	0,08172	0,00022	nan	0,00022
0,9	0,04291	nan	0,04297	0,00011	nan	0,00012
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 3 Grafički prikaz dobivenih rješenja za primjer 1, slučaj 2, za N=10, $\Delta t = 0{,}0001$

Tablica 10 Dobivene vrijednosti rješenja za primjer 1, slučaj 3

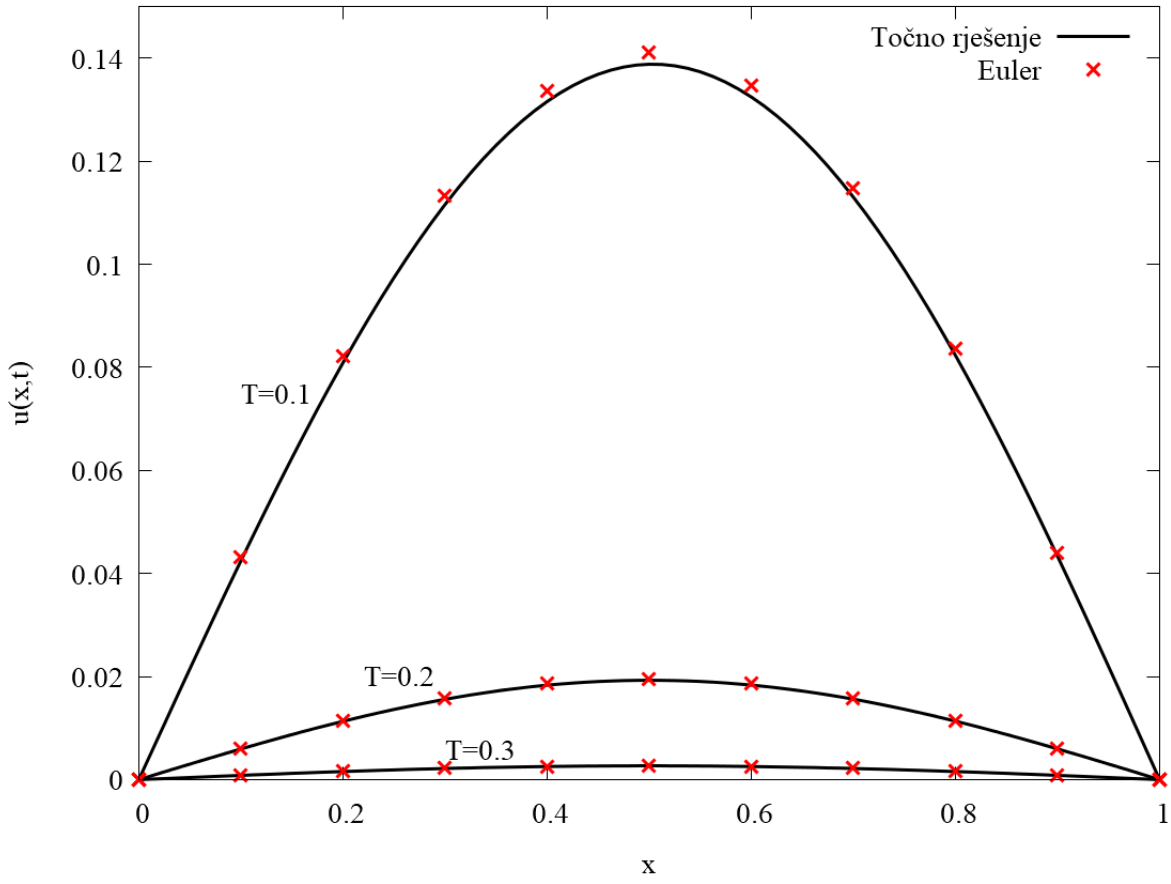
	T=0,1				**T=0,4**			
	Euler			Točno rješenje	Euler			Točno rješenje
x	N=10	N=20	N=40		N=10	N=20	N=40	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,11238	0,11049	nan	0,10954	0,00627	0,00598	nan	0,00593
0,2	0,21545	0,21170	nan	0,20979	0,01193	0,01139	nan	0,01128
0,3	0,30019	0,29472	nan	0,29190	0,01643	0,01569	nan	0,01553
0,4	0,35835	0,35152	nan	0,34792	0,01933	0,01846	nan	0,01828
0,5	0,38326	0,37563	nan	0,37158	0,02035	0,01942	nan	0,01924
0,6	0,37075	0,36312	nan	0,35905	0,01937	0,01849	nan	0,01831
0,7	0,32024	0,31350	nan	0,30990	0,01649	0,01574	nan	0,01559
0,8	0,23548	0,23047	nan	0,22782	0,01199	0,01145	nan	0,01133
0,9	0,12474	0,12209	nan	0,12069	0,00631	0,00602	nan	0,00596
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 4 Grafički prikaz dobivenih rješenja za primjer 1, slučaj 3, za N=20, $\Delta t = 0{,}001$

