



Aalto University
School of Engineering

Lecture 2: Finite Difference Methods in Heat Transfer

V. Vuorinen

Aalto University School of Engineering
Heat and Mass Transfer Course, Autumn 2016

November 1st 2017, Otaniemi
ville.vuorinen@aalto.fi

Overview

Part 1 (“must know”)

- Motivation by the “heat sink” problem (HW1 & HW2 + ILO’s)
- Solution of 1d heat eqn via finite differences
- Discretization in space and time
- Ghost cells
- Stability

Part 2 (“good to know”/very useful in HW2)

- 2d finite difference method
- Computational examples using a 2d FD-code

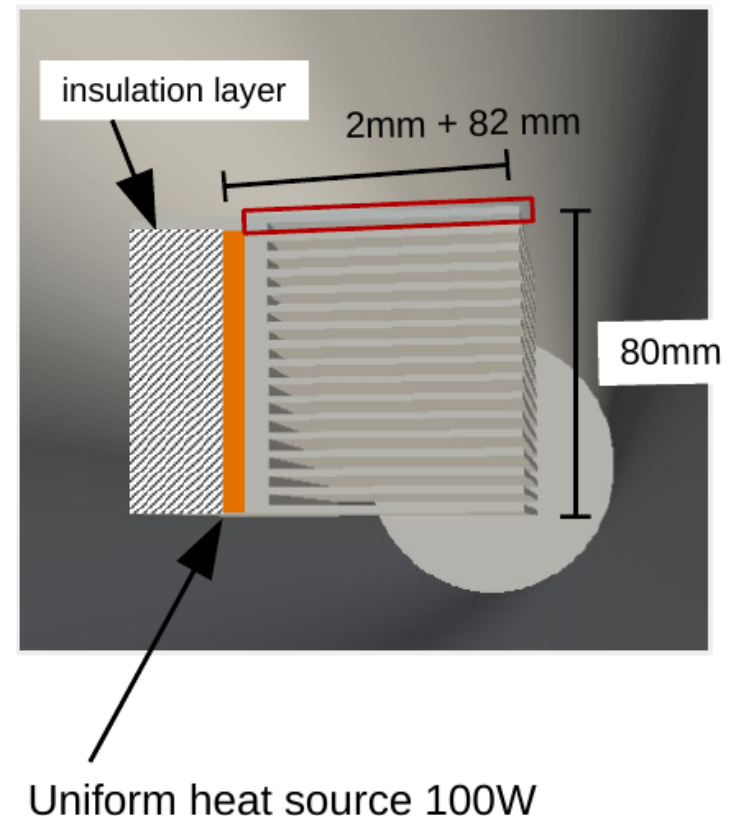
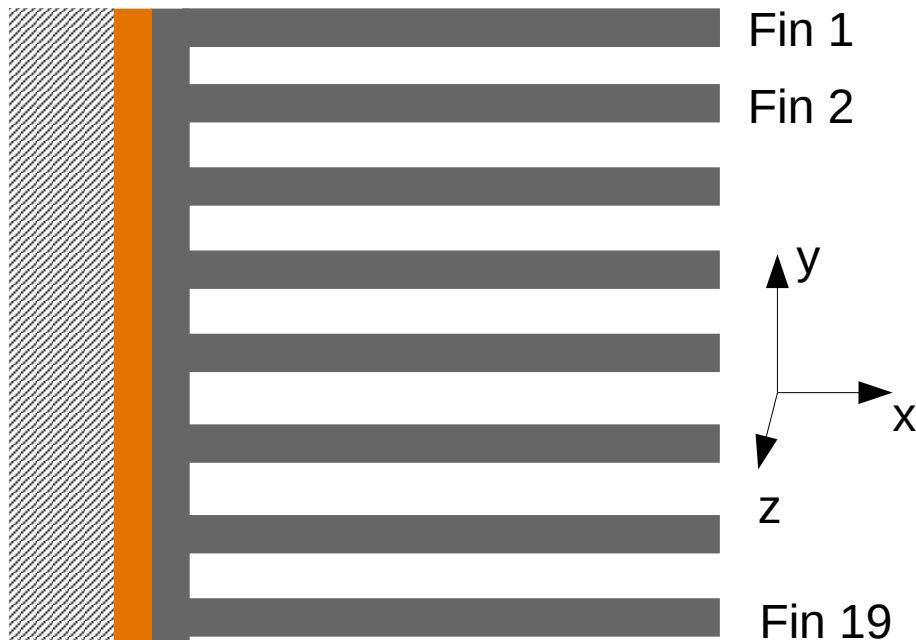


