

Parabolička jednačina u 1D

Sadržaj

Formulacija problema.....	1
Aproksimacija metodom konačnih razlika.....	1
Test primer.....	2
Formiranje mreže.....	3
Matrica sistema.....	3
Desna strana sistema.....	5
Rešenje pomoću ugrađenog matlabovog operatora \.....	6
Numeričko potvrđivanje reda konvergencije kod eksplicitnih metoda.....	6

Formulacija problema

Neka je $\Omega = (a, b)$ i $I = (0, T)$ za neko $T > 0$. Posmatramo jednačinu

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f(x, t), \quad x \in \Omega, t \in I,$$

sa graničnim uslovima Dirihleovog tipa

$$u(a, t) = u_1(t), \quad u(b, t) = u_2(t), \quad t \in \bar{I},$$

i početnim uslovom

$$u(x, 0) = u_0(x), \quad x \in \bar{\Omega}.$$

Aproksimacija metodom konačnih razlika

Neka je $N > 1$, $M > 0$ i $h = (b - a)/N$, $\tau = T/M$. Uvodimo sledeće ravnomerne mreže

$$\bar{\omega}_h = \{x_i = a + ih, i = 0, 1, \dots, N\},$$

$$\bar{\omega}_\tau = \{t_j = j\tau, j = 0, 1, \dots, M\},$$

i definišemo

$$\omega_h := \bar{\omega}_h \cap \Omega,$$

$$\omega_\tau := \bar{\omega}_\tau \cap (0, T), \quad \omega_\tau^+ = \omega_\tau \cup \{T\},$$

$$\omega = \omega_h \times \omega_\tau.$$

Početno-granični problem aproksimiramo sledećom diferencijskom shemom

$$v_{t,i}^j = (v_{xx,i}^{j+1} + f_i^{j+1})\sigma + (v_{xx,i}^j + f_i^j)(1 - \sigma), \quad i = 1, 2, \dots, N - 1, j = 0, 1, \dots, M - 1,$$

$$v_0^j = u_1^j, \quad v_n^j = u_2^j, \quad j = 0, 1, \dots, M,$$

$$v_i^0 = u_0(x_i), \quad i = 0, 1, \dots, N,$$

gde parametar $\sigma \in [0, 1]$ nazivamo težinskim parametrom, a shemu težinskom shemom. Ako je

- $\sigma = 0$ shema je eksplicitna, $O(\tau + h^2)$
- $\sigma = 1$ implicitna, $O(\tau + h^2)$
- $\sigma = 1/2$ Crank-Nicolson-ova shema, $O(\tau^2 + h^2)$.

Shema je apsolutno stabilna ako je $\sigma \geq 1/2$, dok za $0 \leq \sigma < 1/2$ konvergira pod uslovom da je

$$\tau \leq \frac{h^2}{2 - 4\sigma}.$$

Test primer

Ponovo, za prvi test primer ćemo izabrati zadatak za koji će aproksimacija dati tačno rešenje.

Izaberimo interval $\Omega = (0, 2)$ i $T = 1$, a kao tačno rešenje

$$u(x, t) = tx^2.$$

Tada je

$$u_1(t) = 0, \quad u_2(t) = 2t, \quad u_0(x) = 0$$

i

$$f(x, t) = x^2 - 2t.$$

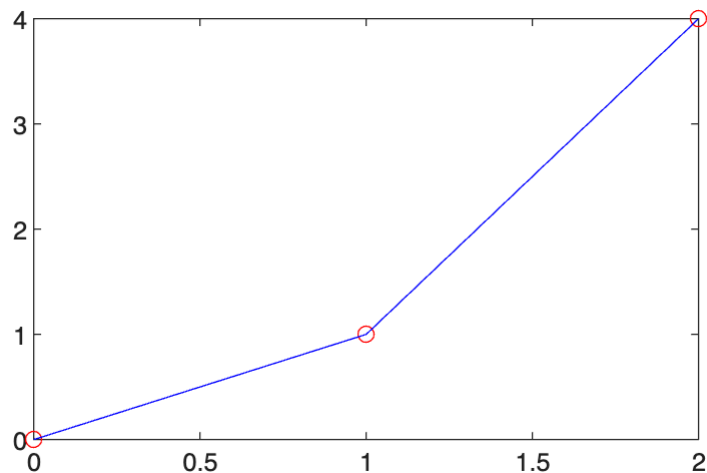
Ove podatke ćemo sačuvati u skript fajlu, sa tim da pazimo da funkcije koje su jednake konstanti moraju vratiti vektor iste dužine kao i prosleđeni argument:

```
%primerP1.m
a=0;
b=2;
T=1;
u1=@(t) 0*t;%zeros(size(t));
u2=@(t) 4*t;
u0=@(x) 0*x;
f=@(x,t) x.^2-2*t;

ue=@(x,t) t.*x.^2;
```

