Addis Ababa University

Computational Data Science Program

Course Title: Numerical Solution of DEs (CDSC 606)

Chapter 2 Finite Difference Methods

Course Instructor : Hailemichael Kebede (PhD)
hailemichael.kebede@aau.edu.et
0911901997

2024

Chapter 2:

2.1 Finite difference Method

2.2 Method of lines

2.3 Numerical solutions to PDEs

2.4 Methods to solve linear system ***(Direct and iterative methods)***

2.1 Finite Difference Method

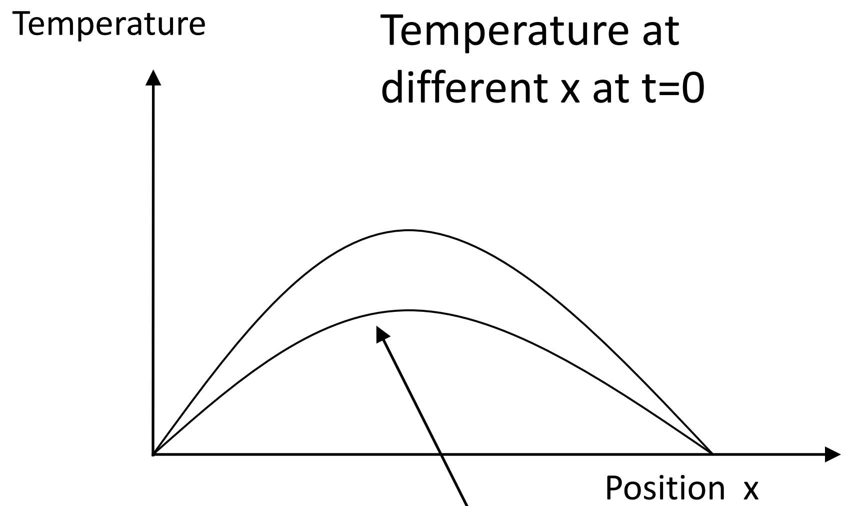
Recall, PDE

- ***PDEs are used to model many systems in many different fields of science and engineering.***

Heat Equation

Different curve is used for each value of t

Thin metal rod insulated everywhere except at the edges.
At t =0 the rod is placed in ice



Temperature at different x at t=0

Temperature at different x at t=h

$$\frac{\partial^2\ T(x,t)}{\partial x^2} - \frac{\partial\ T(x,t)}{\partial t} = 0$$

$$T(0,t) = T(1,t) = 0$$

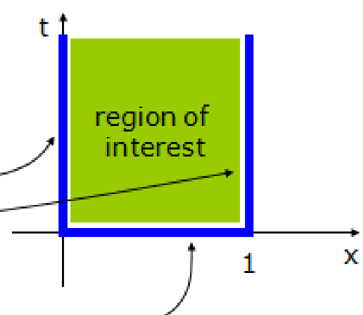
$$T(x,0) = \sin(\pi\ x)$$

Heat Equation : $\alpha \frac{\partial^2 u(x,t)}{\partial x^2} - \frac{\partial u(x,t)}{\partial t} = 0$

$u(0,t) = 0$

$u(1,t) = 0$

$u(x,0) = \sin(\pi\ x)$



$$\frac{\partial u(x,y,z,t)}{\partial t} = \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

BVP: Laplace Equation

$$\frac{\partial^2 u(x,y,z)}{\partial x^2} + \frac{\partial^2 u(x,y,z)}{\partial y^2} + \frac{\partial^2 u(x,y,z)}{\partial z^2} = 0$$

Used to describe the steady state distribution of heat in a body.

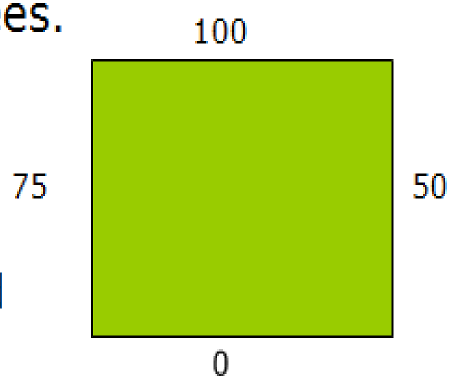
Also used to describe the steady state distribution of electrical charge in a body.

$$\frac{\partial^2 \; T(x,y)}{\partial x^2} + \frac{\partial^2 \; T(x,y)}{\partial y^2} = f(x,y)$$

T : steady state temperature at point (x, y)

$f(x,y)$: heat source (or heat sink)

It is required to determine the steady state temperature at all points of a heated sheet of metal. The edges of the sheet are kept at a constant temperature: 100, 50, 0, and 75 degrees.



The sheet is divided to 5X5 grids.

Another example of a second order linear equation is the following.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0,$$

or more generally

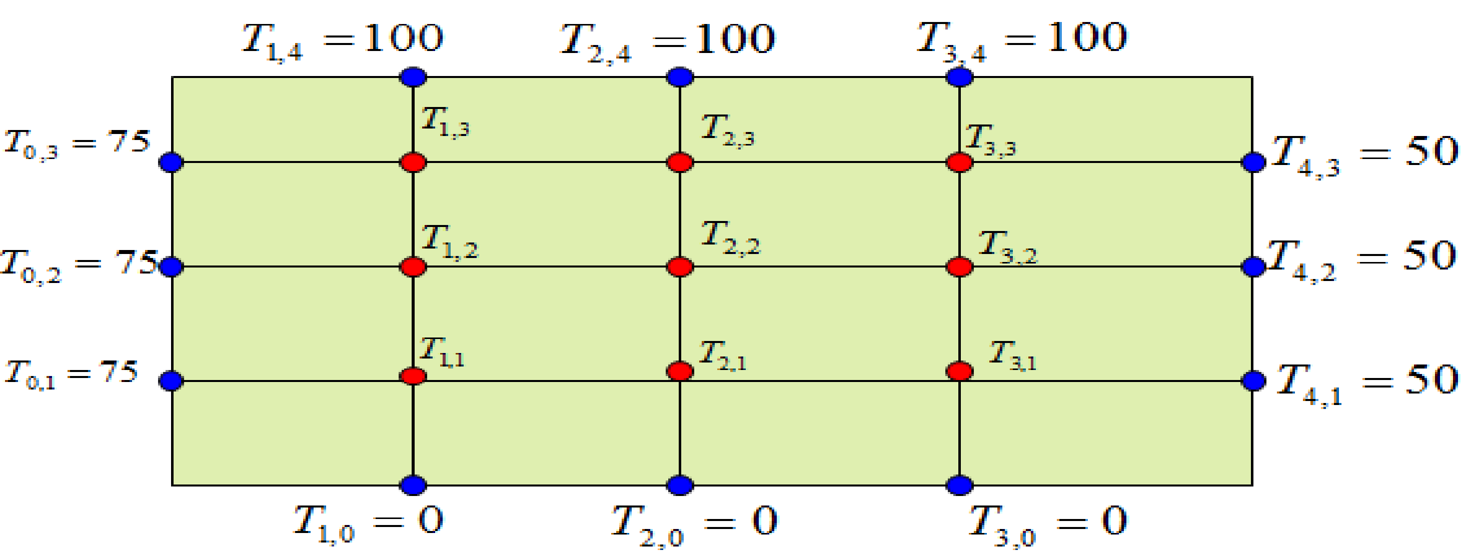
$$\nabla^2 u = 0.$$

This equation usually describes steady processes and is solved subject to some boundary conditions.

Known

To be determined

Example



Wave Equation

$$\frac{\partial^2 u(x,y,z,t)}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

The function $u(x,y,z,t)$ is used to represent the displacement at time t of a particle whose position at rest is (x,y,z) .

The constant c represents the propagation speed of the wave.

Waves on a string, sound waves, waves on stretch membranes, electromagnetic waves, etc.

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2},$$

or more generally

$$\frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} = \nabla^2 u$$

where c is a constant (wave speed).