Aalto University
School of Engineering

Part 1: 1 dimensional finite difference method

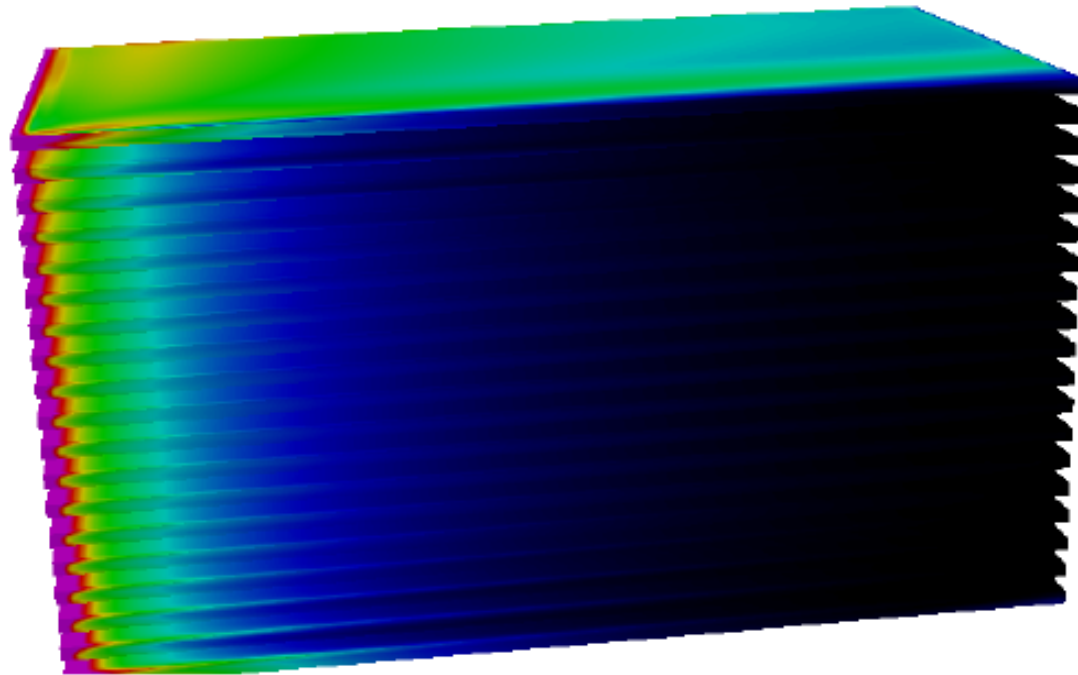
Recall: One of the Key ILO's is the Heat Sink

Problem: Conduction + Convective Transport

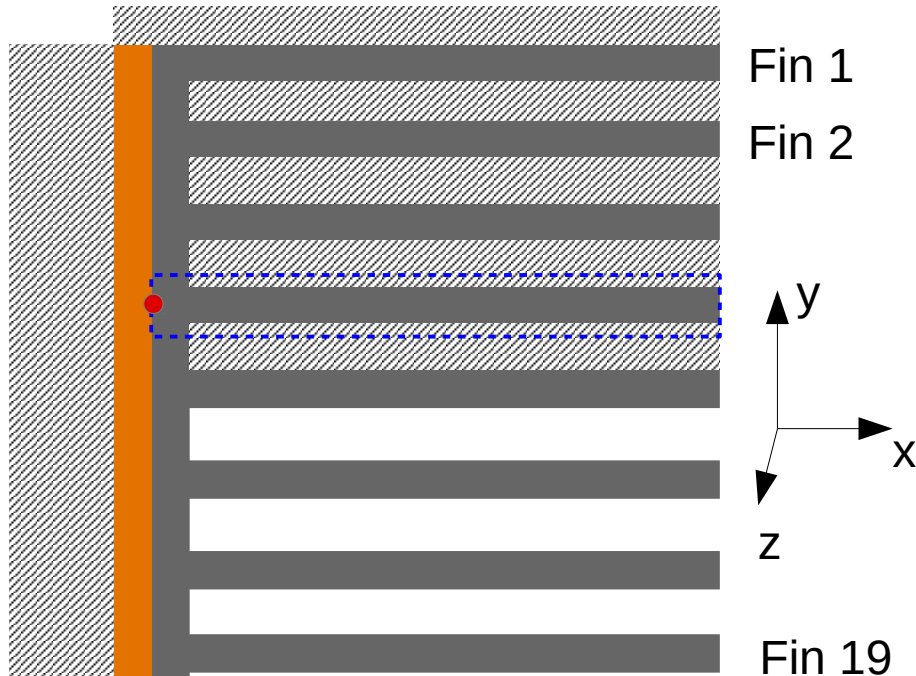


- Heat equation inside the solid material (aluminium). What boundary conditions ?
- Navier-Stokes equations in the gas, primary direction of convection is z. What boundary conditions?

