Metode Numerice în Mecanică - Test 2

Corneliu Meseșan

22 ianuarie 2023

Problema 1

Solve the initial boundary value problem for $u_t = u_{xx}$ on $-1 \le x \le 1$ for $0 \le t \le 0.5$ with initial data given by

$$u_0(x) = \begin{cases} 1 - |x| & \text{for } |x| < \frac{1}{2}, \\ \frac{1}{4} & \text{for } |x| = \frac{1}{2}, \\ 0 & \text{for } |x| = \frac{1}{2}. \end{cases}$$

Use the boundary conditions

$$u(t,-1) = u^*(t,-1) \text{ and } u_x(t,1) = 0,$$

where $u^*(t,x)$ is the exact solution given by

$$u^*(t,x) = \frac{3}{8} + \sum_{\ell=0}^{\infty} \left(\frac{(-1)^\ell}{\pi(2\ell+1)} + \frac{2}{\pi^2(2\ell+1)^2} \right) \cos \pi(2\ell+1)x \; e^{-\pi^2(2\ell+1)^2 t}$$
$$+ \sum_{m=0}^{\infty} \frac{\cos 2\pi(2m+1)x}{\pi^2(2m+1)^2} \, e^{-4\pi^2(2m+1)^2 t}.$$

Descrierea metodei

John Crank (1916 - 2006)

Phyllis Nicolson (1917 - 1968)

Metoda Crank-Nicolson este o metodă implicită propusă de matematicienii britanici John Crank și Phyllis Nicolson în 1947. Această metodă folosește regula trapezului în timp și o discretizare cu diferențe centrale în spațiu ceea ce conduce la o schemă cu acuratețe de ordinul doi.

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \frac{1}{2} \left(\frac{U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}}{(\Delta x)^2} + \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{(\Delta x)^2} \right).$$

Schema anterioară se reduce la rezolvarea unui sistem liniar tridiagonal de forma:

$$-\nu U_{i+1}^{n+1} + 2(1+\nu)U_i^{n+1} - \nu U_{i-1}^{n+1} = \nu U_{i+1}^n + 2(1-\nu)U_i^n + \nu U_{i-1}^n$$

căruia i se adaugă condiția inițială

$$u_0(x) = \begin{cases} 1 - |x| & \text{for } |x| < \dfrac{1}{2}, \\ \dfrac{1}{4} & \text{for } |x| = \dfrac{1}{2}, \\ 0 & \text{for } |x| = \dfrac{1}{2} \end{cases}$$

și condițiile pe frontieră

$$U_0^n = U_{N_x}^n = 0, n = 0, 1, 2, \dots.$$

Implementarea metodei

Implementarea metodei se face prin inițializarea valorilor Δx, Δt, $\nu = \dfrac{\Delta t}{(\Delta x)^2}$, N_x, a timpului final dorit și a intervalului pentru x. Apoi se rezolvă sistemul de forma $Ax = B$.

```
dt=0.001; dx=0.02; niu=dt/(dx*dx);
Nx=2/(dx)+1; x=-1:dx:1;
tf=0.5; Nt=tf/dt;
Uo=zeros(Nx,1);
Un=zeros(Nx,1);
for i=1:Nx
    if abs(x(i))<0.5
        Uo(i)=1-abs(x(i));
    elseif abs(x(i))==0.5
        Uo(i)=0.25;
    else
        Uo(i)=0;
    end
end
%metoda Crank-Nicolson
A=zeros(Nx,Nx);
b=zeros(Nx,1);
n=0;
b(1)=solutiaExacta(-1,tf);
while (n<Nt)
    n=n+1;
    A(1,1)=1;
    for i=2:Nx-1
        A(i,i-1)=-niu;
        A(i,i)=2+2*niu;
        A(i,i+1)=-niu;
        b(i)=niu*Uo(i-1)+(2-2*niu)*Uo(i)+niu*Uo(i+1);
    end
    A(Nx,Nx)=1;
    A(Nx,Nx-1)=-1;
    b(Nx)=0;
    Un=A\b;
    Uo=Un;
end
```