Finite Difference Operators

There are three difference operators namely forward, backward and central difference operators.

Forward Difference Operator

Consider the function y = f(x). Suppose we are given a table of values of the function at the points

$$x_0,\, x_1 = x_0 + h,\, x_2 = x_0 + 2h,\, \cdots, x_n = x_0 + nh\,.$$

Let $f(x_0) = y_0,\, f(x_1) = y_1,\, f(x_n) = y_n$.

We define

$$\Delta[f(x)] = f(x+h) - f(x)$$

Thus $\Delta y_0 = f(x_0 + h) - f(x) = f(x_1) - f(x_0) = y_1 - y_0$.

$\therefore \Delta y_0 = y_1 - y_0$.

Finite Difference Methods . . .

Backward Differences

Consider the function y = f(x). Suppose we are given a table of values of the function at the points

$$x_0,\ x_1 = x_0 + h,\ x_2 = x_0 + 2h,\ \cdots,\ x_n = x_0 + nh\,.$$

Let $f(x_0) = y_0,\ f(x_1) = y_1,\ f(x_2) = y_2,\ \cdots,\ f(x_n) = y_n\,.$

We define

$$\nabla[f(x)] = f(x) - f(x-h)$$

Thus $\nabla y_1 = y_1 - y_0$

$$\nabla y_2 = y_2 - y_1$$

$$\vdots \quad \vdots \quad \vdots$$

$$\nabla y_n = y_n - y_{n-1}\,.$$

∇ is called the backward difference operator and $\nabla y_1, \nabla y_2, \cdots, \nabla y_n$ are called the first order backward differences of the function y = f(x).

Central Difference Operator can be derived from Taylor series expansion

$$f(x+\Delta x)-f(x-\Delta x)=f(x)+\Delta x\frac{df}{dx}\bigg|_x-f(x)+\Delta x\frac{df}{dx}\bigg|_x$$

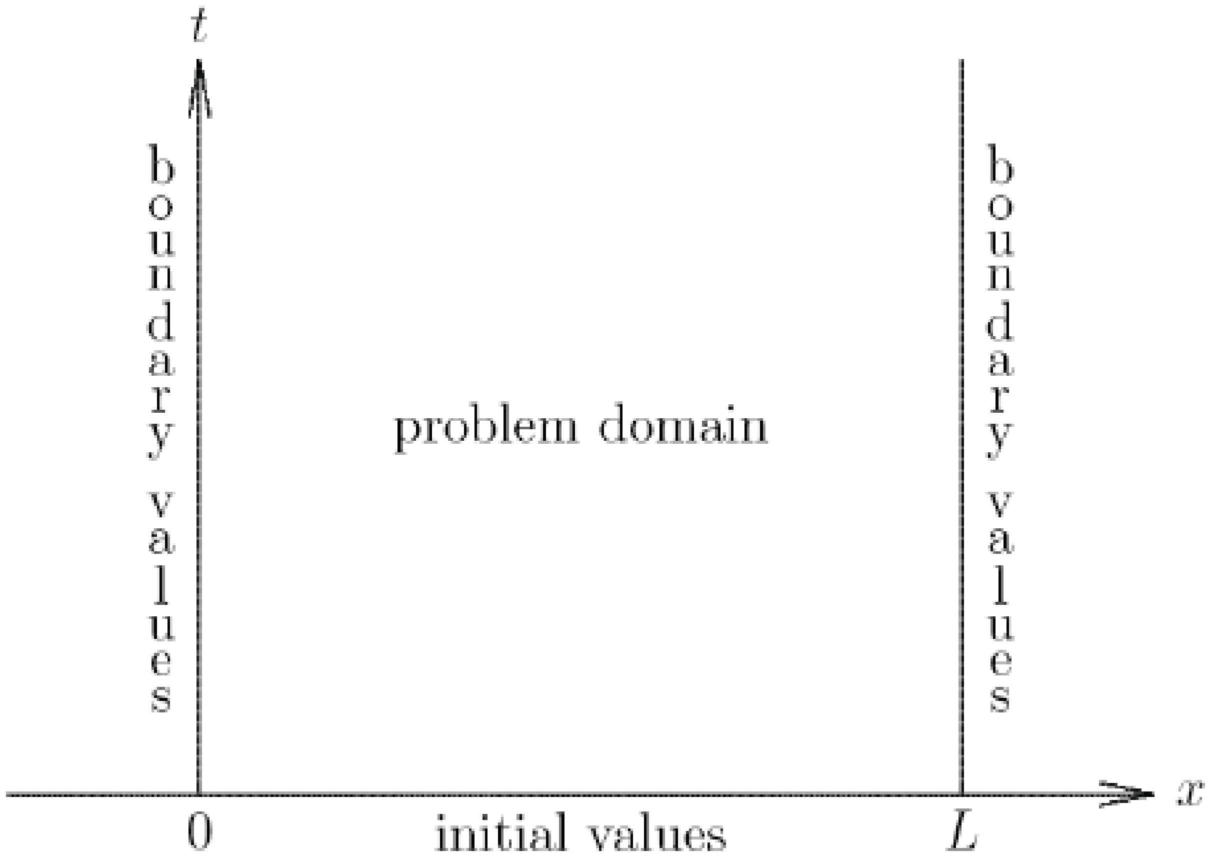
Rearranging eqn we obtain the first central difference derivative at x as:

$$\frac{df}{dx}\bigg|_x=\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}$$

2.4 Numerical solutions to PDEs

Consider the Time-dependent problems

- Time-dependent PDEs usually involve both initial values and boundary values



Semi-discrete Finite difference: Method of Lines

- One way to solve time-dependent PDE numerically is to discretize in space but leave time variable continuous
- Result is system of ODEs that can then be solved by methods previously discussed
- For example, consider heat equation

$$u_t = c\, u_{xx}, \qquad 0 \le x \le 1, \qquad t \ge 0$$

with initial condition

$$u(0, x) = f(x), \qquad 0 \le x \le 1$$

and boundary conditions

$$u(t, 0) = 0, \qquad u(t, 1) = 0, \qquad t \ge 0$$

Semi-discrete Finite difference: Method of Lines

- Define spatial mesh points $x_i = i\Delta x$, $i = 0, \ldots, n+1$, where $\Delta x = 1/(n+1)$
- Replace derivative u_{xx} by finite difference approximation

$$u_{xx}(t, x_i) \approx \frac{u(t, x_{i+1}) - 2u(t, x_i) + u(t, x_{i-1})}{(\Delta x)^2}$$

- Result is system of ODEs

$$y_i'(t) = \frac{c}{(\Delta x)^2}\left(y_{i+1}(t) - 2y_i(t) + y_{i-1}(t)\right), \quad i = 1, \ldots, n$$

  where $y_i(t) \approx u(t, x_i)$
- From boundary conditions, $y_0(t)$ and $y_{n+1}(t)$ are identically zero, and from initial conditions, $y_i(0) = f(x_i)$, $i = 1, \ldots, n$
- We can therefore use ODE method to solve IVP for this system — this approach is called *Method of Lines*

Wave Equation : Solution using Finite Difference

- Consider wave equation

$$u_{tt} = c\,u_{xx}, \qquad 0 \le x \le 1, \qquad t \ge 0$$

with initial and boundary conditions

$$u(0,x) = f(x), \qquad u_t(0,x) = g(x)$$

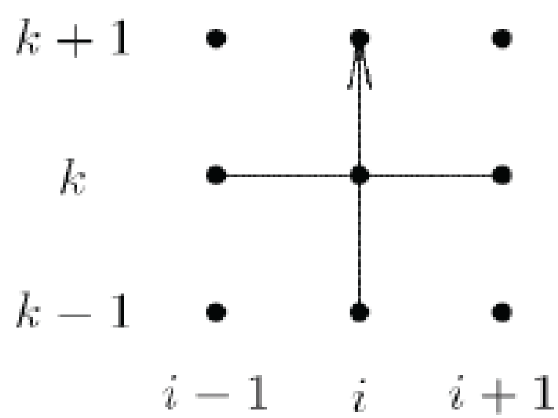
$$u(t,0) = \alpha, \qquad u(t,1) = \beta$$

wave equation : solution

- With mesh points defined as before, using centered difference formulas for both u_{tt} and u_{xx} gives finite difference scheme

$$\frac{u_i^{k+1} - 2u_i^k + u_i^{k-1}}{(\Delta t)^2} = c\,\frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{(\Delta x)^2}, \quad \text{or}$$

$$u_i^{k+1} = 2u_i^k - u_i^{k-1} + c\left(\frac{\Delta t}{\Delta x}\right)^2 \left(u_{i+1}^k - 2u_i^k + u_{i-1}^k\right), \; i = 1, \ldots, n$$