Funkciju koja implementira metodu ćemo nazvati *fdmP* a kao ulazne argumente ćemo izabrati $(primer, si, h, dt)$, gde je $dt = \tau, si = \sigma$.

```
fdmP('primerP1',1,1,1);
```



Formiranje mreže

Ovaj deo ide standardno. Nakon učitavanja podatak neophodno je da formiramo dve mreže, jednu po prostornoj promenljivoj i drugu po vremenskoj. Biće nam potreban i broj tačaka svake mreže.

```
%% mreza
xosa=a:h:b;
nx=length(xosa);
tosa=0:dt:T;
nt=length(tosa);
```

Matrica sistema

Sa ciljem da odredimo matricu sistema linearnih jednačina, u shemi

$$v_{t,i}^j = (v_{xx,i}^{j+1} + f_i^{j+1})\sigma + (v_{xx,i}^j + f_i^j)(1 - \sigma), \quad i = 1, 2, \dots, N - 1, j = 0, 1, \dots, M - 1,$$

prvo razvijmo konačnu razliku po vremenu

$$\frac{v_i^{j+1} - v_i^j}{\tau} = (v_{xx,i}^{j+1} + f_i^{j+1})\sigma + (v_{xx,i}^j + f_i^j)(1 - \sigma), \quad i = 1, 2, \dots, N - 1, j = 0, 1, \dots, M - 1,$$

pa prebacujemo sve nepoznate (vrednosti sa vremenskog sloja $j + 1$) na levu stranu jednakosti i sve poznate na desnu:

$$v_i^{j+1} - \tau\sigma v_{xx,i}^{j+1} = \tau\sigma f_i^{j+1} + \tau(v_{xx,i}^j + f_i^j)(1 - \sigma) + v_i^j, \quad i = 1, 2, \dots, N - 1, j = 0, 1, \dots, M - 1.$$

Označimo desnu stranu sa G_i^j . Nakon razvijanja konačne razlike drugog reda na levoj strani jednakosti dobijamo

$$-\frac{\tau\sigma}{h^2}v_{i-1}^{j+1} + \left(1 + 2\frac{\tau\sigma}{h^2}\right)v_i^{j+1} - \frac{\tau\sigma}{h^2}v_{i+1}^{j+1} = G_i^j, \quad i = 1, 2, \dots, N-1, j = 0, 1, \dots, M-1.$$

Neka je $c = \frac{\tau\sigma}{h^2}$. Dobili smo sistem linearnih jednačina

$$\begin{bmatrix} 1 & & & & & \\ -c & (1+2c) & -c & & & \\ & -c & (1+2c) & -c & & \\ & & \ddots & \ddots & \ddots & \\ & & & -c & (1+2c) & -c \\ & & & & -c & (1+2c) & -c \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} v_0^{j+1} \\ v_1^{j+1} \\ v_2^{j+1} \\ \vdots \\ v_{N-2}^{j+1} \\ v_{N-1}^{j+1} \\ v_N^{j+1} \end{bmatrix} = \begin{bmatrix} u_1^j \\ G_1^j \\ G_2^j \\ \vdots \\ G_{N-2}^j \\ G_{N-1}^j \\ u_2^j \end{bmatrix}$$

Kraće ga možemo zapisati sa

$$Av = G.$$

Očigledno, ovo je trodijagonalan sistem jednačina. Umesto da formiramo celu matricu, konstruisaćemo samo njene dijagonale i to ćemo raditi na sledeći način. Formiraćemo tri vektora d, du, dd koji će predstavljati vektore elemenata na glavnoj dijagonali, prvoj naddijagonali i prvoj poddijagonali, respektivno. Svi će biti iste dužine, pa ćemo ih u matricu A smeštati na sledeći način

$$\begin{array}{c|cccccc} dd_1 & d_1 & du_1 & & & & \\ & dd_2 & d_2 & du_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & dd_{N-1} & d_{N-1} & du_{N-1} & \\ & & & & dd_N & d_N & du_N \end{array}$$