Tablica 11 Dobivene vrijednosti rješenja za primjer 1, slučaj 4

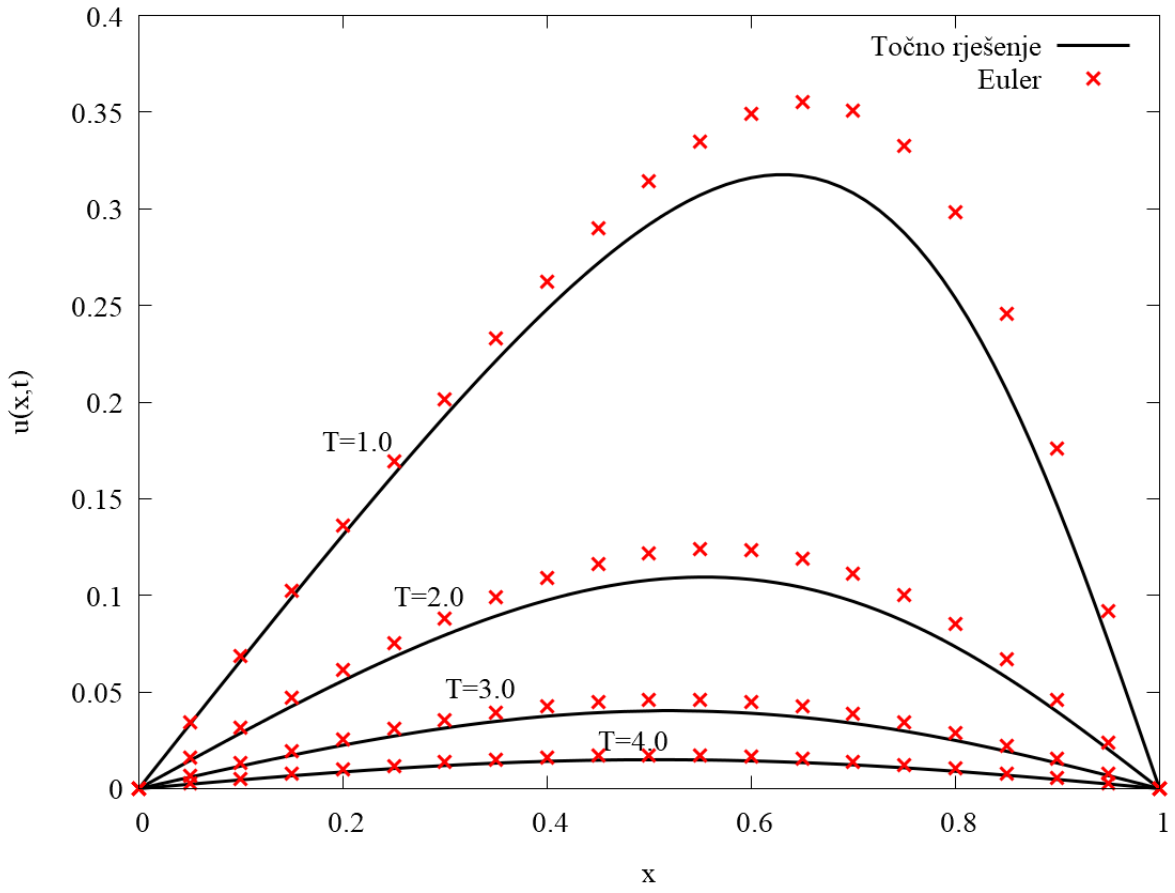
	T=0,1			**T=0,4**		
	Euler		Točno rješenje	Euler		Točno rješenje
x	N=10	N=20		N=10	N=20	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,04309	nan	0,04245	0,00012	nan	0,00011
0,2	0,08210	nan	0,08088	0,00022	nan	0,00022
0,3	0,11330	nan	0,11158	0,00030	nan	0,00030
0,4	0,13362	nan	0,13157	0,00036	nan	0,00035
0,5	0,14100	nan	0,13880	0,00038	nan	0,00037
0,6	0,13457	nan	0,13245	0,00036	nan	0,00035
0,7	0,11483	nan	0,11301	0,00030	nan	0,00030
0,8	0,08363	nan	0,08231	0,00022	nan	0,00022
0,9	0,04403	nan	0,04334	0,00012	nan	0,00011
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 5 Grafički prikaz dobivenih rješenja za primjer 1, slučaj 4, za N=10, $\Delta t = 0{,}001$

Tablica 12 Dobivene vrijednosti rješenja za primjer 1, slučaj 5

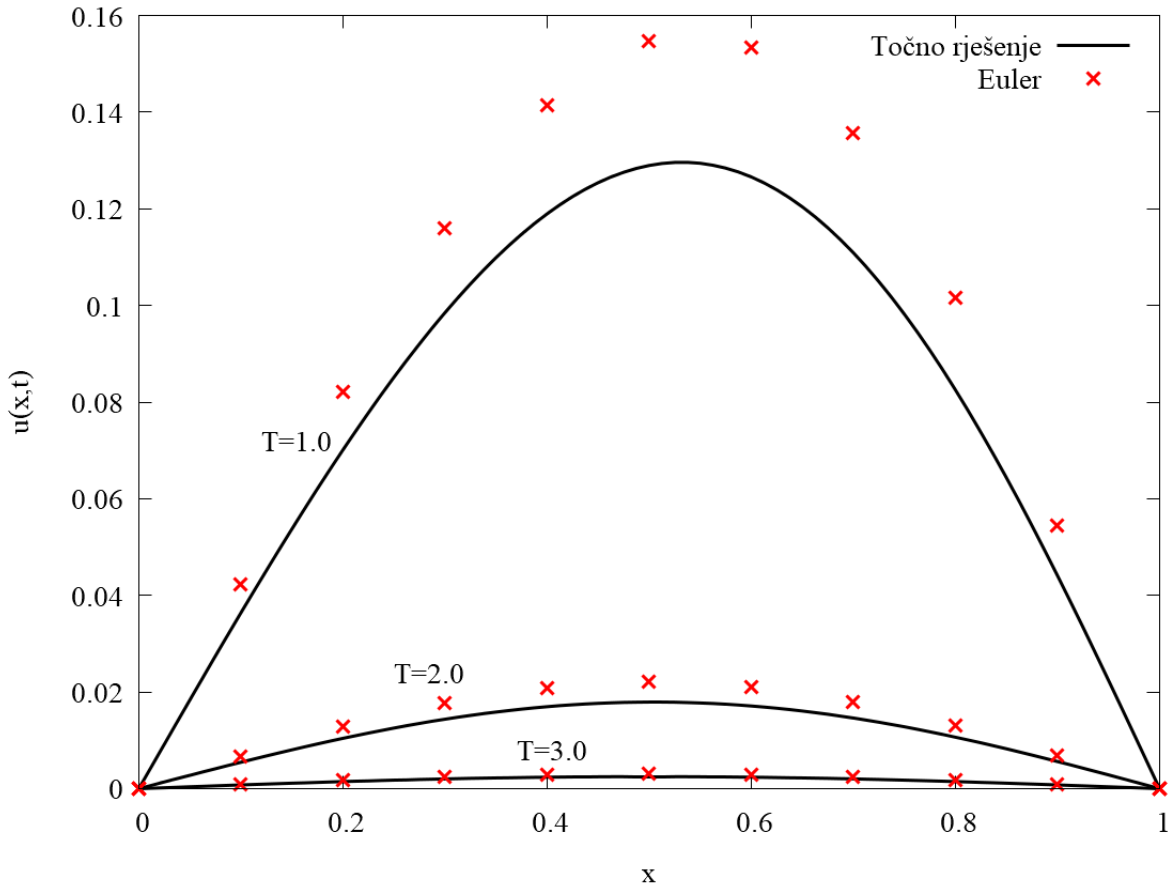
	T=1,0				**T=4,0**			
	Euler			Točno rješenje	Euler			Točno rješenje
x	N=10	N=20	N=40		N=10	N=20	N=40	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,07086	0,06845	nan	0,06632	0,00636	0,00517	nan	0,00453
0,2	0,14140	0,13602	nan	0,13121	0,01216	0,00987	nan	0,00864
0,3	0,21110	0,20142	nan	0,19279	0,01688	0,01367	nan	0,01196
0,4	0,27893	0,26237	nan	0,24804	0,02004	0,01619	nan	0,01415
0,5	0,34246	0,31444	nan	0,29192	0,02131	0,01717	nan	0,01498
0,6	0,39550	0,34905	nan	0,31607	0,02049	0,01647	nan	0,01436
0,7	0,42230	0,35090	nan	0,30809	0,01760	0,01412	nan	0,01230
0,8	0,38772	0,29834	nan	0,25372	0,01289	0,01032	nan	0,00898
0,9	0,24374	0,17586	nan	0,14606	0,00681	0,00545	nan	0,00474
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 6 Grafički prikaz dobivenih rješenja za primjer 1, slučaj 5, za N=20, $\Delta t = 0{,}01$