Wave Equation: solution

- Using data at two levels in time requires additional storage
- We also need u_i^0 and u_i^1 to get started, which can be obtained from initial conditions

$$u_i^0 = f(x_i), \qquad u_i^1 = f(x_i) + (\Delta t) g(x_i)$$

where latter uses forward difference approximation to initial condition $u_t(0, x) = g(x)$

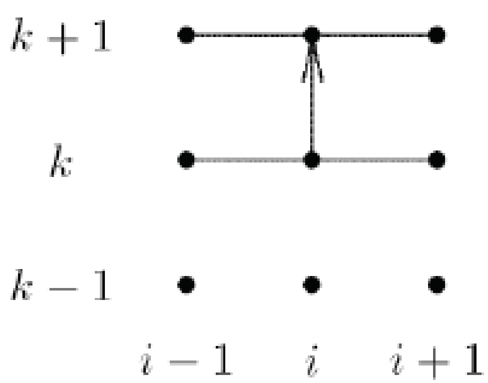
Solution : Implicit finite difference

- This scheme inherits unconditional stability of backward Euler method, which means there is no stability restriction on relative sizes of Δt and Δx

- But first-order accuracy in time still severely limits time step

- Applying *trapezoid method* to semidiscrete system of ODEs for heat equation yields implicit *Crank-Nicolson* method

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



Method is unconditionally stable and second-order accurate in time

❖Finite Difference Formulation

- ✓ **It is a point wise approximation to differential equation**

The discretized form of equation (*) can be written using different approach

- ✓**FTCS(Forward time central space scheme)**
- ✓**FTFS (Forward time forward space)**
- ✓**FTBS (Forward time Backward space)**
- ✓**Upwind scheme**
- ✓**Leapfrog scheme**
- ✓**Crank Nicholson**
- ✓**ADI scheme**

Examples of models: Thermodynamics

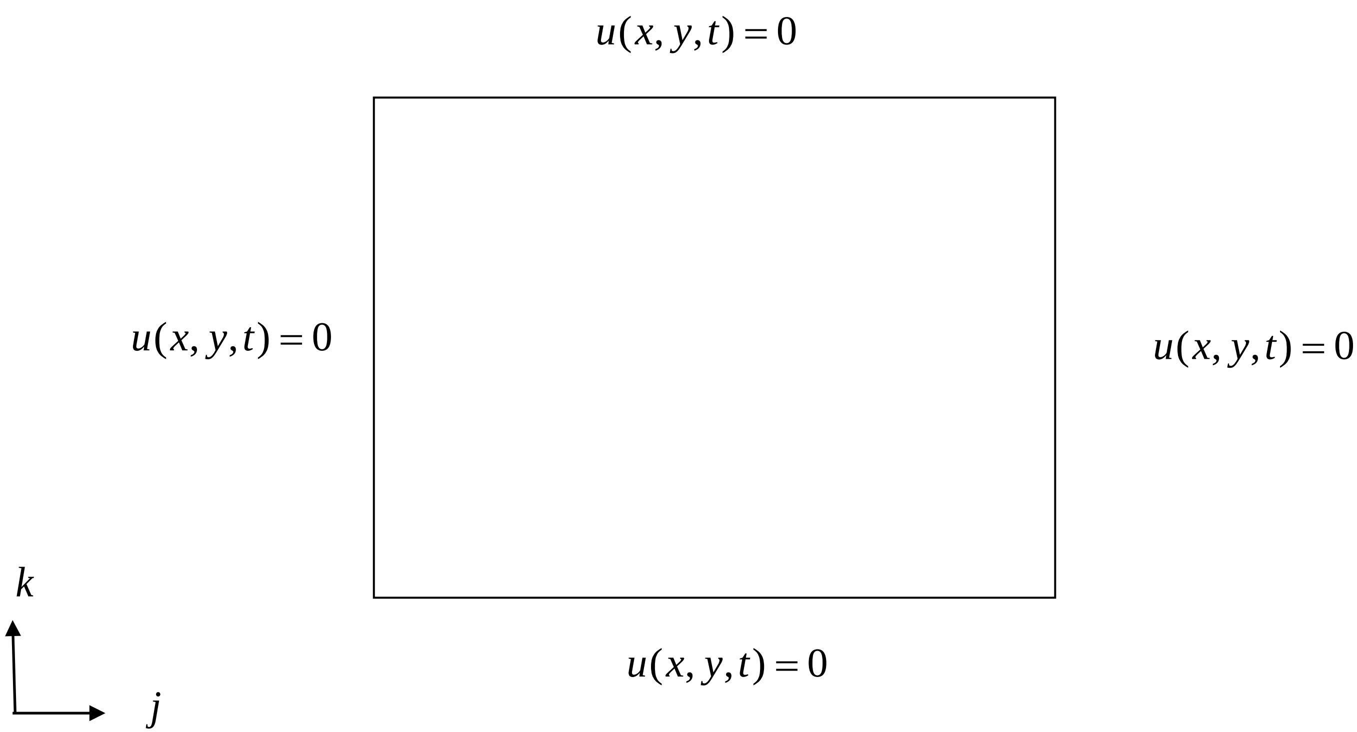
Example: Heat flow equation solved using Finite Difference

- *The mathematical expression, the* ***governing equation, for two dimensional unsteady heat equations is given as***

$$\frac{\partial u}{\partial t} = k\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \qquad (*)$$

1. Boundary conditions

✓ *Boundary conditions are mathematical statements specifying the dependant variable or the derivative of the dependant variable (flux) at the boundaries of the **problem domains**.*



2. *Initial condition*

- ✓ *It refers to the heat distribution everywhere in the system at the beginning of the simulation. Thus the initial condition for the equation is given by*

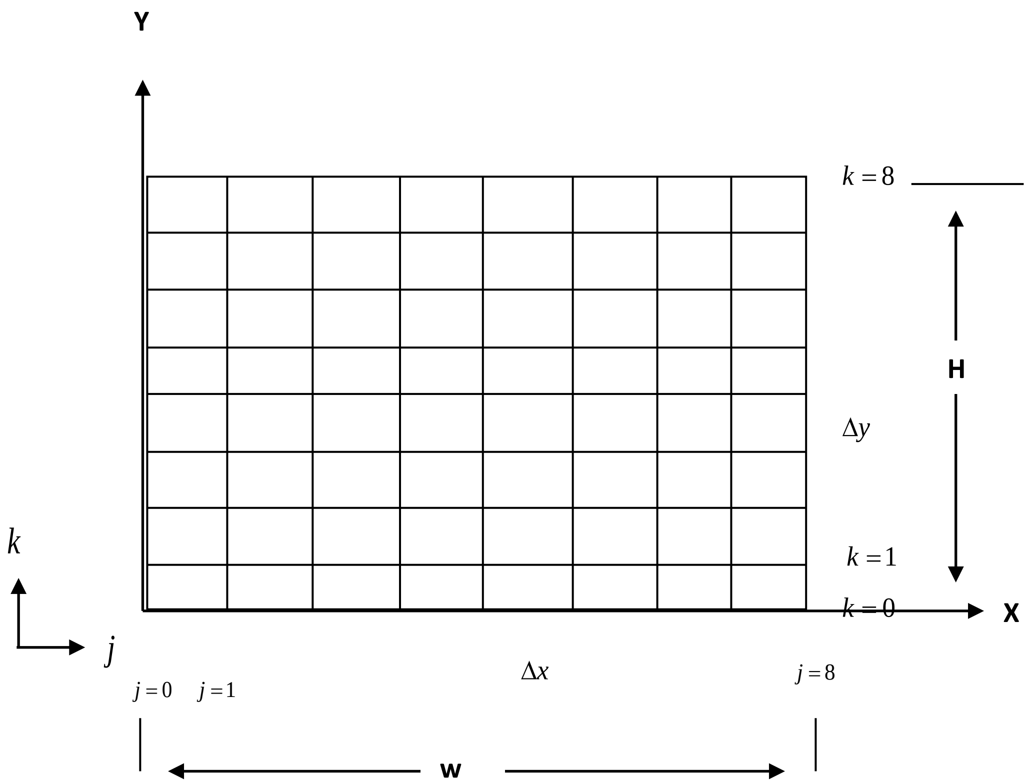
$$u(x, y, 0) = \sin(\pi x) * \sin(\pi y)$$