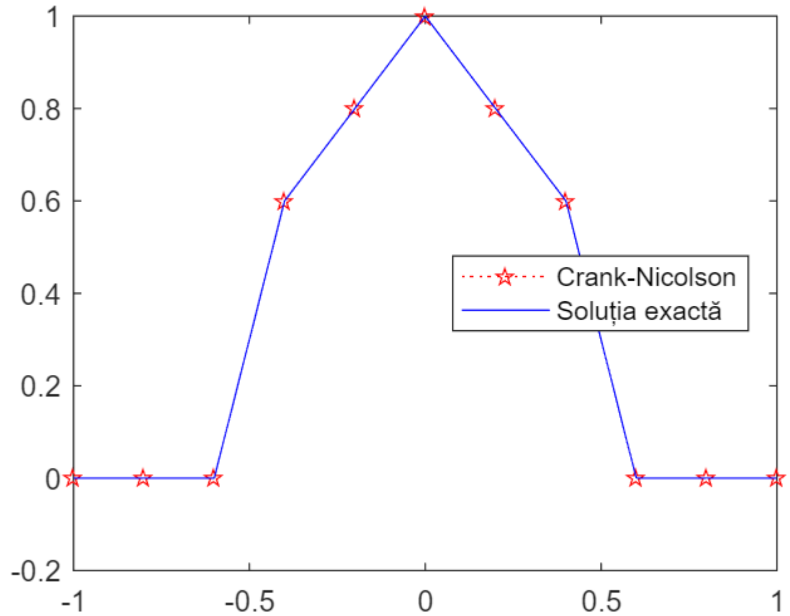
Eroarea

Erorile sunt calculate folosind comanda `norm` din MATLAB, cu setarea `Inf`, pentru a obține norma Cebîșev.

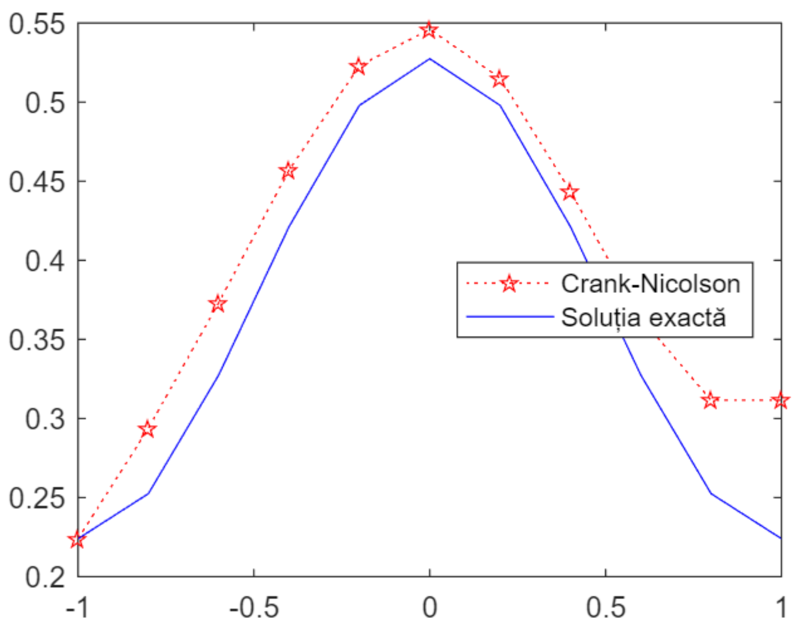
```
fprintf('Eroarea este %.16e.', norm(Ue'-Uo,Inf))
```