Average flow direction

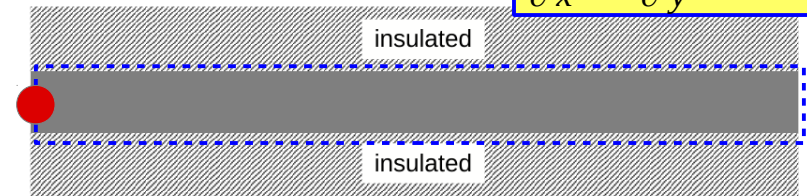


HW1: Find temperature distribution numerically along one fin (assume that gaps insulated)



a) 1d approximation of conduction from base to the fin tip

$$\frac{\partial^2 T}{\partial x^2} \gg \frac{\partial^2 T}{\partial y^2} \rightarrow T = T(x, t)$$



Boundary condition type 1: temperatures given

T_{left}

T_{right}

Boundary condition type 2: temperature at right given
but flux provided at left

q_{left}

T_{right}

Assumptions:

- (100/19) W of heat leaves through each fin.
- 1d conduction in aluminium
- heat enters from left and exits only from right
- gaps between fins are insulated

Discretization by Finite Difference Method

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

Recall Taylor series and basic discretization formulae for derivatives

Time derivative in cell i at timestep n

$$\left(\frac{\partial T}{\partial t} \right)_i^n \approx \frac{T_i^{n+1} - T_i^n}{\Delta t}$$

Second space derivative in cell i at timestep n

$$\left(\frac{\partial^2 T}{\partial x^2} \right)_i^n \approx \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$



Known from previous timestep n:

$$T_i^n$$

Unknown on the n+1 timestep:

$$T_i^{n+1}$$

Discretization by Finite Difference Method

Continuous version of heat equation

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

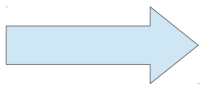
Discrete version of heat equation

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

$$CFL = \frac{\alpha \Delta t}{\Delta x^2}$$

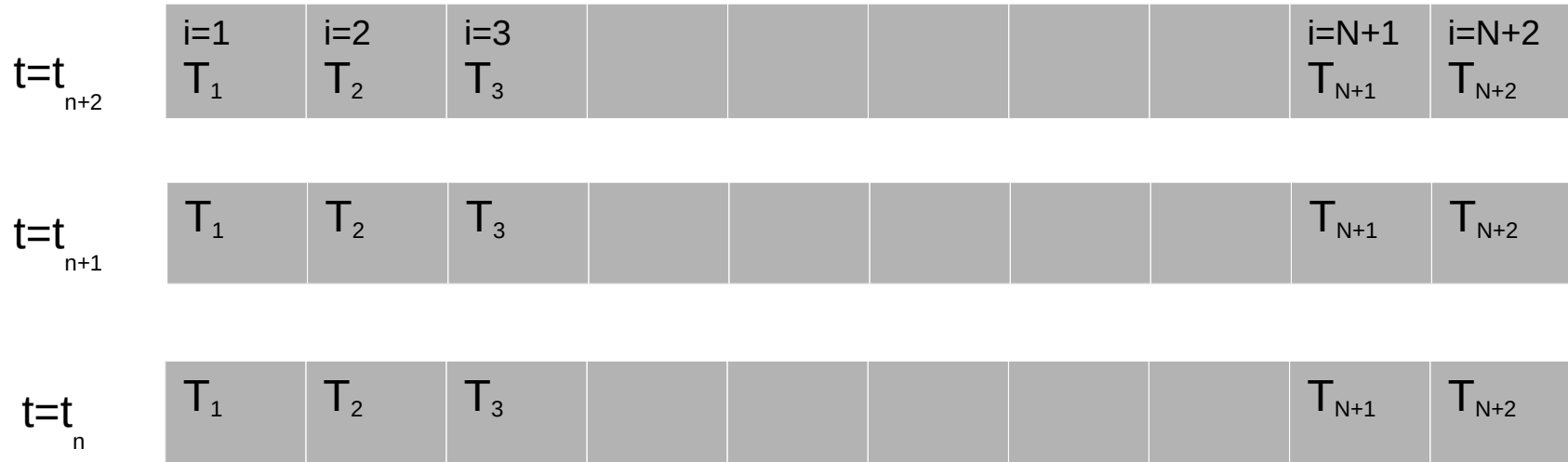
Courant-Friedrichs-Lewy number (CFL < 0.5 for stability).

