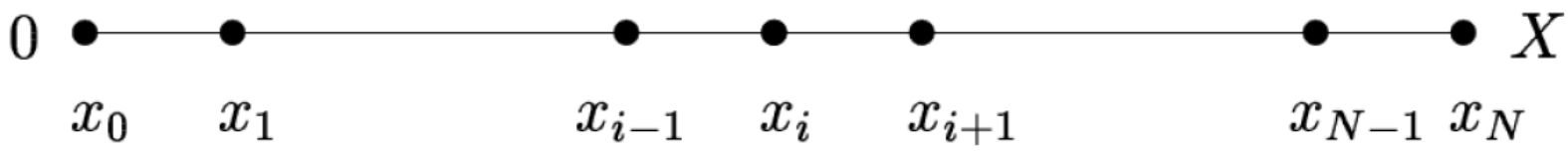
Numeričke metode

Aleksandar Maksimović
IRB

1D finite difference/konačne razlike

1D: $\Omega = (0, X), \qquad u_i \approx u(x_i), \qquad i = 0, 1, \ldots, N$

Diskretizacija: $x_i = i\Delta x$ korak $\Delta x = \frac{X}{N}$

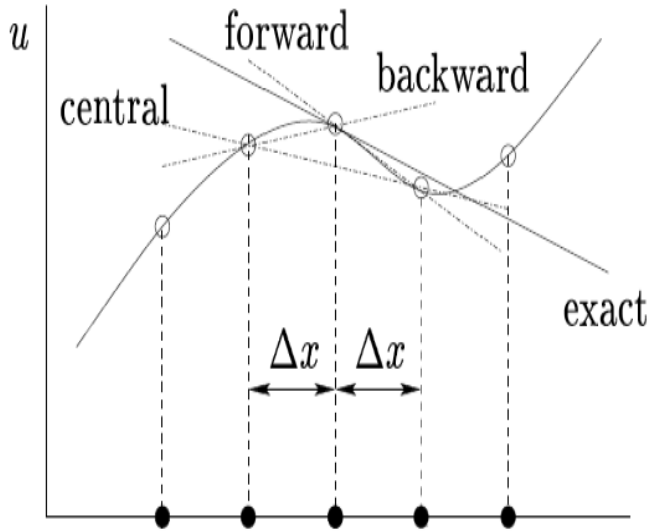


$$\left(\frac{\partial u}{\partial x}\right)_i = \lim_{\Delta x \to 0} \frac{u(x_{i+1}) - u(x_i)}{\Delta x} = \lim_{\Delta x \to 0} \frac{u(x_i) - u(x_{i-1})}{\Delta x}$$

Aleksandar Maksimović,
Institute Ruđer Bošković, Bijenička c. 54

konačne razlike

Geometrijska interpretacija



$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{u_{i+1}-u_i}{\Delta x}$$ naprijed

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{u_i-u_{i-1}}{\Delta x}$$ nazad

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{u_{i+1}-u_{i-1}}{2\Delta x}$$ centralna razlika

konačne razlike

Ekspanzija Taylorovim redom

$$u(x) = \sum_{n=0}^{\infty} \frac{(x-x_i)^n}{n!} \left(\frac{\partial^n u}{\partial x^n}\right)_i , \qquad u \in C^\infty([0,X])$$

$$T_1 : \qquad u_{i+1} = u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{(\Delta x)^2}{2} \left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{(\Delta x)^3}{6} \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

$$T_2 : \qquad u_{i-1} = u_i - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{(\Delta x)^2}{2} \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{(\Delta x)^3}{6} \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

$$T_1 \quad \Rightarrow \quad \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_i}{\Delta x} - \frac{\Delta x}{2} \left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{(\Delta x)^2}{6} \left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

Taylorov red

$$T_2 \quad \Rightarrow \quad \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_i}{\Delta x} + \frac{\Delta x}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{(\Delta x)^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

pogreška: $\mathcal{O}(\Delta x)$

$$T_1 - T_2 \quad \Rightarrow \quad \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} - \frac{(\Delta x)^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \dots$$

pogreška: $\mathcal{O}(\Delta x)^2$

druga derivacija

$$T_1 + T_2 \quad \Rightarrow \quad \left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} + \mathcal{O}(\Delta x)^2$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \left[\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)\right]_i = \lim_{\Delta x \to 0} \frac{\left(\frac{\partial u}{\partial x}\right)_{i+1/2} - \left(\frac{\partial u}{\partial x}\right)_{i-1/2}}{\Delta x}$$

$$\approx \quad \frac{\frac{u_{i+1}-u_i}{\Delta x} - \frac{u_i - u_{i-1}}{\Delta x}}{\Delta x} = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}$$