- ➢ ***Solution: Two approach***
  - ➢ ***Analytical methods***
  - ➢ ***Numerical methods***

➢Analytical solution (Exact solution)

$$u(x, y, t) = e^{-\pi 2t}\sin(\pi t)\cos(\pi t)$$

➢Numerical solution: *Finite difference method*



$for\ldots j = 0,1,2,\ldots N$

$k = 0,1,2\ldots M$

$N = w / \Delta x$

$M = H / \Delta y$

Figure : Finite difference grid

Heat flow equation . . .

The results of the code and corresponding mesh (contour) graph is shown below

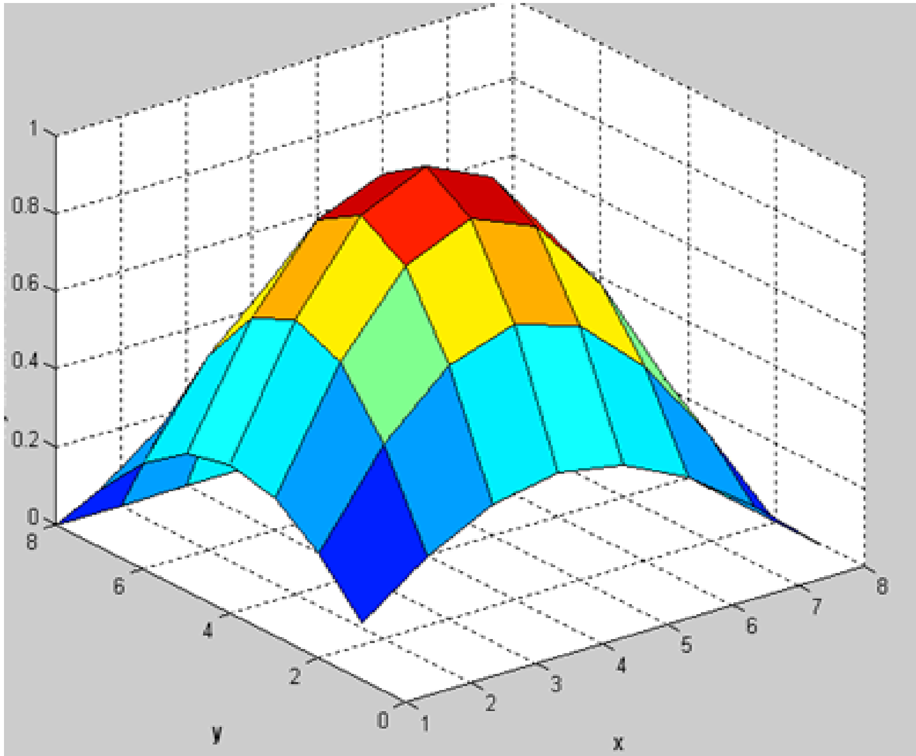
Analytical at t =0

0.1464	0.2706	0.3536	0.3827	0.3536	0.2706	0.1464	0.0000
0.2706	0.5000	0.6533	0.7071	0.6533	0.5000	0.2706	0.0000
0.3536	0.6533	0.8536	0.9239	0.8536	0.6533	0.3536	0.0000
0.3827	0.7071	0.9239	1.0000	0.9239	0.7071	0.3827	0.0000
0.3536	0.6533	0.8536	0.9239	0.8536	0.6533	0.3536	0.0000
0.2706	0.5000	0.6533	0.7071	0.6533	0.5000	0.2706	0.0000
0.1464	0.2706	0.3536	0.3827	0.3536	0.2706	0.1464	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

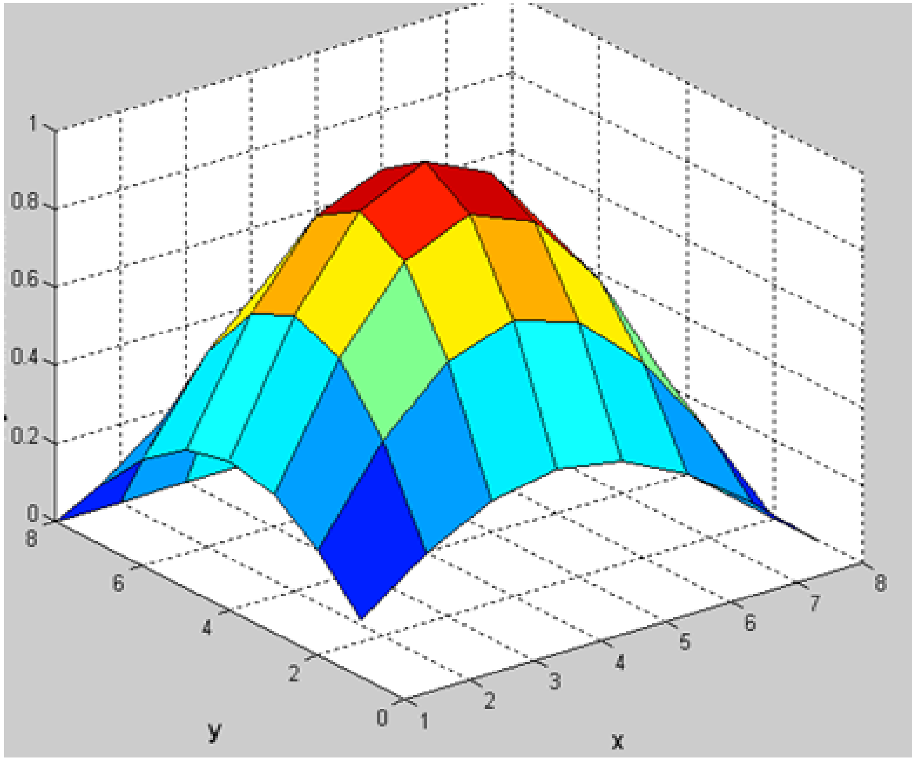
Numerical solution at t=0

0.1464	0.2706	0.3536	0.3827	0.3536	0.2706	0.1464	0.0000
0.2706	0.5000	0.6533	0.7071	0.6533	0.5000	0.2706	0.0000
0.3536	0.6533	0.8536	0.9239	0.8536	0.6533	0.3536	0.0000
0.3827	0.7071	0.9239	1.0000	0.9239	0.7071	0.3827	0.0000
0.3536	0.6533	0.8536	0.9239	0.8536	0.6533	0.3536	0.0000
0.2706	0.5000	0.6533	0.7071	0.6533	0.5000	0.2706	0.0000
0.1464	0.2706	0.3536	0.3827	0.3536	0.2706	0.1464	0.0000
0	0	0	0	0	0	0	0.0000

Heat flow equation . . .