U skladu sa prethodnim, u Matlabu ove vektore formiramo sa

```
c=dt*si/h^2;
d=(1+2*c)*ones(nx,1);%glavna dijagonala
d(1)=1; d(nx)=1;%popravka
du=-c*ones(nx,1);%prva naddijagonala
dd=du;%prva poddijagonala
du(1)=0;%popravka
du(nx)=0;%ovaj element ne ucestvuje u sistemu, moze biti bilo sta
dd(nx)=0;%popravka
dd(1)=0;%ovaj element ne ucestvuje u sistemu, moze biti bilo sta
```

Tada je cela matrica

```
A=diag(d)+diag(du(1:nx-1),1)+diag(dd(2:nx),-1);
disp(A)
```

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 3 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Desna strana sistema

Na svakom novom vremenskom sloju desna strana sistema se menja, tj. imamo za

$i = 1, 2, \dots, N-1, j = 0, 1, \dots, M-1$ da je

$$G_i^j = \tau \sigma f_i^{j+1} + \tau (v_{xx,i}^j + f_i^j)(1 - \sigma) + v_i^j.$$

Kada razvijemo kanačnu razliku dobijamo

$$G_i^j = \tau (\sigma f_i^{j+1} + (1 - \sigma) f_i^j) + \left[\frac{\tau(1 - \sigma)}{h^2} v_{i-1}^j + \left(1 - 2 \frac{\tau(1 - \sigma)}{h^2} \right) v_i^j + \frac{\tau(1 - \sigma)}{h^2} v_{i+1}^j \right] + v_i^j.$$

Neka je $c1 = \frac{\tau(1 - \sigma)}{h^2}$. Primetimo da izraz u uglastoj zagradi možemo predstaviti množenjem trodijagonalne

matrice, označićemo je sa B , sa aproksimacijom sa vremenskog sloja j :

$$\begin{bmatrix} 1 & & & & & & \\ c1 & (1-2c1) & c1 & & & & \\ & c1 & (1-2c1) & c1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & c1 & (1-2c1) & c1 & \\ & & & & c1 & (1-2c1) & c1 \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} v_0^j \\ v_1^j \\ v_2^j \\ \vdots \\ v_{N-2}^j \\ v_{N-1}^j \\ v_N^j \end{bmatrix} = \begin{bmatrix} v_0^j \\ c1v_0^j + (1-2c1)v_1^j + c1v_2^j \\ c1v_1^j + (1-2c1)v_2^j + c1v_3^j \\ \vdots \\ c1v_{N-3}^j + (1-2c1)v_{N-2}^j + c1v_{N-1}^j \\ c1v_{N-2}^j + (1-2c1)v_{N-1}^j + c1v_N^j \\ v_N^j \end{bmatrix}$$

To ćemo iskoristiti u kodu, jer nam omogućava da desnu stranu izračunamo bez dodatne for petlje po prostornim indeksima.

Imajući u vidu,

$$v_i^{j+1} - \tau \sigma v_{xx,i}^{j+1} = v_i^j + \tau(1 - \sigma) v_{xx,i}^j + (1 - \sigma) f_i^j + \sigma f_i^{j+1} = G_i^j,$$

može se uočiti neka logička veza radi lakšeg pamćenja matrice B ako znamo matricu A .

```
c1=dt*(1-si)/h^2;
d1=(1-2*c1)*ones(nx,1);%glavna dijagonala
d1(1)=1; d1(nx)=1;%popravka
d1u=c1*ones(nx,1);%prva naddijagonala
d1d=d1u;%prva poddijagonala
d1u(1)=0;
d1d(nx)=0;
B=diag(d1)+diag(d1u(1:nx-1),1)+diag(d1d(2:nx),-1);
disp(B)
```

1	0	0
0	1	0
0	0	1

Rešenje pomoću ugrađenog matlabovog operatora \

Rešenje ćemo upisivati u matricu

```
v=zeros(nx,nt);
```

u kojoj svakoj koloni odgovara novi vremenski sloj. Na početku postavljamo početnu vrednost problema u prvu kolonu matrice

```
v(:,1)=u0(xosa);
```