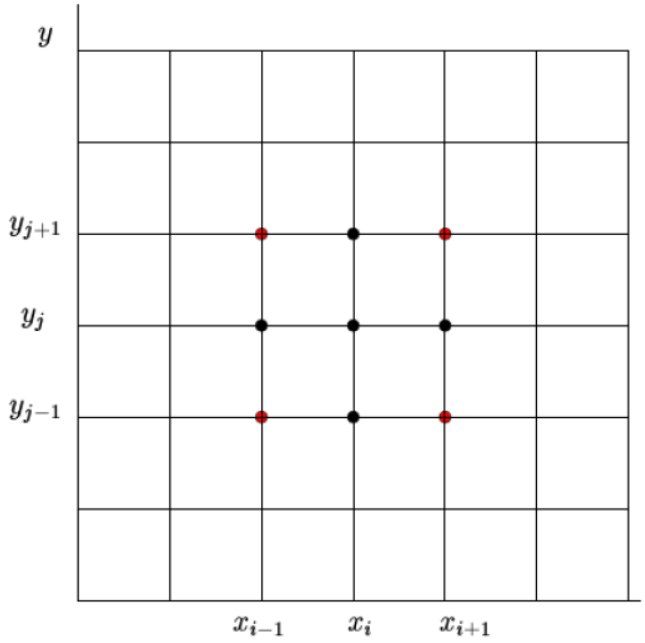
Miješana derivacija, 2D

$$\text{2D:} \quad \frac{\partial^2 u}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}\right) = \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}\right)$$

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{\left(\frac{\partial u}{\partial y}\right)_{i+1,j} - \left(\frac{\partial u}{\partial y}\right)_{i-1,j}}{2\Delta x} + \mathcal{O}(\Delta x)^2$$

$$\left(\frac{\partial u}{\partial y}\right)_{i+1,j} = \frac{u_{i+1,j+1} - u_{i+1,j-1}}{2\Delta y} + \mathcal{O}(\Delta y)^2$$

$$\left(\frac{\partial u}{\partial y}\right)_{i-1,j} = \frac{u_{i-1,j+1} - u_{i-1,j-1}}{2\Delta y} + \mathcal{O}(\Delta y)^2$$



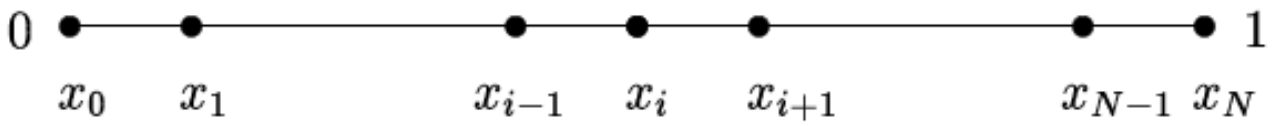
$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y} + \mathcal{O}[(\Delta x)^2, (\Delta y)^2]$$

Aleksandar Maksimović,
Institute Ruđer Bošković, Bijenička c. 54

Primjer: 1D Poisson

Problem s rubnim uvjetima (BVP, boundary value problem, BC boundary condition)

$$-\frac{\partial^2 u}{\partial x^2} = f \quad \text{in } \Omega = (0,1), \qquad u(0) = u(1) = 0$$



$$u_i \approx u(x_i), \quad f_i = f(x_i) \qquad x_i = i\Delta x, \quad \Delta x = \frac{1}{N}, \quad i = 0, 1, \ldots, N$$

$$\begin{cases} -\frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2} = f_i, & \forall i = 1, \ldots, N-1 \\ u_0 = u_N = 0 & \text{Dirichletov rubni uvjet} \end{cases}$$

Neumannov rubni uvjet: postavlja se uvjet na gradijent funkcije

Robinov uvjet: kombinacija uvjeta Dirichleta i Neumanna

1D Poisson

$$
\begin{cases}
i = 1 & -\frac{u_0 - 2u_1 + u_2}{(\Delta x)^2} & = f_1 \\
i = 2 & -\frac{u_1 - 2u_2 + u_3}{(\Delta x)^2} & = f_2 \\
i = 3 & -\frac{u_2 - 2u_3 + u_4}{(\Delta x)^2} & = f_3 \\
 & \cdots & \\
i = N-1 & \frac{u_{N-2} - 2u_{N-1} + u_N}{(\Delta x)^2} & = f_{N-1}
\end{cases}
$$

$$Au = F \qquad A \in \mathbb{R}^{N-1 \times N-1} \qquad u, F \in \mathbb{R}^{N-1}$$

$$
A = \frac{1}{(\Delta x)^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & & \cdots & \\ & & & -1 & 2 \end{bmatrix}, \qquad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ . \\ u_{N-1} \end{bmatrix}, \qquad F = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ . \\ f_{N-1} \end{bmatrix}
$$

1D Poisson, rubni uvjeti

Dirichlet-Neumann BC $\qquad u(0) = \frac{\partial u}{\partial x}(1) = 0$

$$u_0 = 0, \qquad \frac{u_{N+1} - u_{N-1}}{2\Delta x} = 0 \quad \Rightarrow \quad u_{N+1} = u_{N-1}$$

Korekcija za *i*=*N* zbog Neumannovog BC

$$Au = F$$

$$-\frac{u_{N-1} - 2u_N + u_{N+1}}{(\Delta x)^2} = f_N \quad \longrightarrow \quad \frac{-u_{N-1} + u_N}{(\Delta x)^2} = \frac{1}{2} f_N$$

$$A = \frac{1}{(\Delta x)^2} \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & & \dots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}, \qquad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ . \\ u_{N-1} \\ u_N \end{bmatrix}, \qquad F = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ . \\ f_{N-1} \\ \frac{1}{2} f_N \end{bmatrix}$$

drugi tipovi rubnih uvjeta

Nehomogeni Dirichlet BC: $u(0) = g_0$

prva jednadžba $u_0 = g_0 \quad \Rightarrow \quad \frac{2u_1 - u_2}{(\Delta x)^2} = f_1 + \frac{g_0}{(\Delta x)^2}$