Analytical Solution



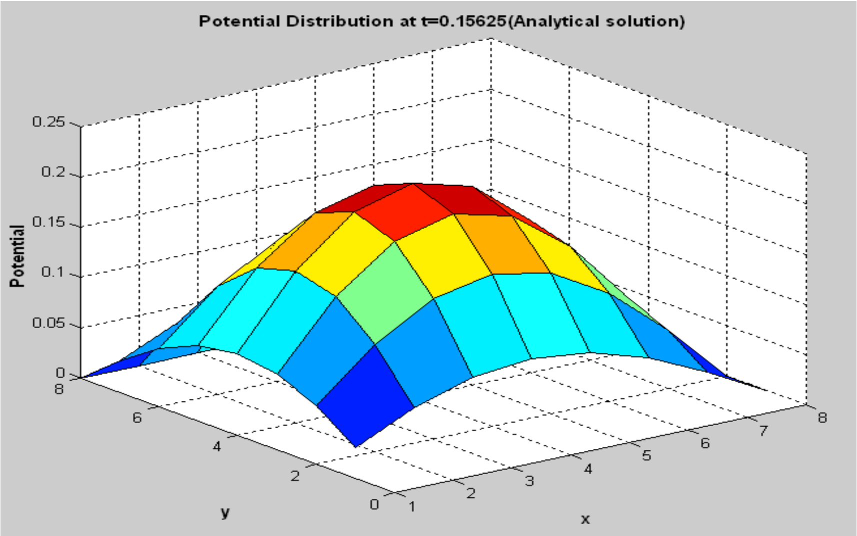
Numerical Solution

There is a complete overlap between analytical and numerical method at time t=0

Heat flow equation . . .

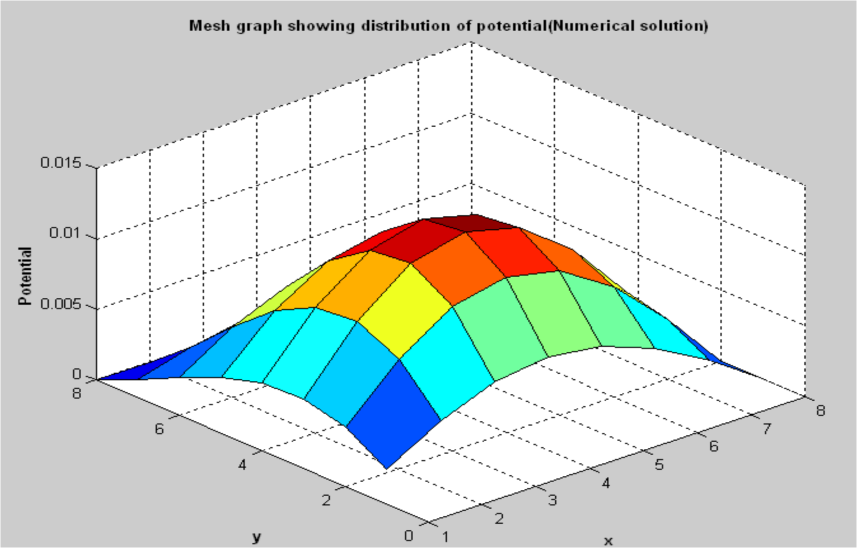
Analytical solution nm=10(t=0.15625)

0.0313	0.0579	0.0756	0.0819	0.0756	0.0579	0.0313	0.0000
0.0579	0.1070	0.1398	0.1513	0.1398	0.1070	0.0579	0.0000
0.0756	0.1398	0.1826	0.1976	0.1826	0.1398	0.0756	0.0000
0.0819	0.1513	0.1976	0.2139	0.1976	0.1513	0.0819	0.0000
0.0756	0.1398	0.1826	0.1976	0.1826	0.1398	0.0756	0.0000
0.0579	0.1070	0.1398	0.1513	0.1398	0.1070	0.0579	0.0000
0.0313	0.0579	0.0756	0.0819	0.0756	0.0579	0.0313	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000



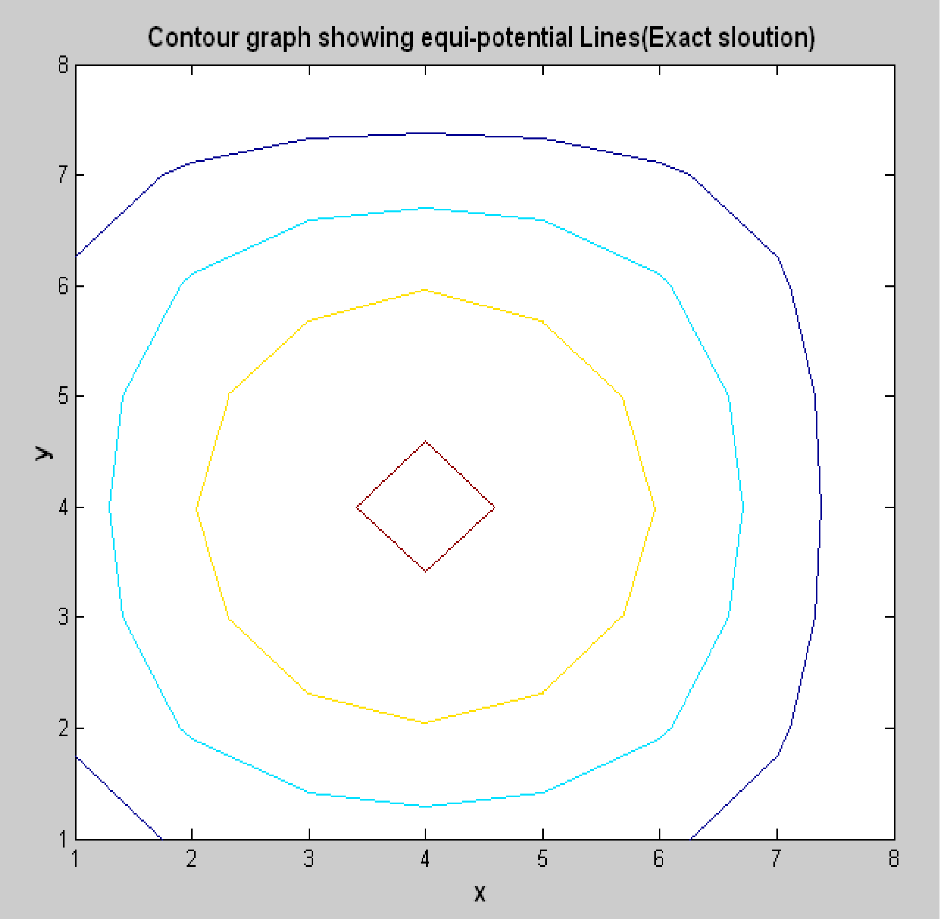
Numerical nm=10(t =0.15625)