$$T_i^{n+1} = T_i^n + \Delta t \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$



Now we have an explicit update scheme for T in each discrete grid point i. This is the explicit Euler scheme (most simple timestepping).

Discrete Representation of the Scheme

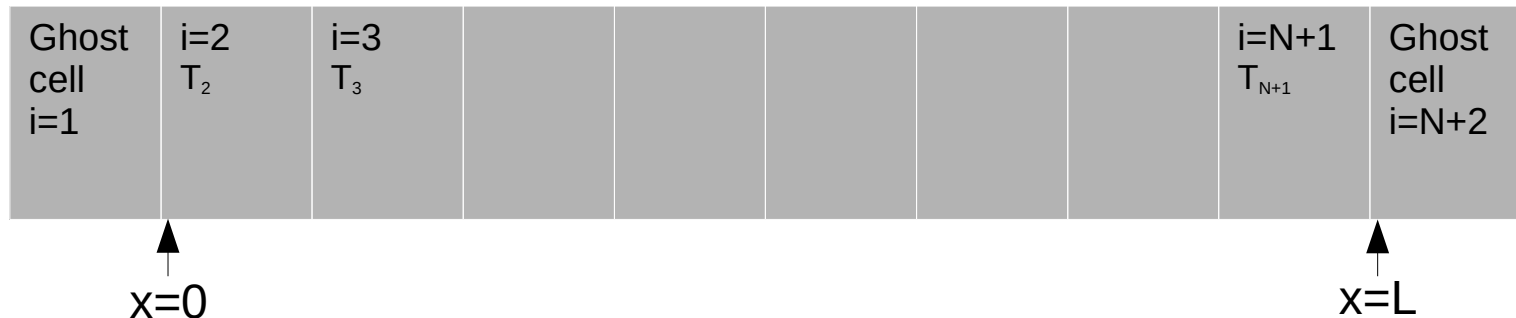


$$t_n = n\Delta t, n=0,1,2,\dots$$

$$T_i^{n+1} = T_i^n + \Delta t \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

Boundary Condition Types

- **The problem:** some numerical value needs to be assigned to the **"ghost cells"**
- **Otherwise:** we can not calculate second derivative of T in cells $i=2$ and $i=N+1$
- **Case 1:** Given temperature value (Dirichlet or "fixed value") → heat flux through boundary
- **Case 2:** Isolated (Neumann or "zero-gradient") → no heat flux through boundary
- **Case 3:** Given heat flux → heat flux through boundary



Case 1:

$$(T_1^n + T_2^n) / 2 = T_{min}$$

Case 2:

$$T_1^n = T_2^n$$

Case 3:

$$-k (T_2^n - T_1^n) / \Delta x = q_L$$

$$(T_{N+1}^n + T_{N+2}^n) / 2 = T_{max}$$

$$T_{N+1}^n = T_{N+2}^n$$

$$k (T_{N+2}^n - T_{N+1}^n) / \Delta x = q_R$$

Summary of the Numerical Solution Scheme for 1d Heat Equation

1) Set boundary conditions to cells 1 and N+2 using T from step n.

2) Update new temperature at timestep n+1 in the internal cells 2...N+1

3) Update time according to $t = t + \Delta t$

4) Go back to 1)

T_i^n (Known)

$$T_i^{n+1} = T_i^n + \Delta t \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$



T_i^{n+1}

$$t_{n+1} = t_n + \Delta t$$

This Scheme is Extremely Short to Program in Matlab

Program: /Example1d/HeatDiffusion.m

Execution: >> HeatDiffusion

What it does: Solves 1d heat equation in equispaced grid, fixed T_{left} and T_{right} .