Tablica 13 Dobivene vrijednosti rješenja za primjer 1, slučaj 6

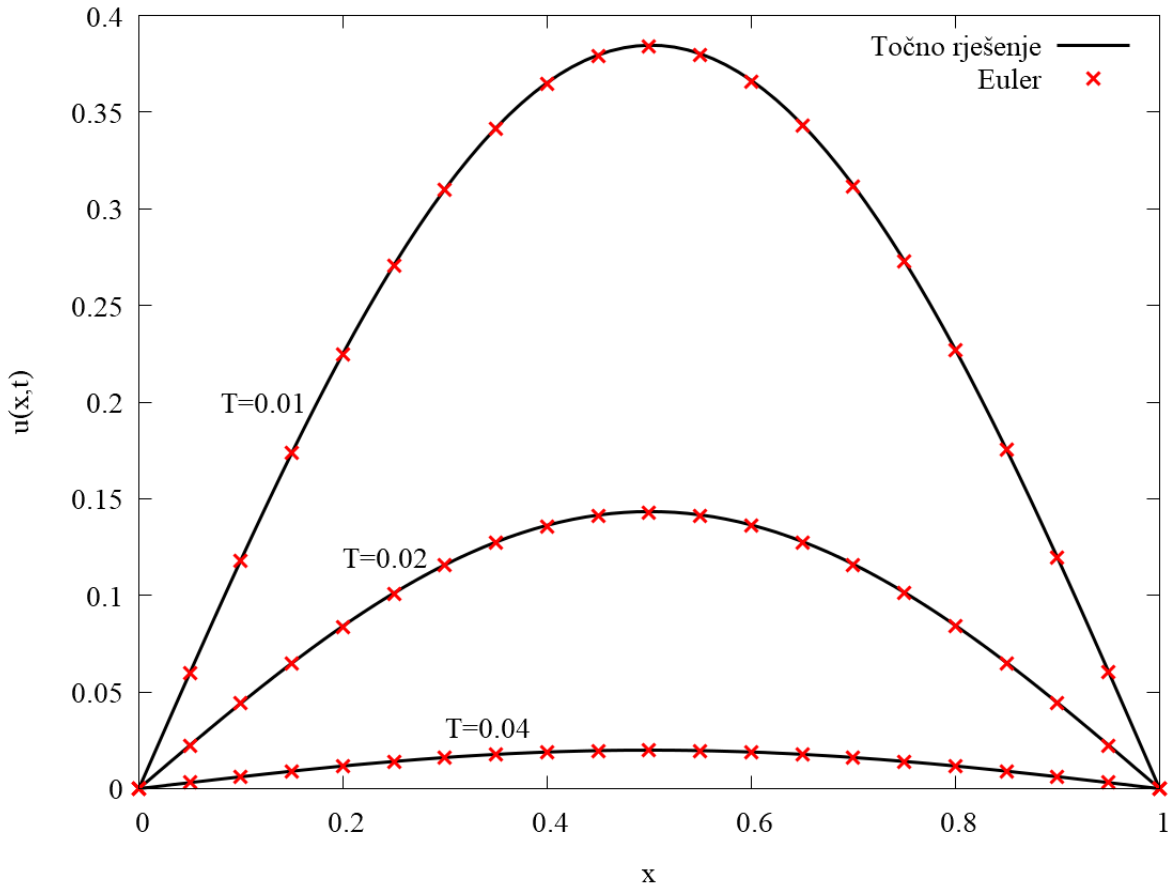
	T=1,0			**T=4,0**		
	Euler		Točno rješenje	Euler		Točno rješenje
x	N=10	N=20		N=10	N=20	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,04237	nan	0,03638	0,00013	nan	0,00011
0,2	0,08202	nan	0,07011	0,00025	nan	0,00020
0,3	0,11605	nan	0,09852	0,00034	nan	0,00028
0,4	0,14136	nan	0,11896	0,00040	nan	0,00033
0,5	0,15477	nan	0,12897	0,00043	nan	0,00035
0,6	0,15343	nan	0,12659	0,00040	nan	0,00033
0,7	0,13561	nan	0,11090	0,00034	nan	0,00028
0,8	0,10159	nan	0,08252	0,00025	nan	0,00020
0,9	0,05447	nan	0,04407	0,00013	nan	0,00011
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 7 Grafički prikaz dobivenih rješenja za primjer 1, slučaj 6, za N=10, $\Delta t = 0{,}01$