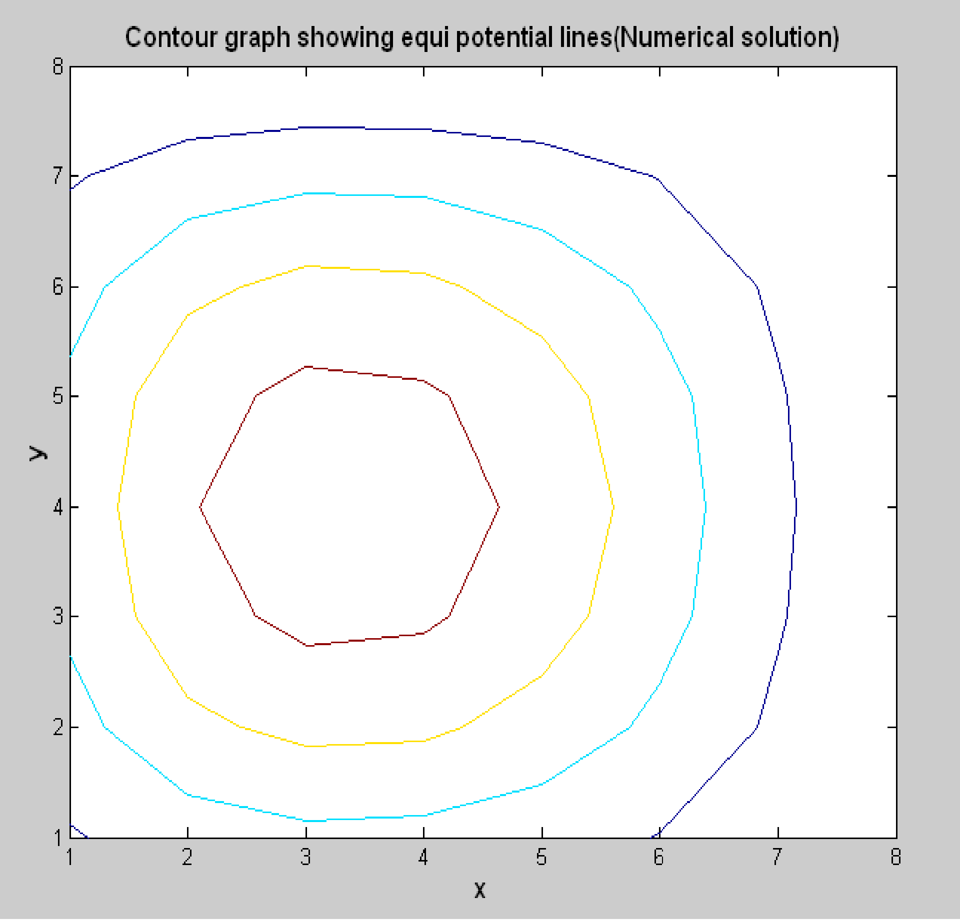
0.0025	0.0042	0.0050	0.0048	0.0040	0.0027	0.0013	0
0.0047	0.0078	0.0092	0.0090	0.0074	0.0050	0.0023	0
0.0062	0.0102	0.0120	0.0117	0.0097	0.0066	0.0031	0
0.0067	0.0111	0.0130	0.0127	0.0105	0.0071	0.0033	0
0.0062	0.0102	0.0120	0.0117	0.0097	0.0066	0.0031	0
0.0047	0.0078	0.0092	0.0090	0.0074	0.0050	0.0023	0
0.0025	0.0042	0.0050	0.0048	0.0040	0.0027	0.0013	0
0	0	0	0	0	0	0	0



Heat flow equation . . .

There corresponding contour graph is shown below

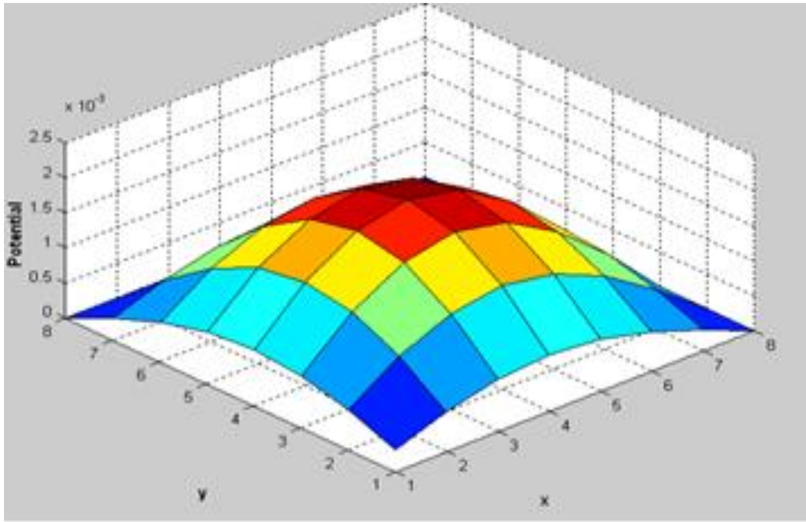




Heat flow equation . . .

Exact solution nm=40(t=.625)

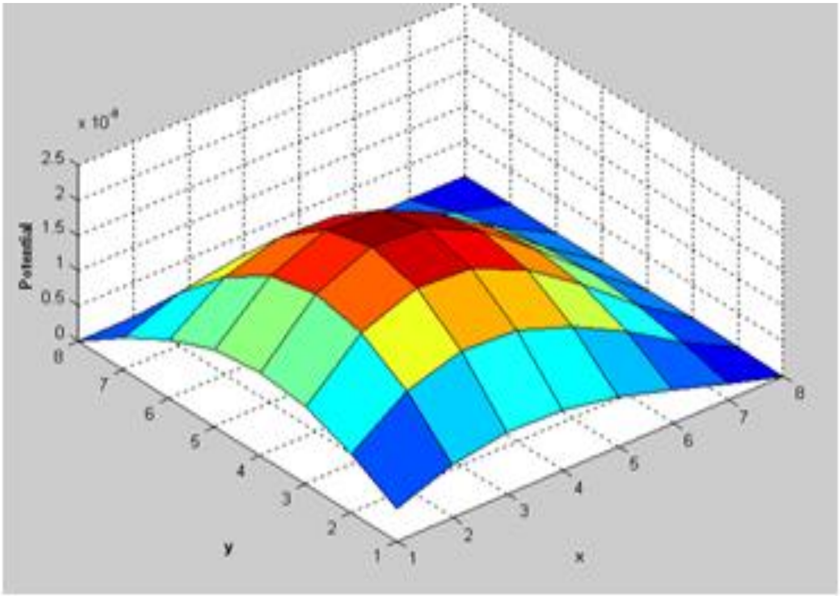
0.0003	0.0006	0.0007	0.0008	0.0007	0.0006	0.0003	0.0000
0.0006	0.0010	0.0014	0.0015	0.0014	0.0010	0.0006	0.0000
0.0007	0.0014	0.0018	0.0019	0.0018	0.0014	0.0007	0.0000
0.0008	0.0015	0.0019	0.0021	0.0019	0.0015	0.0008	0.0000
0.0007	0.0014	0.0018	0.0019	0.0018	0.0014	0.0007	0.0000
0.0006	0.0010	0.0014	0.0015	0.0014	0.0010	0.0006	0.0000
0.0003	0.0006	0.0007	0.0008	0.0007	0.0006	0.0003	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000



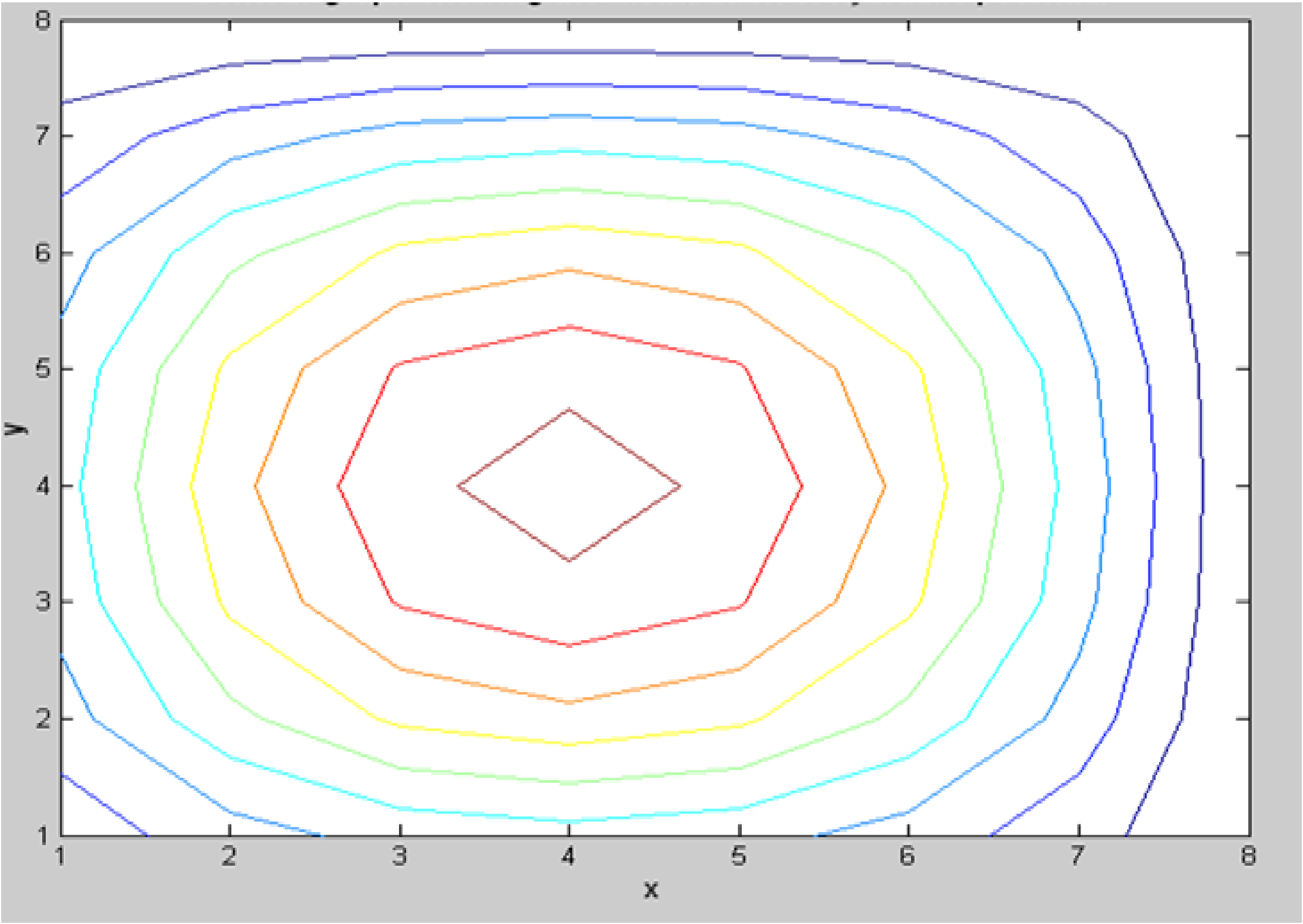
Numerical solution at nm=40(t=.625)