Nehomogeni Neumann BC: $\frac{\partial u}{\partial x}(1) = g_1$

$$\frac{u_{N+1} - u_{N-1}}{2\Delta x} = g_1 \quad \Rightarrow \quad u_{N+1} = u_{N-1} + 2\Delta x g_1$$

$$-\frac{u_{N-1} - 2u_N + u_{N+1}}{(\Delta x)^2} = f_N \quad \longrightarrow \quad \frac{-u_{N-1} + u_N}{(\Delta x)^2} = \frac{1}{2} f_N + \frac{g_1}{\Delta x}$$

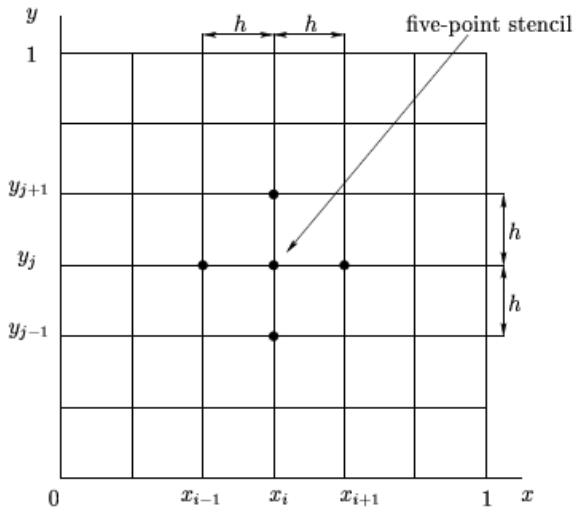
Nehomogeni Robin BC: $\frac{\partial u}{\partial x}(1) + \alpha u(1) = g_2$

$$\frac{u_{N+1} - u_{N-1}}{2\Delta x} + \alpha u_N = g_2 \quad \Rightarrow \quad u_{N+1} = u_{N-1} - 2\Delta x \alpha u_N + 2\Delta x g_2$$

$$-\frac{u_{N-1} - 2u_N + u_{N+1}}{(\Delta x)^2} = f_N \quad \longrightarrow \quad \frac{-u_{N-1} + (1 + \alpha\Delta x) u_N}{(\Delta x)^2} = \frac{1}{2} f_N + \frac{g_2}{\Delta x}$$

2D Poisson

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f & \text{in } \Omega = (0,1) \times (0,1) \\ u = 0 & \text{on } \Gamma = \partial\Omega \end{cases}$$



Uniformna mreža, konstantan korak, N broj koraka

$$\Delta x = \Delta y = h, \quad N = \frac{1}{h}$$

$$u_{i,j} \approx u(x_i, y_j), \quad f_{i,j} = f(x_i, y_j), \quad (x_i, y_j) = (ih, jh), \quad i,j = 0, 1, \ldots, N$$

$$\begin{cases} -\frac{u_{i-1,j}+u_{i,j-1}-4u_{i,j}+u_{i+1,j}+u_{i,j+1}}{h^2} = f_{i,j}, & \forall i,j = 1, \ldots, N-1 \\ u_{i,0} = u_{i,N} = u_{0,j} = u_{N,j} = 0 & \forall i,j = 0, 1, \ldots, N \end{cases}$$

2D Poisson

$$Au = F \qquad A \in \mathbb{R}^{(N-1)^2 \times (N-1)^2} \qquad u, F \in \mathbb{R}^{(N-1)^2}$$

$$u = [u_{1,1} \dots u_{N-1,1}\ u_{1,2} \dots u_{N-1,2}\ u_{1,3} \dots u_{N-1,N-1}]^T$$

$$F = [f_{1,1} \dots f_{N-1,1}\ f_{1,2} \dots f_{N-1,2}\ f_{1,3} \dots f_{N-1,N-1}]^T$$

$$A = \begin{bmatrix} B & -I & & & \\ -I & B & -I & & \\ & \dots & \dots & & \\ & & -I & B & -I \\ & & & -I & B \end{bmatrix}, \qquad B = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \dots & \dots & & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}$$

matrica A je "šupljikava" (sparse).

povećanjem broja točaka u mreži efikasnost riješavanja opada.

1D jednadžba vođenja topline

$$\frac{\partial u}{\partial t} = \beta \frac{\partial^2 u}{\partial x^2}, \quad a \le x \le b$$