Tablica 14 Dobivene vrijednosti rješenja za primjer 2, slučaj 1

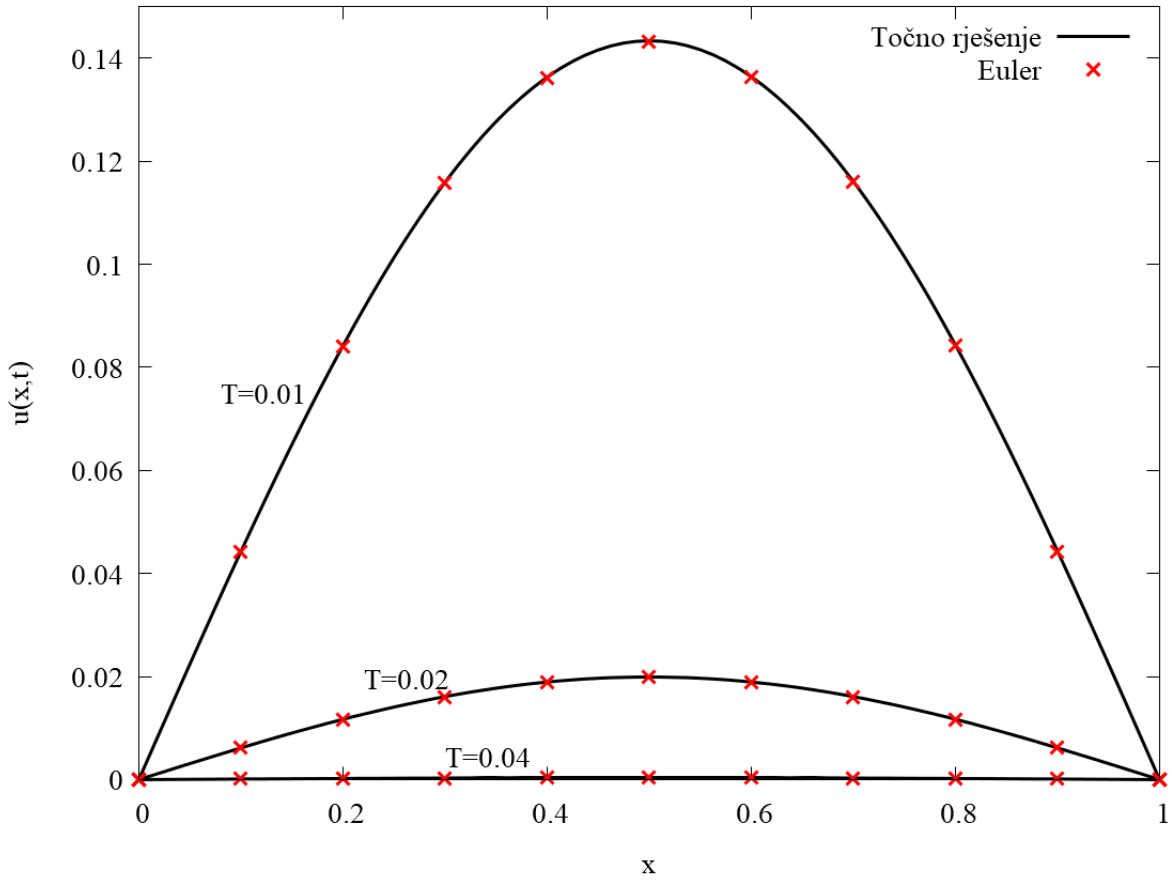
	T=0,01				**T=0,04**			
	Euler			Točno rješenje	Euler			Točno rješenje
x	N=10	N=20	N=40		N=10	N=20	N=40	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,11895	0,11808	nan	0,11827	0,00626	0,00610	nan	0,00615
0,2	0,22643	0,22478	nan	0,22514	0,01191	0,01159	nan	0,01170
0,3	0,31204	0,30975	nan	0,31023	0,01639	0,01596	nan	0,01611
0,4	0,36739	0,36468	nan	0,36522	0,01927	0,01876	nan	0,01894
0,5	0,38695	0,38408	nan	0,38464	0,02026	0,01973	nan	0,01991
0,6	0,36863	0,36588	nan	0,36641	0,01927	0,01877	nan	0,01894
0,7	0,31405	0,31169	nan	0,31215	0,01640	0,01597	nan	0,01611
0,8	0,22844	0,22672	nan	0,22706	0,01191	0,01160	nan	0,01171
0,9	0,12018	0,11928	nan	0,11946	0,00626	0,00610	nan	0,00616
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 8 Grafički prikaz dobivenih rješenja za primjer 2, slučaj 1, za N=20, $\Delta t = 0{,}0001$

Tablica 15 Dobivene vrijednosti rješenja za primjer 2, slučaj 2

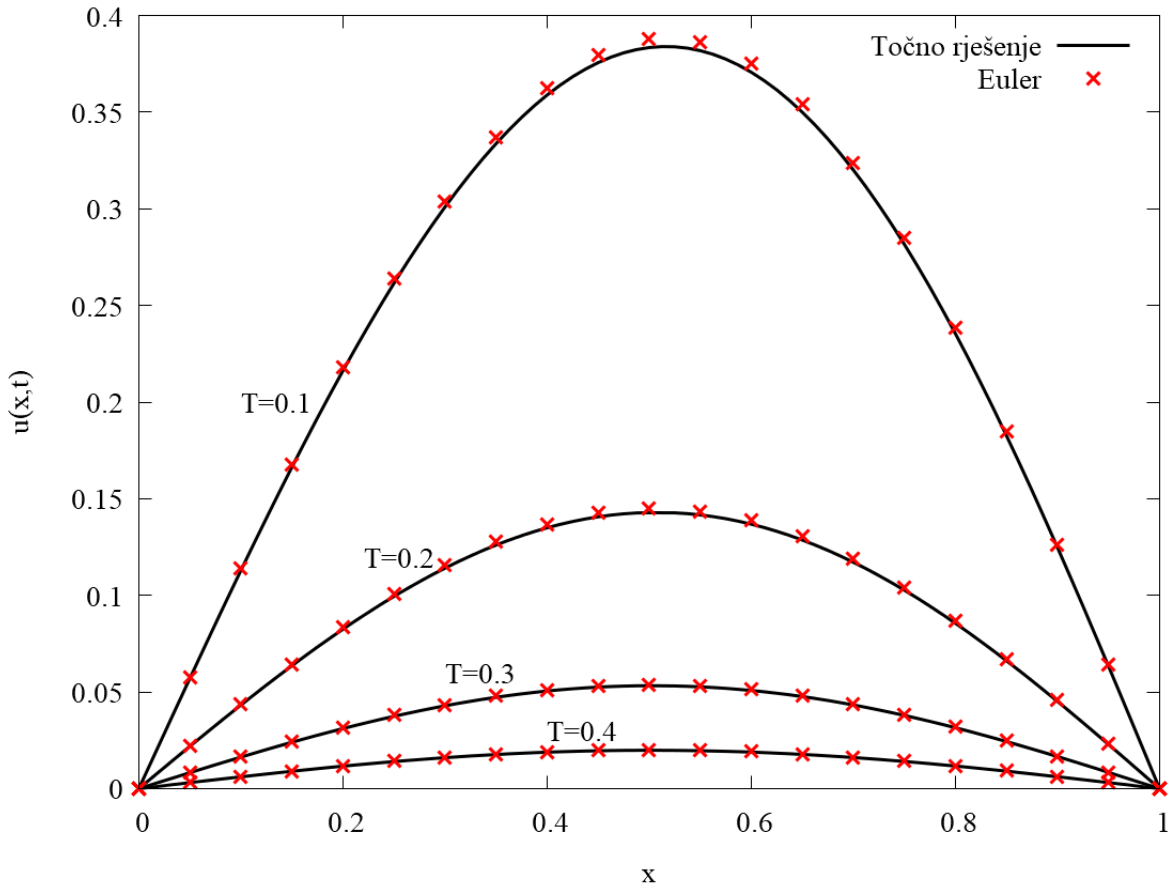
	T=0,01			**T=0,04**		
	Euler		Točno rješenje	Euler		Točno rješenje
x	N=10	N=20		N=10	N=20	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,04419	nan	0,04425	0,00012	nan	0,00012
0,2	0,08407	nan	0,08419	0,00022	nan	0,00023
0,3	0,11575	nan	0,11591	0,00031	nan	0,00031
0,4	0,13611	nan	0,13630	0,00036	nan	0,00037
0,5	0,14317	nan	0,14336	0,00038	nan	0,00038
0,6	0,13621	nan	0,13639	0,00036	nan	0,00037
0,7	0,11590	nan	0,11606	0,00031	nan	0,00031
0,8	0,08423	nan	0,08434	0,00022	nan	0,00023
0,9	0,04429	nan	0,04435	0,00012	nan	0,00012
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 9 Grafički prikaz dobivenih rješenja za primjer 2, slučaj 2, za N=10, $\Delta t = 0{,}0001$