1.0e-007 *

0.0452	0.0752	0.0883	0.0859	0.0711	0.0483	0.0224	0
0.0836	0.1389	0.1632	0.1588	0.1313	0.0893	0.0415	0
0.1092	0.1814	0.2133	0.2074	0.1716	0.1166	0.0542	0
0.1182	0.1964	0.2309	0.2245	0.1857	0.1262	0.0587	0
0.1092	0.1814	0.2133	0.2074	0.1716	0.1166	0.0542	0
0.0836	0.1389	0.1632	0.1588	0.1313	0.0893	0.0415	0
0.0452	0.0752	0.0883	0.0859	0.0711	0.0483	0.0224	0
0	0	0	0	0	0	0	0



Heat flow equation . . .



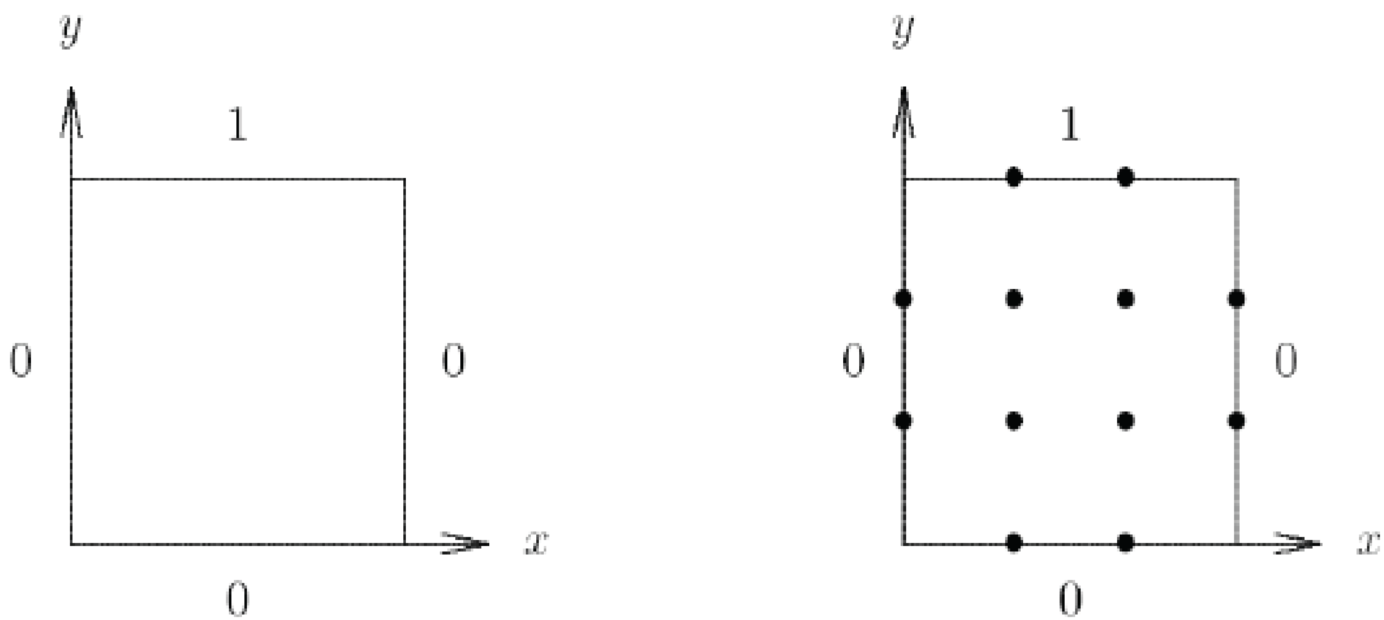
As time went on we obtained several points having equal heat distribution considering no external heat source.

Example: Laplace equation

- Consider Laplace equation

$$u_{xx} + u_{yy} = 0$$

on unit square with boundary conditions shown below left



- Define discrete mesh in domain, including boundaries, as shown above right

Laplace equation, cont.

- Interior grid points where we will compute approximate solution are given by

$$(x_i, y_j) = (ih, jh), \qquad i, j = 1, \ldots, n$$

where in example $n = 2$ and $h = 1/(n+1) = 1/3$

- Next we replace derivatives by centered difference approximation at each interior mesh point to obtain finite difference equation

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0$$

where $u_{i,j}$ is approximation to true solution $u(x_i, y_j)$ for $i, j = 1, \ldots, n$, and represents one of given boundary values if i or j is 0 or $n+1$

Laplace equation, cont.

- Simplifying and writing out resulting four equations explicitly gives

$$4u_{1,1} - u_{0,1} - u_{2,1} - u_{1,0} - u_{1,2} = 0$$

$$4u_{2,1} - u_{1,1} - u_{3,1} - u_{2,0} - u_{2,2} = 0$$

$$4u_{1,2} - u_{0,2} - u_{2,2} - u_{1,1} - u_{1,3} = 0$$

$$4u_{2,2} - u_{1,2} - u_{3,2} - u_{2,1} - u_{2,3} = 0$$

Laplace equation, cont.

- Writing previous equations in matrix form gives

$$A x = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{1,2} \\ u_{2,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = b$$

- System of equations can be solved for unknowns $u_{i,j}$ either by direct method based on factorization or by iterative method, yielding solution

$$x = \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{1,2} \\ u_{2,2} \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.125 \\ 0.375 \\ 0.375 \end{bmatrix}$$

Generally,

- Finite difference methods for such problems proceed as before
  - Define discrete mesh of points within domain of equation
  - Replace derivatives in PDE by finite difference approximations
  - Seek numerical solution at mesh points

- Unlike time-dependent problems, solution is not produced by marching forward step by step in time

- Approximate solution is determined at all mesh points simultaneously by solving single system of algebraic equations

Laplace equation, cont.