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = \beta \frac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{(\Delta x)^2}$$

$$u(a,t) = g_0(t), \quad u(b,t) = g_1(t)$$

$$u_i^{k+1} = u_i^k + r(u_{i-1}^k - 2u_i^k + u_{i+1}^k)$$

$$r = \beta \Delta t / (\Delta x)^2.$$

$$u(x,0) = g(x),$$

Rubni uvjeti

$$u_0^k = g_0(t_k) = g_0^k, \qquad u_{N+1}^k = g_1(t_k) = g_1^k$$

$$i = 1, \quad u_1^{1+k} = u_1^k + r\,(u_0^k - 2\,u_1^k + u_2^k)$$

$$i, \quad u_i^{1+k} = u_i^k + r\,(u_{-1+i}^k - 2\,u_i^k + u_{1+i}^k)$$

$$i = N, u_N^{1+k} = u_N{}^k + r\,(u_{-1+N}^k - 2\,u_N{}^k + u_{1+N}^k)$$

Sustav jednadžbi

eksplicitna metoda

$$\mathbf{U}^{k+1} = A\mathbf{U}^k + rg(t)$$

$$A = \begin{pmatrix} (1-2r) & r & & & & \\ r & (1-2r) & r & & & \\ & r & (1-2r) & r & & \\ & & \ddots & \ddots & \ddots & \\ & & & r & (1-2r) & r \\ & & & & r & (1-2r) \end{pmatrix}$$

$$\Delta t \leq \frac{1}{2}h^2.$$

ili $r \leq 1/2$ uvjet stabilnosti

$$\mathbf{U}^k = \begin{pmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_i^k \\ \vdots \\ u_N^k \end{pmatrix} \qquad g(t) = \begin{pmatrix} g_0^k \\ 0 \\ 0 \\ \vdots \\ 0 \\ g_1^k \end{pmatrix}$$

$h = 0.1, \Delta t = 0.5 \times 10^{-2}$

$h = 0.001, \Delta t = 0.5 \times 10^{-6}.$

Implicitna metoda

Za derivaciju po vremenu koristimo Eulerov metod unazad, a prostor centralnom konačnom razlikom. Metoda je implicitna u vremenu, rješenje u *k+1* vremenu je međusobno povezano. Algoritam je uvijek stabilan.

$$\frac{u_i^k - u_i^{k-1}}{\Delta t} = \beta \frac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{(\Delta x)^2}$$

$$i = 1, \quad u_1^{k+1} - r\left(u_0^{k+1} - 2\,u_1^{k+1} + u_2^{k+1}\right) = u_1^k$$

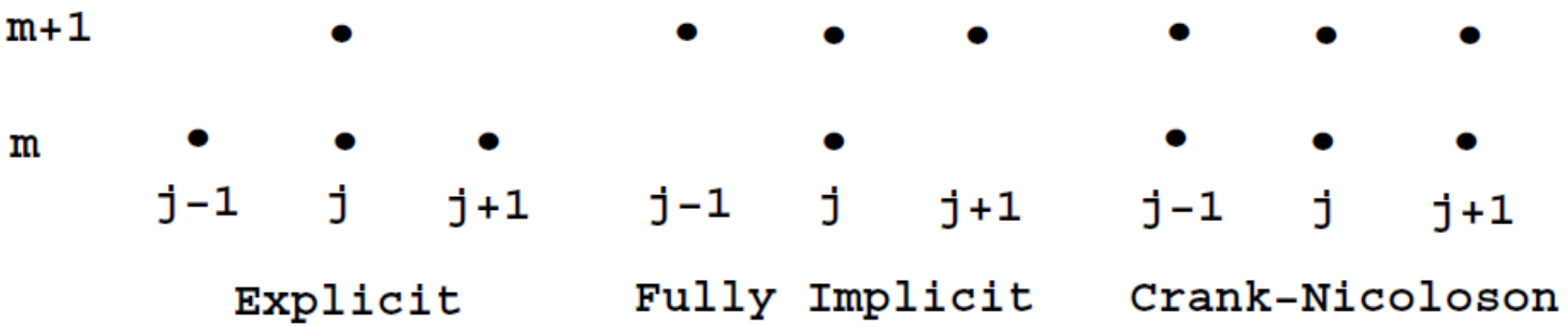
$$i, \quad u_i^{k+1} - r\left(u_{i-1}^{k+1} - 2\,u_i^{k+1} + u_{i+1}^{k+1}\right) = u_i^k$$

$$i = N, u_N^{k+1} - r\left(u_{N-1}^{k+1} - 2\,{u_N}^{k+1} + u_{N+1}^{k+1}\right) = {u_N}^k$$

$$\begin{pmatrix} (1+2r) & -r & & & & & \\ -r & (1+2r) & -r & & & & \\ & -r & (1+2r) & -r & & & \\ & & & \ddots & \ddots & & \\ & & & -r & (1+2r) & -r & \\ & & & & -r & (1+2r) \end{pmatrix} \begin{pmatrix} u_1^{k+1} \\ u_2^{k+1} \\ \vdots \\ u_i^{k+1} \\ \vdots \\ u_N^{k+1} \end{pmatrix} =$$

$$= \begin{pmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_i^k \\ \vdots \\ u_N^k \end{pmatrix} + r \begin{pmatrix} g_0^{k+1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ g_1^{k+1} \end{pmatrix}$$

Crank-Nicolson



Crank-Nicolson metoda predstavlja srednju vrijednost implicitne i eksplicitne metode u prostoru, vremenski dio je implicitan.

$$u_j^{m+1} - u_j^m = \frac{\Delta t}{2h^2}(u_{j+1}^{m+1} - 2u_j^{m+1} + u_{j-1}^{m+1} + u_{j+1}^m - 2u_j^m + u_{j-1}^m)$$