Tablica 16 Dobivene vrijednosti rješenja za primjer 2, slučaj 3

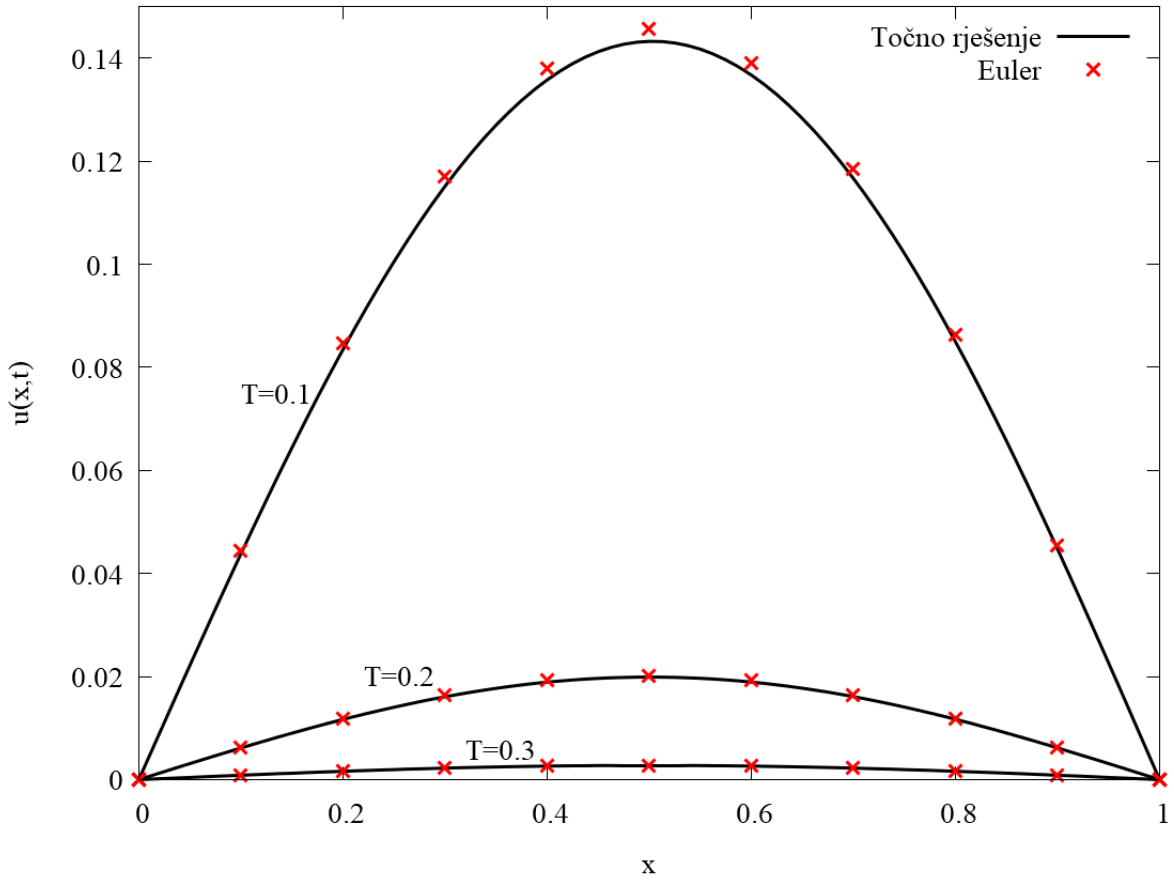
	T=0,1				**T=0,4**			
	Euler			Točno rješenje	Euler			Točno rješenje
x	N=10	N=20	N=40		N=10	N=20	N=40	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,11585	0,11389	nan	0,11289	0,00647	0,00618	nan	0,00612
0,2	0,22216	0,21826	nan	0,21625	0,01232	0,01176	nan	0,01164
0,3	0,30963	0,30394	nan	0,30097	0,01697	0,01619	nan	0,01603
0,4	0,36977	0,36265	nan	0,35886	0,01997	0,01905	nan	0,01886
0,5	0,39566	0,38770	nan	0,38342	0,02102	0,02005	nan	0,01985
0,6	0,38294	0,37497	nan	0,37066	0,02001	0,01909	nan	0,01890
0,7	0,33093	0,32387	nan	0,32007	0,01704	0,01625	nan	0,01609
0,8	0,24343	0,23819	nan	0,23537	0,01239	0,01182	nan	0,01170
0,9	0,12898	0,12620	nan	0,12472	0,00652	0,00622	nan	0,00615
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 10 Grafički prikaz dobivenih rješenja za primjer 2, slučaj 3, za N=20, $\Delta t = 0{,}001$