- In practical problem, mesh size h would be much smaller, and resulting linear system would be much larger
- Matrix would be very sparse, however, since each equation would still involve only five variables, thereby saving substantially on work and storage

2.3 Methods To Solve Linear System

Direct Methods

- ***Matrix Inverse Method***
- ***Cramer's Method***
- ***Gaussian Eliminations***

Iterative methods

- ***Gauss Jacobi Method***
- ***Gauss Seidel Method***

We want to solve the following linear system

$$Ax = b$$

Example

$$\begin{aligned}
10x_1 - \quad x_2 + 2x_3 \qquad\quad &= 6 \\
-x_1 + 11x_2 - \quad x_3 + 3x_4 &= 25 \\
2x_1 - \quad x_2 + 10x_3 - x_4 &= -11 \\
3x_2 - \quad x_3 - 8x_4 &= 15
\end{aligned}$$

$$\underbrace{\begin{bmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ 25 \\ -11 \\ 15 \end{bmatrix}}_{b}$$

Jacobi Iterative Method

We want to solve the following linear system

Example

solve the linear system

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 3x_2 - x_3 - 8x_4 &= 15 \end{aligned}$$

Keep the diagonal on the left hand side

$$\begin{aligned} 10x_1 &= x_2 - 2x_3 + 6 \\ 11x_2 &= x_1 + x_3 - 3x_4 + 25 \\ 10x_3 &= -2x_1 + x_2 + x_4 - 11 \\ -8x_4 &= -3x_2 + x_3 + 15 \end{aligned}$$

Change the coeff of LHS to be one by division

$$\begin{aligned} x_1 &= (x_2 - 2x_3 + 6)/10 \\ x_2 &= (x_1 + x_3 - 3x_4 + 25)/11 \\ x_3 &= (-2x_1 + x_2 + x_4 - 11)/10 \\ x_4 &= (-3x_2 + x_3 + 15)/(-8) \end{aligned}$$

From the initial approx. $\boldsymbol{x}^{(0)} = (0,0,0,0)^T$ we *get* $\boldsymbol{x}^{(1)}$

$$\begin{aligned} x_1^{(1)} &= (x_2^{(0)} - 2x_3^{(0)} + 6)/10 \\ x_2^{(1)} &= (x_1^{(0)} + x_3^{(0)} - 3x_4^{(0)} + 25)/11 \\ x_3^{(1)} &= (-2x_1^{(0)} + x_2^{(0)} + x_4^{(0)} - 11)/10 \\ x_4^{(1)} &= (-3x_2^{(0)} + x_3^{(0)} + 15)/(-8) \end{aligned}$$

Jacobi Iterative Method

From the initial approx. $\boldsymbol{x}^{(0)} = (0,0,0,0)^T$ **we** $\boldsymbol{x}^{(1)}$

$$x_1^{(1)} = (\qquad x_2^{(0)} - 2x_3^{(0)} \qquad + 6)/10$$

$$x_2^{(1)} = (\quad x_1^{(0)} \qquad + x_3^{(0)} - 3x_4^{(0)} + 25)/11$$

$$x_3^{(1)} = (-2x_1^{(0)} + \; x_2^{(0)} \qquad + \; x_4^{(0)} - 11)/10$$

$$x_4^{(1)} = (\qquad - 3x_2^{(0)} + x_3^{(0)} \qquad + 15)/(-8)$$

$$x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad x^{(1)} = \begin{bmatrix} 6/10 \\ 25/11 \\ -11/10 \\ -15/8 \end{bmatrix} = \begin{bmatrix} 0.600 \\ 2.272 \\ -1.100 \\ -1.875 \end{bmatrix}$$

From the k-th approx. $\boldsymbol{x}^{(k)} = (0,0,0,0)^T$ **we** $\boldsymbol{x}^{(k+1)}$

$$x_1^{(k+1)} = (\qquad x_2^{(k)} - 2x_3^{(k)} \qquad + 6)/10$$

$$x_2^{(k+1)} = (\quad x_1^{(k)} \qquad + x_3^{(k)} - 3x_4^{(k)} + 25)/11$$

$$x_3^{(k+1)} = (-2x_1^{(k)} + \; x_2^{(k)} \qquad + \; x_4^{(k)} - 11)/10$$

$$x_4^{(k+1)} = (\qquad - 3x_2^{(k)} + x_3^{(k)} \qquad + 15)/(-8)$$

$$x^{(1)} = \begin{bmatrix} 0.600 \\ 2.272 \\ -1.100 \\ -1.875 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 1.047 \\ 1.715 \\ -0.805 \\ 0.885 \end{bmatrix}$$

Jacobi Iterative Method

$$x_1^{(k+1)} = (\qquad x_2^{(k)} - 2x_3^{(k)} \qquad + 6)/10$$

$$x_2^{(k+1)} = (\quad x_1^{(k)} \qquad + x_3^{(k)} - 3x_4^{(k)} + 25)/11$$

$$x_3^{(k+1)} = (-2x_1^{(k)} + \ x_2^{(k)} \qquad + \ x_4^{(k)} - 11)/10$$

$$x_4^{(k+1)} = (\qquad - 3x_2^{(k)} + x_3^{(k)} \qquad + 15)/(-8)$$

$$x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$x_1^{(k)}$	0.6000	1.0473	0.9326	1.0152	0.9890
$x_2^{(k)}$	2.2727	1.7159	2.0533	1.9537	2.0114
$x_3^{(k)}$	-1.1000	-0.8052	-1.0493	-0.9681	-1.0103
$x_4^{(k)}$	1.8750	0.8852	1.1309	0.9738	1.0214

	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
$x_1^{(k)}$	1.0032	0.9981	1.0006	0.9997	1.0001
$x_2^{(k)}$	1.9922	2.0023	1.9987	2.0004	1.9998
$x_3^{(k)}$	-0.9945	-1.0020	-0.9990	-1.0004	-0.9998
$x_4^{(k)}$	0.9944	1.0036	0.9989	1.0006	0.9998

$$x^* = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}$$

Gauss-Seidel Method

➢ ***Jacobi iteration does not use the most recently available information. However, Gauss-Seidel does***

From the k-th approx. $\boldsymbol{x}^{(k)} = (0,0,0,0)^T$ **we** $\boldsymbol{x}^{(k+1)}$

$$
\begin{aligned}
x_1^{(k+1)} &= (\quad x_2^{(k)} - 2x_3^{(k)} + 6)/10 \\
x_2^{(k+1)} &= (\quad x_1^{(k)} + x_3^{(k)} - 3x_4^{(k)} + 25)/11 \\
x_3^{(k+1)} &= (-2x_1^{(k)} + x_2^{(k)} + x_4^{(k)} - 11)/10 \\
x_4^{(k+1)} &= (\quad -3x_2^{(k)} + x_3^{(k)} + 15)/(-8)
\end{aligned}
$$

From the k-th approx. $\boldsymbol{x}^{(k)} = (0,0,0,0)^T$ **we** $\boldsymbol{x}^{(k+1)}$

$$
\begin{aligned}
x_1^{(k+1)} &= (\quad x_2^{(k)} - 2x_3^{(k)} + 6)/10 \\
x_2^{(k+1)} &= (\quad x_1^{(k+1)} + x_3^{(k)} - 3x_4^{(k)} + 25)/11 \\
x_3^{(k+1)} &= (-2x_1^{(k+1)} + x_2^{(k+1)} + x_4^{(k)} - 11)/10 \\
x_4^{(k+1)} &= (\quad -3x_2^{(k+1)} + x_3^{(k+1)} + 15)/(-8)
\end{aligned}
$$

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
$x_1^{(k)}$	0.6000	1.0302	1.0066	1.0009	1.0001
$x_2^{(k)}$	2.3273	2.0369	2.0036	2.0003	2.0000
$x_3^{(k)}$	-0.9873	-1.0145	-1.0025	-1.0003	-1.0000
$x_4^{(k)}$	0.8789	0.9843	0.9984	0.9998	1.0000

Note that Jacobi's method in this example required twice as many iterations for the same accuracy.

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Jacobi iteration for general n:

$$\mathrm{for\,i = 1:n}$$

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii}$$

$$end$$

Gauss-Seidel iteration for general n:

$$\mathrm{for\,i = 1:n}$$

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii}$$

$$end$$

End of Chapter 2