MOL

MOL (method of lines), metoda linija sastoji se od aproksimacije konačnim razlikama u prostoru, dobivamo sustav običnih diferencijalnih jednadžbi.

$$\frac{\partial u_i}{\partial t} = \beta \frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2}$$

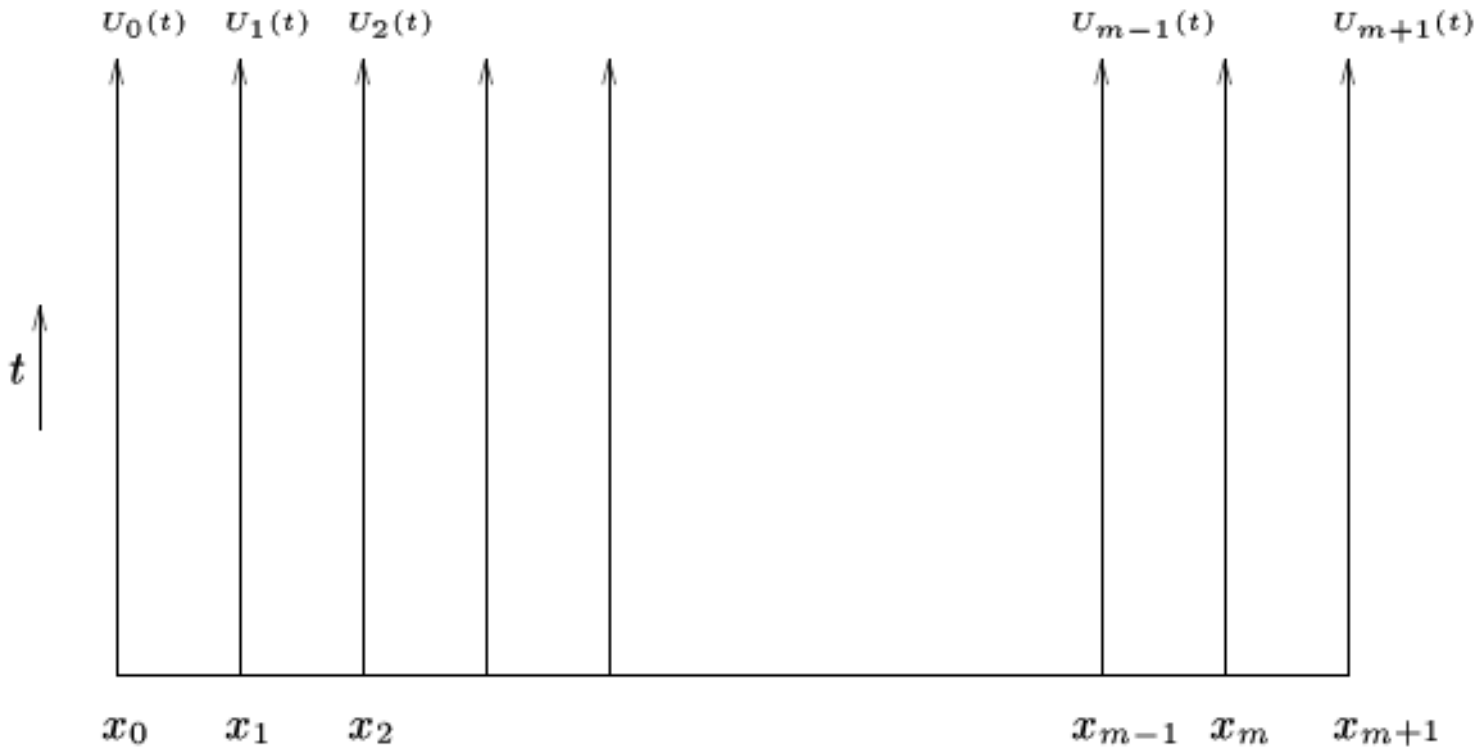
$$u_i' = r(u_{i-1} - 2u_i + u_{i+1}),$$

$$\mathbf{U}'(t) = A\mathbf{U}(\mathbf{t}) + g(t)$$

$$\begin{aligned}
i = 0 \quad & u_0' = r(u_{-1} - 2u_0 + u_1) \\
i = 1 \quad & u_1' = r(u_0 - 2u_1 + u_2) \\
i \quad & u_i' = r(u_{i-1} - 2u_i + u_{i+1}) \\
i = N \quad & u_N' = r(u_{N-1} - 2u_N + u_{N+1})
\end{aligned}$$

$$A = r \begin{pmatrix}
-2 & 1 & & & & \\
1 & -2 & 1 & & & \\
 & 1 & -2 & 1 & & \\
 & & & \ddots & \ddots & \\
 & & & 1 & -2 & 1 \\
 & & & & 1 & -2
\end{pmatrix}$$

MOL



Riješenje napreduje duž linije vremena. Riješenje sustava dobivamo korištenjem metoda za ODJ.