Tablica 17 Dobivene vrijednosti rješenja za primjer 2, slučaj 4

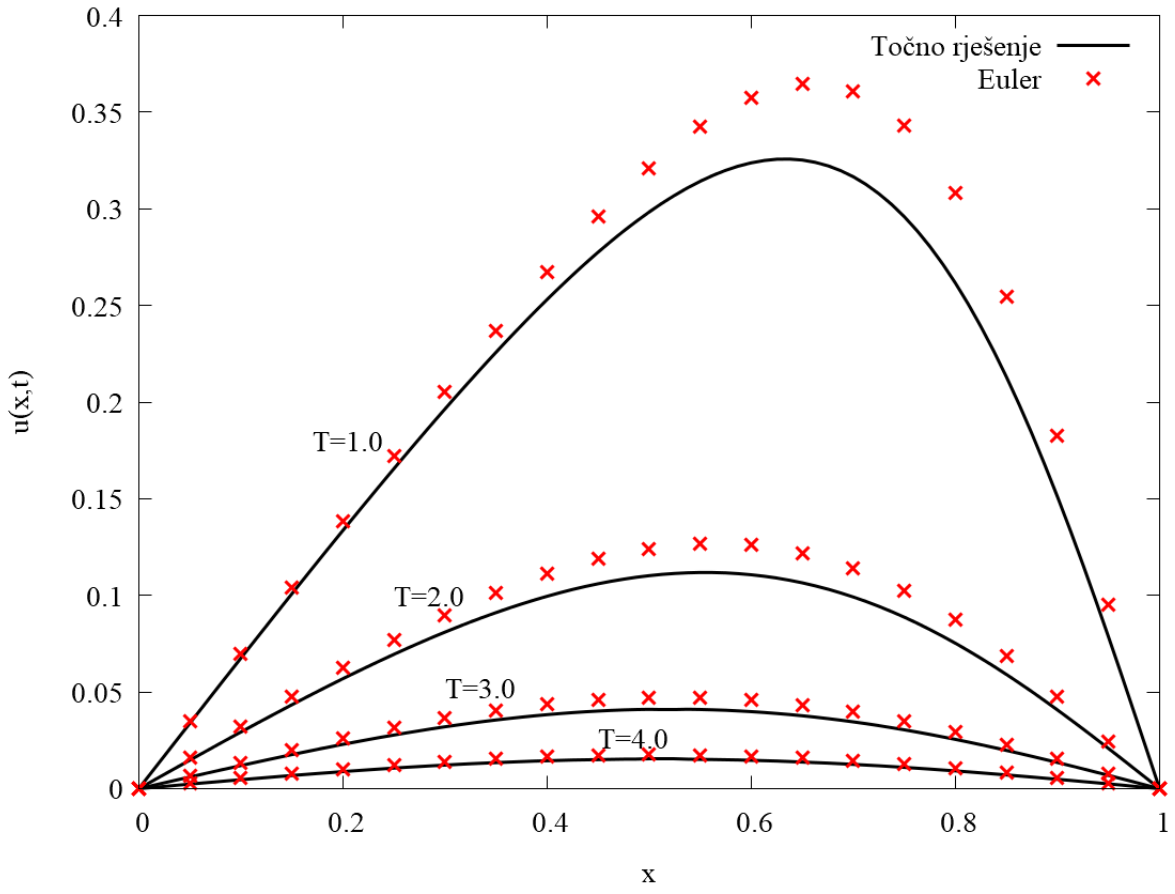
	T=0,1			**T=0,4**		
	Euler		Točno rješenje	Euler		Točno rješenje
x	N=10	N=20		N=10	N=20	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,04447	nan	0,04380	0,00012	nan	0,00012
0,2	0,08473	nan	0,08344	0,00023	nan	0,00023
0,3	0,11694	nan	0,11513	0,00031	nan	0,00031
0,4	0,13794	nan	0,13576	0,00037	nan	0,00037
0,5	0,14557	nan	0,14324	0,00039	nan	0,00038
0,6	0,13895	nan	0,13671	0,00037	nan	0,00037
0,7	0,11858	nan	0,11665	0,00031	nan	0,00031
0,8	0,08637	nan	0,08496	0,00023	nan	0,00023
0,9	0,04548	nan	0,04474	0,00012	nan	0,00012
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 11 Grafički prikaz dobivenih rješenja za primjer 2, slučaj 4, za N=10, $\Delta t = 0{,}001$

Tablica 18 Dobivene vrijednosti rješenja za primjer 2, slučaj 5

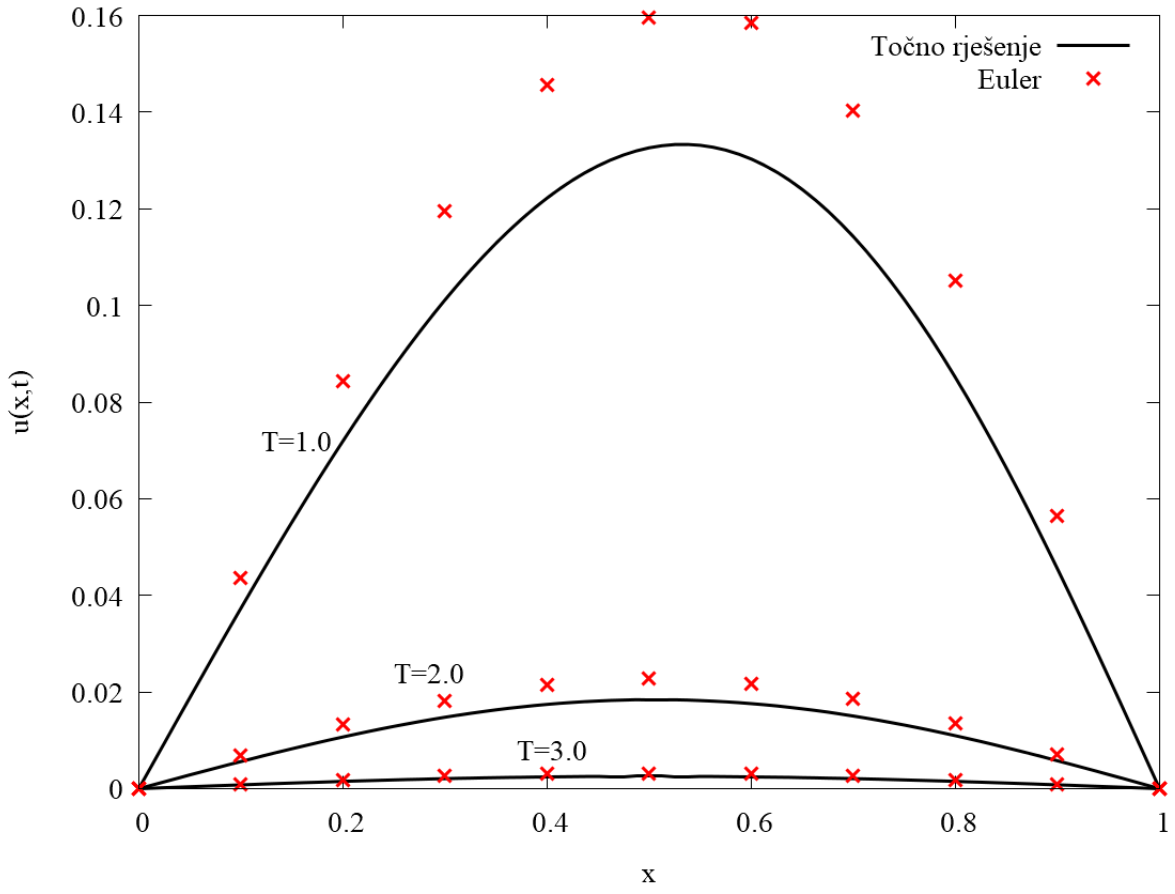
	T=1,0				**T=4,0**			
	Euler			Točno rješenje	Euler			Točno rješenje
x	N=10	N=20	N=40		N=10	N=20	N=40	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,07202	0,06963	nan	0,06750	0,00650	0,00528	nan	0,00462
0,2	0,14373	0,13841	nan	0,13362	0,01244	0,01009	nan	0,00882
0,3	0,21466	0,20511	nan	0,19647	0,01726	0,01397	nan	0,01221
0,4	0,28384	0,26748	nan	0,25307	0,02050	0,01655	nan	0,01445
0,5	0,34902	0,32117	nan	0,29834	0,02180	0,01756	nan	0,01531
0,6	0,40429	0,35752	nan	0,32379	0,02097	0,01685	nan	0,01467
0,7	0,43409	0,36083	nan	0,31656	0,01802	0,01444	nan	0,01256
0,8	0,40204	0,30821	nan	0,26154	0,01320	0,01056	nan	0,00918
0,9	0,25500	0,18240	nan	0,15097	0,00697	0,00557	nan	0,00484
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 12 Grafički prikaz dobivenih rješenja za primjer 2, slučaj 5, za N=20, $\Delta t = 0{,}01$