Nakon ovoga možemo ući u for petlju i izračunavati vrednosti na novim slojevima. U svakoj iteraciji koristimo vrednosti funkcije f na dva vremenska sloja j i $j + 1$. Imamo dve opcije za njihovo izračunavanje. Prva je da sve vrednosti izračunamo pre for petlje i upišemo u jednu matricu, ali to zahteva formiranje mreže sa *ndgrid* i korišćenje više radnog memorijskog prostora. Druga opcija je da izračunavamo vrednosti samo na dva neophodna vremenska sloja j i $j + 1$. Tako ćemo i uraditi. Vrednosti sa vremenskog j čuvaćemo u vektoru $F1$, a vrednosti sa vremenskog sloja $j + 1$ u vektoru $F2$.

```
U1=u1(tosa);%levi granicni uslov
U2=u2(tosa);%desni granicni uslov
F1=f(xosa,tosa(1))';
for j=1:nt-1
    F2=f(xosa,tosa(j+1))';
    G=dt*(si*F2+(1-si)*F1)+B*v(:,j);
    %popravka desne strane, treba da je jednaka granicnim uslovima:
    G(1)=U1(j+1);G(nx)=U2(j+1);
    v(:,j+1)=A\G;%resenje
    F1=F2; %sada F1 postaje F2, tj. vrednost sa starog sloja
end
disp(v)
```

0	0
0	1
0	4

Numeričko potvrđivanje reda konvergencije kod eksplicitnih metoda

Konvergencija eksplicitne metoda je $O(\tau + h^2)$. Međutim, ako prethodne obrasce primenimo na nju, nećemo dobiti dobar rezultat:

```
redm=redt(@fdmP,'primerP2',0)
```

```
Warning: Shema je nestabilna!
Warning: Shema je nestabilna!
Warning: Shema je nestabilna!
```

Warning: Shema je nestabilna!

Warning: Shema je nestabilna!

Warning: Shema je nestabilna!

redm = 1×5

-28.0677 -71.8397 -135.7823 -255.5886 -479.3352

Problem je u tome što je eksplicitna metoda samo uslovno stabilne, odnosno, shema sa težinom, za $0 \leq \sigma < 1/2$

konvergira samo pod uslovom da je $\tau \leq \frac{h^2}{2 - 4\sigma}$. Dakle, moramo obezbediti da ovaj uslov bude ispunjen.

Stoga, stavimo da je $K = 2 - 4\sigma$ i izaberimo korake tako da je

$$K\tau = h^2 = p,$$

gde je p neka vrednost koju ćemo mi birati i u odnosu na nju određivati korake τ i h .

Tada je

$$z_{h,\tau} = \mathcal{O}(h^2 + \tau) = C_x p + C_t p/K = (C_x + C_t/K)p = C_p p = \mathcal{O}(p) = z_p, \quad C_x, C_t, K, C_p = \text{const.}$$

Za izbor vrednosti parametra p , izabraćemo najpraktičniji pristup, da se svaka nova manja vrednost može jednostavno korenovati. Na primer, ako za prvi korak izaberemo

$$p = 2^{-2},$$

sledeći će biti

$$p/4 = 2^{-4}. \text{ Tada je}$$

$$\frac{z_p}{z_{p/4}} = \frac{C_p p}{C_p p/4} = 4.$$

Neka je $K^k \tau^k = h^{2k} = p^k$, tada je

$$\frac{z_p}{z_{p/4}} = \frac{C_p p^k}{C_p p^k/4^k} = 4^k = 2^{2k}.$$

Ako logaritmujemo prethodni izaz sa bazom 2, dobijamo

$$\log_2 \frac{z_p}{z_{p/4}} = 2k.$$

Dakle, logaritam po bazi 2 greške, sa koracima izabranim po prethodnom obrascu nam daje red konvergencije po prostornoj promenljivoj $2k$, dok je polovina te vrednosti red konvergencije po vremenskoj promenljivoj.

Ovo ćemo implementirati u funkciji `redtxe(fun, primer, si)`

```
redm=redtxe(@fdmP, 'primerP2', 0)
```

redm = 1×7

2.1033 2.0257 2.0064 2.0016 2.0004 2.0001 2.0000

```
redm=redtxe(@fdmP, 'primerP2', 1/4)
```

```
redm = 1×7  
2.1002    2.0263    2.0066    2.0016    2.0004    2.0001    2.0000
```

Ovaj postupak možemo primeniti i za $1/2 \leq \sigma \leq 1$, samo tada ne možemo birati $K = 2 - 4\sigma$, jer ćemo dobiti negativan korak. Međutim, kako je u ovom slučaju shema apsolutno stabilna, za K možemo izabrati bilo šta, pa je onda najjednostavnije izabrati da je $K = 1$.

```
redm=redtxe(@fdmP,'primerP2',1/2)
```

```
redm = 1×7  
2.1200    2.0292    2.0073    2.0018    2.0005    2.0001    2.0000
```