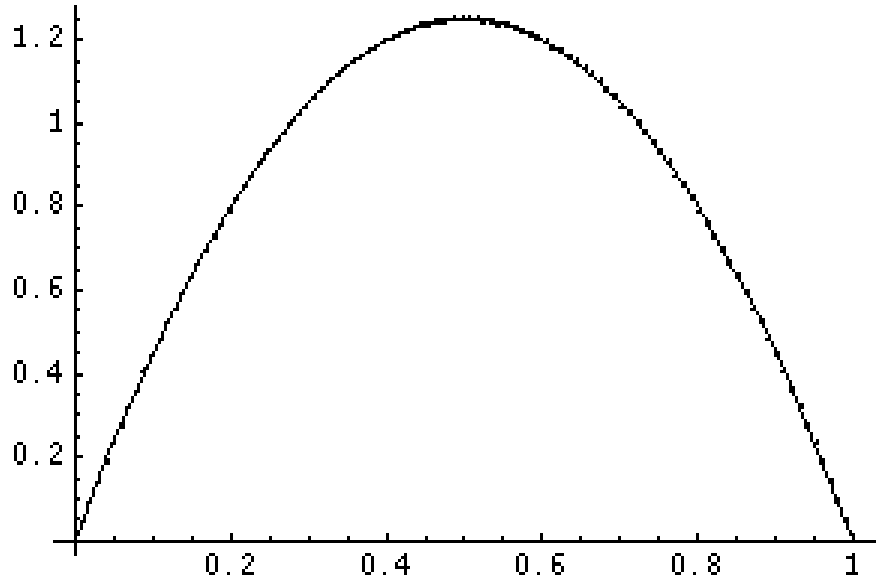
Primjer 1D poisson

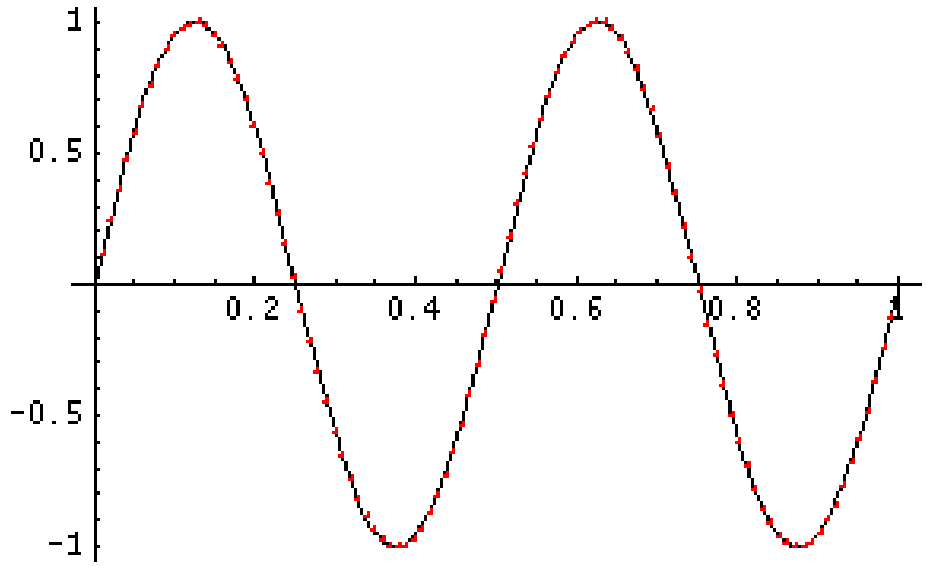
$$-\frac{\partial^2 u}{\partial x^2} = f \quad \text{in } \Omega = (0,1), \qquad u(0) = u(1) = 0$$

f(x)=10

$$f(x) = 16\pi^2 \sin 4\pi x$$

```
nd1 = NDSolve[{u''[x] == -10, u[0] == 0, u[1] == 0}, u[x], {x, 0, 1}]
tn1 = Plot[u[x] /. nd1, {x, 0, 1}, PlotRange -> All]
```





Aleksandar Maksimović,
Institute Ruđer Bošković, Bijenička c. 54

poisson

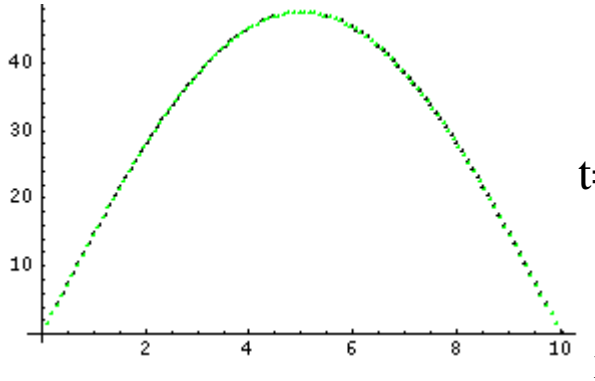
```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "nrutil.h"
float InitTemp(float );
float KonstTemp(float );

int  main()
{
  unsigned long N,M,i,j,l;
  float h,k,r,T,L;
  float T0,TN,Tini;
  /* matrica ya temperaturu */
  float **U,*f,*x,d;
   FILE  *fp;
   char fname[20];
   int *indx;
   float (*InitFunc)(float);
  /* parametri */

  L=1.0; /* duljina domene u prostoru */

  /*   InitFunc=InitTemp;  */
   InitFunc=KonstTemp;
  M=3000;
  N=100;
  h=L/((float) N+1.0);
  k=T/((float) M);
  r=k/(h*h);
```

```
 U=matrix(1,N,1,N);
  f=vector(1,N);
  indx=ivector(1,N);
  x=vector(1,N);
  for(i=1;i<N+1;i++)
    for(j=1;j<N+1;j++)
      U[i][j]=0.0;

  U[1][1]=2.0;
  U[1][2]=-1;

  for (i=2;i<=N-1;i++)
    {
      f[i]=h*h*InitFunc(i*h);
      for (j=i-1 ; j<i+2; j++)
    {
      if ( i==j )
        U[i][j]= 2;
      else
        U[i][j]=-1;
    }
    }
  U[N][N]=2.0; U[N][N-1]=-1;
   ludcmp(U,N,indx,&d);  lubksb(U,N,indx,f);

PRINT TO FILE .....................
.....................
return 1;
}
```

```
float InitTemp(float x)
{
  float rez,pi=3.14;
  rez=16.0*pi*pi*sin(4*pi*x);
  return rez;
}


float KonstTemp(float x)
{
  return 10.0;
    }
```