Tablica 19 Dobivene vrijednosti rješenja za primjer 2, slučaj 6

	T=1,0			**T=4,0**		
	Euler		Točno rješenje	Euler		Točno rješenje
x	N=10	N=20		N=10	N=20	
0,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,1	0,04354	nan	0,03732	0,00014	nan	0,00011
0,2	0,08433	nan	0,07194	0,00026	nan	0,00021
0,3	0,11942	nan	0,10115	0,00036	nan	0,00029
0,4	0,14562	nan	0,12223	0,00042	nan	0,00034
0,5	0,15964	nan	0,13262	0,00044	nan	0,00036
0,6	0,15848	nan	0,13030	0,00042	nan	0,00034
0,7	0,14026	nan	0,11424	0,00036	nan	0,00029
0,8	0,10520	nan	0,08507	0,00026	nan	0,00021
0,9	0,05644	nan	0,04546	0,00014	nan	0,00011
1,0	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000



Slika 13 Grafički prikaz dobivenih rješenja za primjer 2, slučaj 6, za N=10, $\Delta t = 0{,}01$

7. ZAKLJUČAK

Cilj ovog rada bio je približiti Burgersovu jednadžbu, njenu primjenu u fizici te numeričke metode za rješavanje iste. Jednadžba je opisana u viskoznom obliku, dok se neviskozni oblik može promatrati analogno, uz pojednostavljenje $\nu = 0$.

Pokazano je da se jednadžba može primijeniti za promatranje temeljnog, idealnog ponašanja fluida opisanog Navier-Stokes jednadžbama. Time jednadžba ima važnu ulogu jer se od promatranja idealnog slučaja dalje grade kompleksniji slučajevi.

Za rješavanje jednadžbe korištene su dvije numeričke metode: Eulerova metoda prema naprijed te metoda točnog rješenja. Eulerova metoda kreće od početnih i rubnih uvjeta te prema njima uzlazno „gradi“ ostale vrijednosti rješenja. Metoda točnog rješenja bazira se na Hopf-Cole transformaciji koja Burgersovu jednadžbu svodi na jednadžbu topline. Potom funkciju rješenja jednadžbe topline razvija u Fourierov red s Fourierovim koeficijentima, te se ponovno korištenjem transformacije dobije izraz za točno rješenje.

Promotrena su dva primjera, za dva različita početna uvjeta. U prvom primjeru, početni uvjet je sinusna funkcija, dok je u drugom primjeru početni uvjet polinom drugog reda. U svakom je primjeru promotreno po šest slučajeva koji se međusobno razliku za ulazne konstante: konstanta viskoznosti ν, vremenski korak *dt*, te ukupni vremenski interval $[0, T]$.

Dobivene vrijednosti za različite metode uspoređene su tablično i grafički. Uočeno je da Eulerova metoda, povećanjem konstante ν, pokazuje sve veći odmak od vrijednosti dobivene za točno rješenje. Na ukupnu grešku utječe i vremenski korak, kao i ukupni vremenski interval. Ipak, najveći utjecaj ima broj prostornih koraka, odnosno veličina istog. Smanjenjem prostornog koraka *dx*, odnosno povećanjem broja prostornih koraka *N*, dobije se znatno preciznije rješenje.

Račun se može poboljšati korištenjem naprednijih metoda kao što je Crank-Nicolson metoda. U toj metodi kombiniraju se Eulerova metoda prema naprijed (eksplicitna metoda), koja je korištena u ovom radu, te Eulerova metoda prema nazad (implicitna metoda). Metoda je kao takva stabilna za sve odabire vremenskog i prostornog koraka [10].

8. LITERATURA

[1] H. Bateman, *Some recent researches in motion of fluids, Mon. Weather Rev. 43*, 1915.

[2] P.A. Lagerstrom, J.D. Cole, L. Trilling, *Problems in the theory of viscous compressible fluids, Calif. Inst. Technol*. 1949.

[3] J.D. Cole, *On a quasilinear parabolic equation occurring in aerodynamics, Quart. Appl. Math. 9*, 1951., 225–236

[4] E. Hopf, *The partial differential equation* $u_t + uu_x = vu_{xx}$, *Commun. Pure Appl. Math. 3*, 1950., 201–230

[5] van Dommelen, L., *20.5 The inviscid Burgers' equation*, *Analysis in Mechanical Engineering*,
URL: http://www.eng.fsu.edu/~dommelen/aim/style_a/burgers.html (22.8.2019.)