Pentru `tf` $= 0$ și `dt` $= 0.01$, `dx` $= 0.2$ eroarea maximă este de $\approx 6.2445604243066555 \cdot 10^{-12}$.
Pentru `tf` $= 0.125$ și `dt` $= 0.01$, `dx` $= 0.2$ eroarea maximă este de $\approx 8.7565420933363780 \cdot 10^{-2}$.
Pentru `tf` $= 0.25$ și `dt` $= 0.01$, `dx` $= 0.2$ eroarea maximă este de $\approx 6.7364804909225862 \cdot 10^{-2}$.
Pentru `tf` $= 0.375$ și `dt` $= 0.01$, `dx` $= 0.2$ eroarea maximă este de $\approx 6.2801342077688338 \cdot 10^{-2}$.
Pentru `tf` $= 0.5$ și `dt` $= 0.01$, `dx` $= 0.2$ eroarea maximă este de $\approx 6.0804559749061349 \cdot 10^{-2}$.
Pentru `tf` $= 1$ și `dt` $= 0.01$, `dx` $= 0.2$ eroarea maximă este de $\approx 4.8407940938600624 \cdot 10^{-2}$.
Pentru `tf` $= 0.5$ și `dt` $= 0.001$, `dx` $= 0.02$ eroarea maximă este de $\approx 3.1015713104709852 \cdot 10^{-2}$.

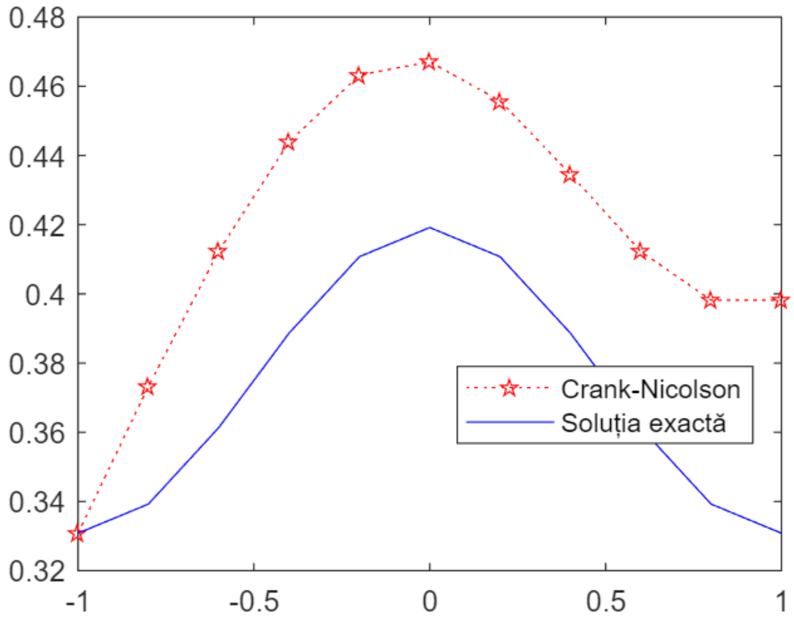
Graficele



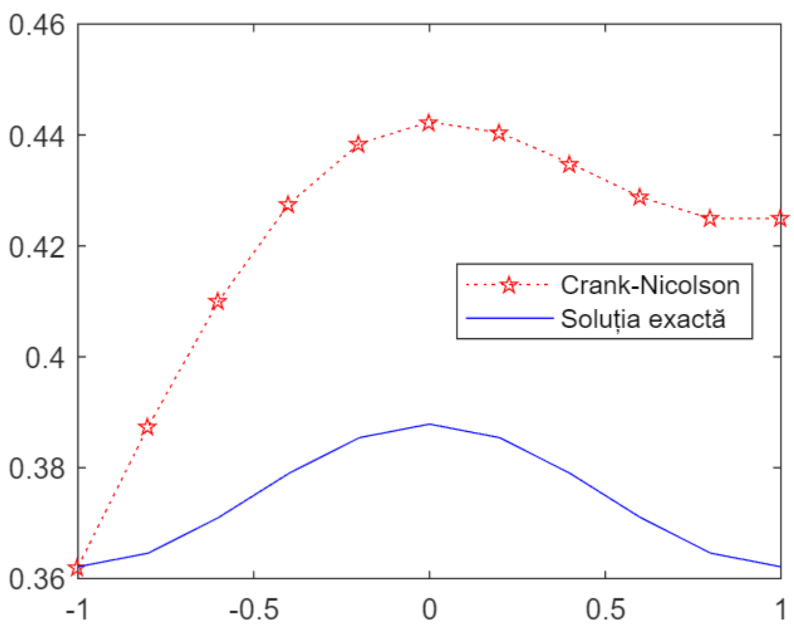
Soluția numerică obținută cu ajutorul metodei Crank-Nicolson și soluția exactă, $\texttt{tf} = 0$ **și** $\texttt{dt} = 0.01, \texttt{dx} = 0.2$