vođenje topline

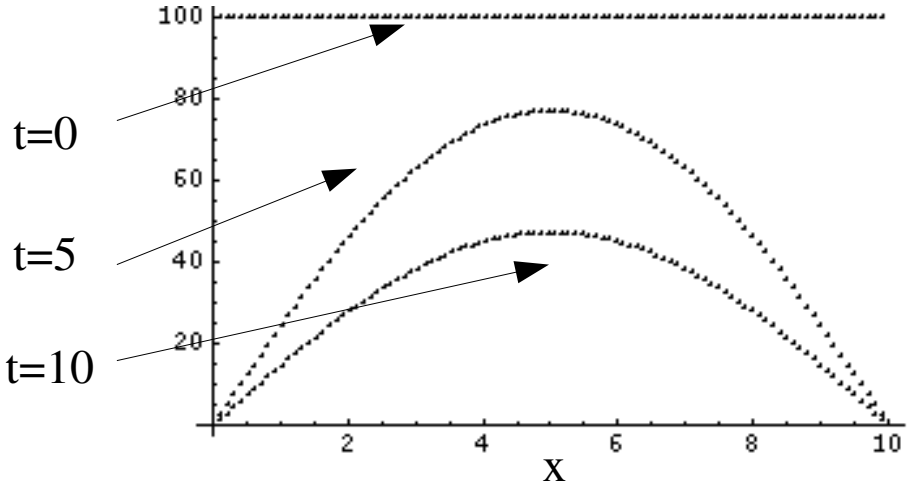
$$\frac{\partial u}{\partial t} = \beta \frac{\partial^2 u}{\partial x^2}, \quad a \le x \le b$$



t=5

Usporedba eksplicitne i MOL metode

eksplicitna metoda

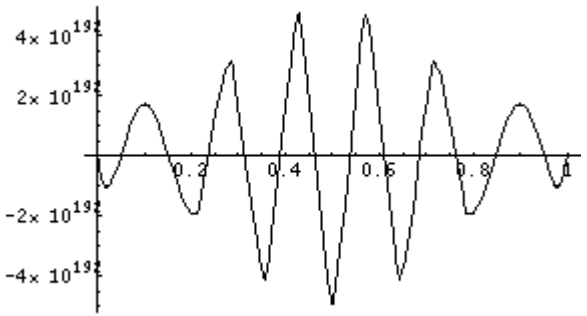


t=0

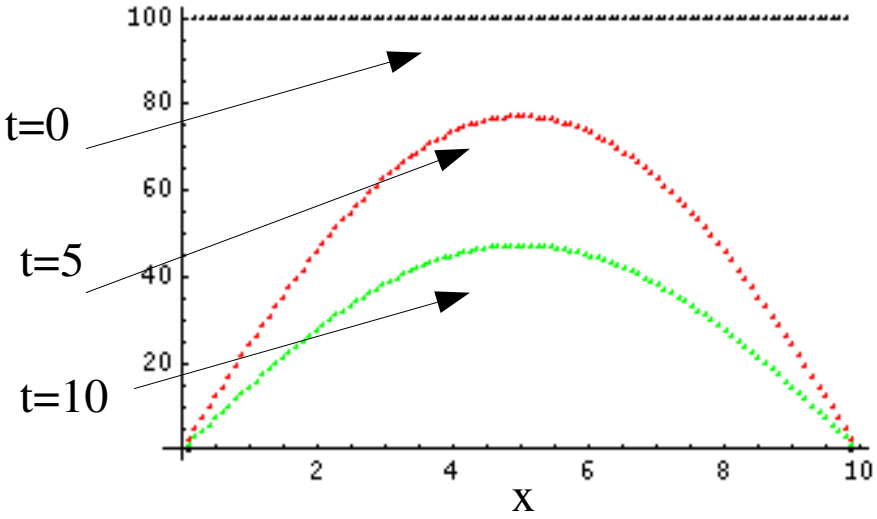
t=5

t=10

NDSolve[{D[u[x, t],t] + D[u[x, t],{x,2}] == 0, u[0, t] == 0, u[1, t] == 0, u[x, 0] == 100}, u[x, t], {x, 0, 1}, {t, 0, 1}]