[6] M.P. Bonkile, A. Awasthi, C. Lakshmi, V. Mukundan, V.S. Aswin, *A systematic literature review of Burgers' equation with recent advances*, 30.04.2018.,
URL: https://www.ias.ac.in/article/fulltext/pram/090/06/0069 (22.8.2019.)

[7] M. K. Kadalbajoo, A. Awasthi, *A numerical method based on Crank-Nicolson scheme for Burgers' equation*, Applied Mathematics and Computation 182, 2006.

[8] *Chapter 2 Burgers Equation*,
URL: https://www.uni-muenster.de/imperia/md/content/physik_tp/lectures/ss2017/numerische_Methoden_fuer_komplexe_Systeme_II/burgers.pdf (18.8.2019.)

[9] M. Landajuela, *Burgers equation*, 2011.,
URL:
http://www.bcamath.org/projects/NUMERIWAVES/Burgers_Equation_M_Landajuela.pdf (18.8.2019.)

[10] M. Hjorth-Jensen, *Computational physics*, 2010.

9. DODATAK

Kod za koeficijent A_0:

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define a 0
#define b 1
#define h 0.000000125
#define M (b-a)/h

double fja(double x, double v) { //funkcija vraca vrijednost podintegralne funkcije u
tocki x
	double vrij;
	vrij = exp((cos(PI*x) - 1) / (2 * v * PI));
	return vrij;
}

int main(void) {
	double A0[6], v[6], x;
	int i, j;
	FILE *tok;
	tok = fopen("Primjer_1_Vrijednosti_A0.txt", "w");
	fprintf(tok, "v\tA0\t\n");
	v[0] = 10;
	v[1] = 20;
	v[2] = 1.0;
	v[3] = 2.0;
	v[4] = 0.1;
	v[5] = 0.2;
	x = a;
	for (i = 0; i < 6; i++) {
		A0[i] = fja(a, v[i]) + fja(b, v[i]);
		for (j = 1; j <= M; j++) {
			x += h;
			if (j % 2 == 0)
				A0[i] += 2 * fja(x, v[i]);
			else
				A0[i] += 4 * fja(x, v[i]);
		}
		A0[i] *= h / 3;
		fprintf(tok, "%.2f\t%.6f\t\n", v[i], A0[i]);
	}
	fclose(tok);
	return 0;
}
```

Kod za točno rješenje:

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define v 10
#define A0 0.984273 //vrijednost dobivena programom za prethodno navedeni v
#define a 0.0
#define b 1.0
```

```
#define dy 0.00000125 //korak za racunanje integrala
#define K (b-a)/dy //broj koraka dy
#define dx 0.0125 //korak za racunanje rjesenja
#define M (b-a)/dx //broj koraka dx
#define dt 0.0001 //vremenski korak za racunanje rjesenja
#define T 0.04
#define N T/dt //broj koraka dt

//racunanje koeficijenta An
long double fjaAn(long double x, int n) { //racunanje vrijednosti podint. fje iz An
	long double vrijAn;
	vrijAn = 2 * exp((cos(PI*x) - 1) / (2 * v*PI))*cos(n*PI*x);
	return vrijAn;
}

long double fAn(int n) { //racunanje integrala An pomocu Simpsonove formule
	long double y, intAn;
	int i;
	intAn = fjaAn(a, n) + fjaAn(b, n);
	y = a;
	for (i = 1; i <= K; i++) {
		y += dy;
		if (i % 2 == 0)
			intAn += 2 * fjaAn(y, n);
		else
			intAn += 4 * fjaAn(y, n);
	}
	intAn *= dy / 3.0;
	return intAn;
}

int main(void) {
	long double x = a, t, ut, brojnik, nazivnik, An[50];
	int n, i, j; //i - prostorni korak, j - vremenski korak
	FILE *tok;
	tok = fopen("Primjer_1_Euler_slucaj_1.txt", "w");
	fprintf(tok, "x\tt\tu(x,t)\n");
	for (n = 0; n < 50; n++) {
		An[n] = fAn(n + 1);
	}
	for (i = 0; i <= M; i++) {
		x = i * dx;
		for (j = 0; j <= N; j++) {
			t = j * dt;
			if (t == 0.01 || t == 0.02 || t == 0.04) {
				brojnik = 0;
				nazivnik = A0;
				for (n = 1; n <= 50; n++) {//racunanje suma u brojn. i naz.
					brojnik += An[n-1]*exp(-n*n*PI*PI*v*t)*n*sin(n*PI*x);
					nazivnik += An[n-1]*exp(-n*n*PI*PI*v*t)*cos(n*PI*x);
				}
				ut = (2 * PI*v)*brojnik / nazivnik;
				fprintf(tok, "%.4lf\t%.2lf\t%.5lf\t\n", x, t, ut);
			}
		}
	}
	fclose(tok);
	return 0;
}
```

Kod za Eulerovu metodu:

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define v 10.0
#define dt 0.0001 //vremenski korak
#define T 0.04 //ukupno vrijeme
#define M 400 //broj koraka dt M=T/dt
#define dx 0.1 //prostorni korak
#define N 10 //broj koraka dx N=1/dx
#define r (v*dt)/(dx*dx)
#define p dt/dx


int main(void) {
	long double x, t, u[N+2][M+1];
	int i, j; //i-brojac za prostorni korak, j-brojac za vremenski korak
	FILE *tok;
	tok = fopen("Primjer_1_Euler_slucaj_1_N=10.txt", "w");
	fprintf(tok, "x\t\tu(x,t=0.01)\tu(x,t=0.02)\tu(x,t=0.04)\n");

	for (i = 1; i <= N; i++) { //postavljanje vrijednosti pocetnog uvjeta
		x = i * dx;
		u[i][0] = sin(PI*x);
	}

	for (j = 0; j <= M; j++) { //postavljanje vrijednosti rubnog uvjeta
		u[0][j] = 0.0;
		u[N][j] = 0.0;
	}

	for (j = 0; j < M; j++) {
		for (i = 1; i < N; i++) {
		 u[i][j+1]=u[i][j]*(1-p*u[i+1][j]+p*u[i][j]-2*r)+r*(u[i+1][j]+u[i-1][j]);
		}
	}

	x = 0.0;
	for (i = 0; i <= N; i++) {
		fprintf(tok, "%lf\t", x);
		x = x + dx;
		for (j = 0; j <= M; j++) {
			t = j * dt;
			if (t == 0.01 || t == 0.02 || t == 0.04)
				fprintf(tok, "%lf\t", u[i][j]);

		}
		fprintf(tok, "\n");
	}
	fclose(tok);
	return 0;
}
```