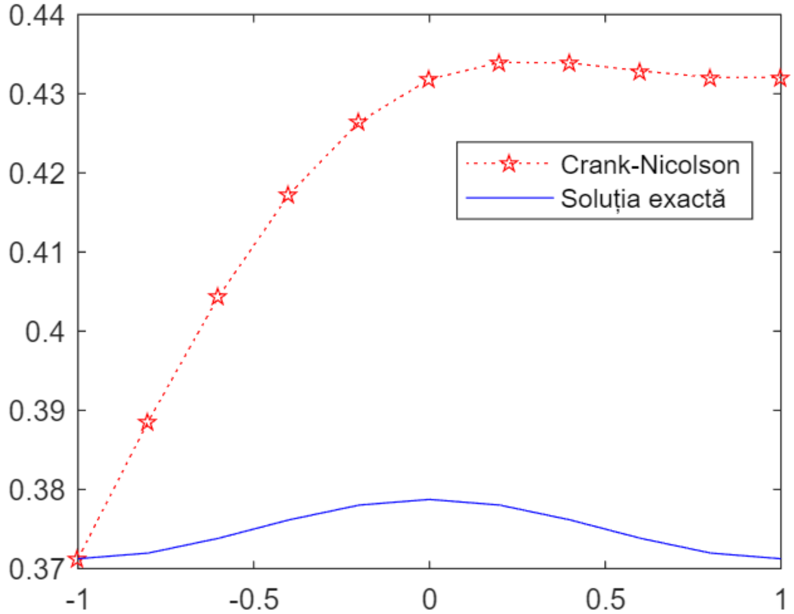
Soluția numerică obținută cu ajutorul metodei Crank-Nicolson și soluția exactă, $\texttt{tf} = 0.125$ **și** $\texttt{dt} = 0.01, \texttt{dx} = 0.2$



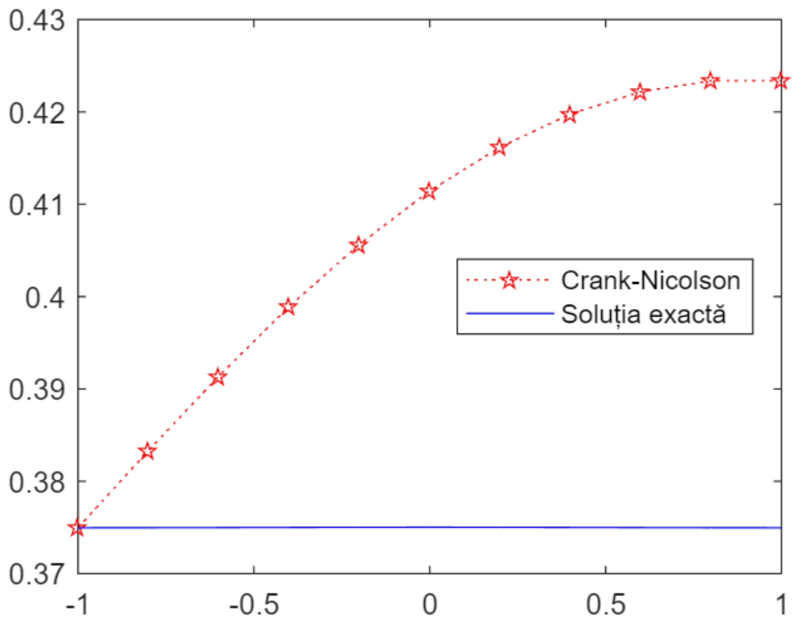
Soluția numerică obținută cu ajutorul metodei Crank-Nicolson și soluția exactă, $\texttt{tf} = 0.25$ **și** $\texttt{dt} = 0.01, \texttt{dx} = 0.2$