MOL metoda



t=0

t=5

t=10

Temperatura ekspilicitni FD

```
#include <stdlib.h>
#include <stdio.h>
#include "nrutil.h"

void main()
{
  unsigned long N,M,i,j,l;
  float h,k,r,T,L;
  float T0,TN,Tini;
  /* matrica ya temperaturu */
   float** U;
   FILE  *fp;
   char fname[20];
  /* parametri */
  T=10; /* 10 sec konacno vrijeme */
  L=10.0; /* duljina domene u prostoru */
  /*pocetne i rubne temperature */
  T0=TN=0; Tini=100;
  M=3000;
  N=100;
  h=L/((float) N-1.0);
  k=T/((float) M);
  r=k/(h*h);

  if ( r > .5)
    {
      printf("nestabilna shema, podesi N i M, r=%f\n",r);
      exit(1);
    }
  U=matrix(0,M,0,N);
  for (i=0;i<N;i++) {U[0][i]=Tini;}
  for(j=0;j<M;j++) { U[j][0]= T0; U[j][N-1]=TN;}

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

    }
  /* otvori file i spremi odabrane podatke */
  for (l=0;l<3;l++)
    {

      sprintf(fname,"Temp%i.dat",l);
      if (!(fp=fopen(fname,"w")))
    { printf("Error opening file\n"); exit(1);}
      if (l==0)
    j= 0;
      else if (l== 1)
    j=M/2;
      else if (l==2)
    j= M-1;
        for (i=0 ; i<N ;i++)
      {
    fprintf(fp,"%10.5f  %10.5f\n",i*h,U[j][i]);
      }

    fclose(fp);
    }

  exit(1);

}
```

Temperatura MOL

```c
#include <stdio.h>
#include <stdlib.h>
#define NRANSI
#include "nr.h"
#include "nrutil.h"

#define NVAR 100
#define NSTEP 3000

float r;
extern float **y,*xx;
/* referencing declaration */

void main()
{
  unsigned long N,M,i,j,l;
  float h,T,L;
  float T0,TN,Tini;
  /* matrica ya temperaturu */
  float *v;
   FILE  *fp;
   char fname[20];

  T=10; /* 10 sec konacno vrijeme */
  L=10.0; /* duljina domene u prostoru */
  /*pocetne i rubne temperature */
  T0=TN=0; Tini=100;
  M=NSTEP;
  N=NVAR;
  h=L/((float) N+1.0);
  r=1.0/(h*h);
```

```c
v=vector(1,NVAR);
  xx=vector(1,NSTEP+1);
  y=matrix(1,NVAR,1,NSTEP+1);
  for (i=1;i<=N;i++)
    v[i]=Tini;
  v[1]=v[N]=0;
   rkdumb(v,NVAR,0.0,T,NSTEP,derivs);

  /* otvori file i spremi odabrane podatke */
  for (l=0;l<3;l++)
    {

      sprintf(fname,"TempMOL%i.dat",l);
      if (!(fp=fopen(fname,"w")))
    { printf("Error opening file\n"); exit(1);}
      if (l==0)
    j= 1;
      else if (l== 1)
    j=M/2;
      else if (l==2)
    j= M-1;
      printf("Data for t=%g\n",xx[j]);
        for (i=1 ; i<=N ;i++)
      {
    fprintf(fp,"%10.5f  %10.5f\n",i*h,y[i][j]);
      }

    fclose(fp);
    }

  exit(1);}
```

```c
void derivs(float x, float y[], float dy[]) {
  unsigned long i,N;
  N=NVAR;
  dy[1] =r*(-2*y[1]+y[2]);
  for ( i=2; i<=N-1;i++)
    {
      dy[i] = r*(y[i-1]-2*y[i]+y[i+1]);
    }
  dy[N] =r*(y[N-1]-2*y[N]);}
```

Kompajliranje

```
gcc  -o poisson1 poisson_exfd.c ludcmp.c lubksb.c -I nrutils/ nrutils/nrutil.c  -lm
gcc  -o heatMOL heat_MOL.c -I nrutils/ nrutils/nrutil.c  rk4.c  rkdumb.c
gcc -o heat_ex heat_explicitfd.c  -I nrutils/ nrutils/nrutil.c

f77  -o hmol heatMOL.for rk4.for
```

Usporedbe riješenja i grafički prikaz nalaze se u TempDist.nb notebooku.
Programi se nalaze u v9 direktoriju.

Zadatak za praktikum

Riješi 1D poissonov problem pomoću RK metode za f(x)=10 i $f(x) = 16\pi^2 \sin 4\pi x$

Problem je što ne možemo specificirati rubni uvjet za x=1 kod RK metode, već moramo znati derivaciju u x=0. Problemi s rubnim uvjetima riješavaju se pomoću relaksacijske ili shooting metode (vidi Numerical recipes poglavlje 17).

Budući da je problem jednostavan iz analitičkog riješenja možemo odrediti vrijednost derivacije na lijevom kraju, tj. u'(0)=5 i u'(0)= 4 π.

Usporedi riješenje s onim dobivenim pomoću metode konačnih razlika.

Literatura

- Finite Difference Methods for Differential Equations, R. J. LaVeque
- Advanced Methods in Scientific Computing, Cristopher R. Johnson

- George Em Karniadakis and Robert M. Kirby II: Parallel Scientific Computing in C++ and MPI, Cambridge University Press.