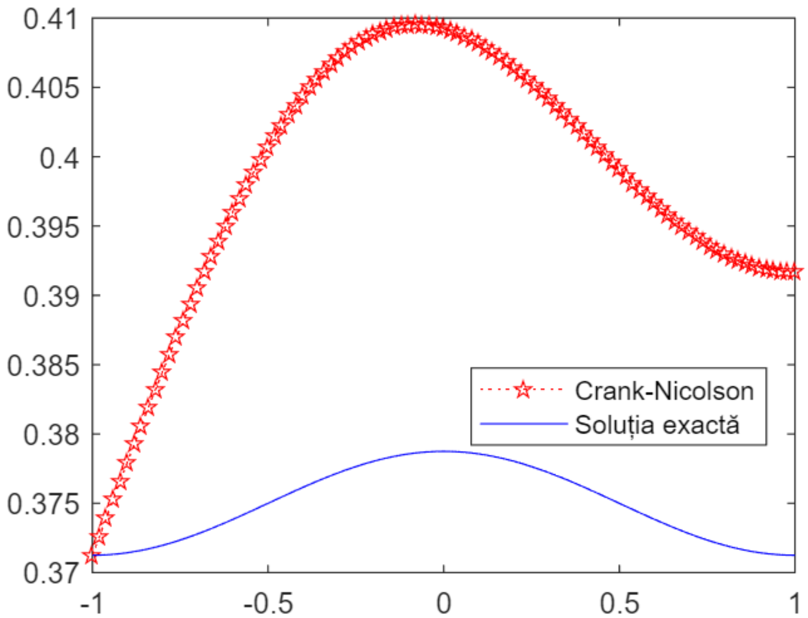
Soluția numerică obținută cu ajutorul metodei Crank-Nicolson și soluția exactă, $\texttt{tf} = 0.375$ **și** $\texttt{dt} = 0.01, \texttt{dx} = 0.2$



Soluția numerică obținută cu ajutorul metodei Crank-Nicolson și soluția exactă, $\mathtt{tf} = 0.5$ **și** $\mathtt{dt} = 0.01, \mathtt{dx} = 0.2$



Soluția numerică obținută cu ajutorul metodei Crank-Nicolson și soluția exactă, $\mathtt{tf} = 1$ **și** $\mathtt{dt} = 0.01, \mathtt{dx} = 0.2$



Soluția numerică obținută cu ajutorul metodei Crank-Nicolson și soluția exactă, $\texttt{tf} = 0.5$ **și** $\texttt{dt} = 0.001, \texttt{dx} = 0.02$

Funcții auxiliare

Funcția `solutiaExacta` calculează simbolic soluția exactă a problemei cu valori pe frontiere. Metoda simbolică a fost aleasă deoarece soluția exactă conține serii care nu pot fi calculate eficient numeric.

```
function rez=solutiaExacta(x,t)
    syms k
    s1=symsum(((-1)^k/(pi*(2*k+1))+2/(pi^2*(2*k+1)^2))...
    *cos(pi*(2*k+1).*x).*exp(-pi^2*(2*k+1)^2.*t),k,0,Inf);
    s2=symsum((cos(2*pi*(2*k+1).*x)/(pi^2*(2*k+1)^2).* ...
    exp(-4*pi^2*(2*k+1)^2.*t)),k,0,Inf);
    v1=double(s1);
    v2=double(s2);
    rez=3/8+v1+v2;
end
```