Main for-loop:

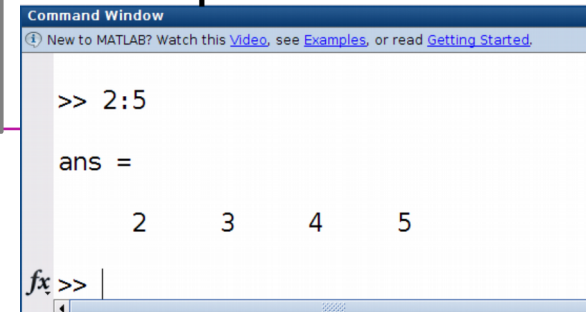
```
for (t=1:K)
    % set boundary conditions
    T(1) = 2*Tleft - T(2); T(N+2) = 2*Tright - T(N+1);

    % update temperature in inner points
    T(in) = T(in) + (dt*kappa/dx^2)*(T(in+1)-2*T(in)+T(in-1));
end
```

Note: I use constantly the “trick” which makes Matlab-programs often very fast.

```
% define a table which refers to the 'inner points'
in = 2:(N+1);
```

Example for $N+1 = 5$



Command Window

New to MATLAB? Watch this [video](#), see [Examples](#), or read [Getting Started](#).

```
>> 2:5

ans =

     2     3     4     5
```

fx >> |

The Following Folders and Files for Matlab Programs Provided (Week 1)

/Example0d/

cool0d.m

Execute by

>> cool0d

/Examples1d/

HeatDiffusion.m

Execute by e.g.

>> HeatDiffusion

ConvectionDiffusion.m

/HowToPlot/

DrawingSurface.m

Demos on plotting Figures.

Execute by e.g.:

PlottingFigure.m

>> DrawingSurface

SurfaceAnimation.m

/Examples2d/

CaseDefinition.m computedT.m GradX.m

GradY.m HeatDiffusion2d.m project.m

2d heat transfer code.

Execute by:

solveTemperature.m

>> HeatDiffusion2d

circle.m DivDiv.m GradXskew.m

GradYskew.m Laplacian.m setTBCs.m

visualizeResults.m





Aalto University
School of Engineering

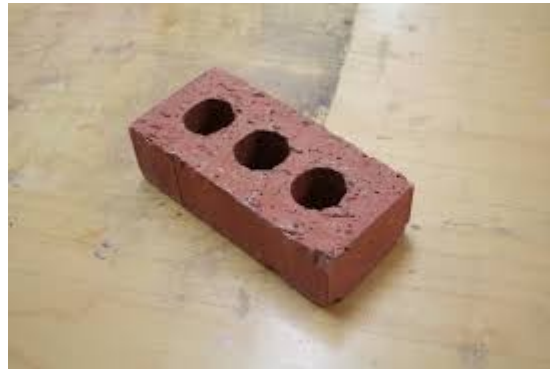
Part 2: 2 dimensional finite difference methods

Overview

- Previously you learned about 0d and 1d heat transfer problems and their numerical solution
- Here we extend things into 2d (3d) cases which is straightforward
- We consider the simple case of a square domain
- Real geometries: Welcome to CFD (spring) & CFD-modeling (autumn) courses
- Real geometries: finite volume methods or finite element methods

Thinking task

- How would you model heat transfer across a brick wall ?
- What could be boundary conditions for brick heat transfer ?
- How could you model the holes ?
- Why are there holes ?
- How can you explain the holes by heat conductivity or thermal resistance



Discretization by Finite Difference Method

General form of heat equation

$$\frac{\partial T}{\partial t} = \nabla \cdot \alpha \nabla T$$

Terms opened in 2d

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \alpha \frac{\partial T}{\partial x} + \frac{\partial}{\partial y} \alpha \frac{\partial T}{\partial y}$$

Time derivative in cell (i, j) at timestep n

$$\left(\frac{\partial T}{\partial t} \right)_{i,j}^n \approx \frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t}$$

Second space derivatives at cell (i,j)

X:

$$\left(\frac{\partial^2 T}{\partial x^2} \right)_{i,j}^n \approx \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2}$$

Y:

$$\left(\frac{\partial^2 T}{\partial y^2} \right)_{i,j}^n \approx \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2}$$

Discretization by Finite Difference Method

$$CFL = \frac{\alpha \Delta t}{\Delta x^2}$$

Explicit Euler timestepping for 2d heat equation:

$$T_{i,j}^{n+1} = T_{i,j}^n + \Delta t \alpha \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} + \Delta t \alpha \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2}$$

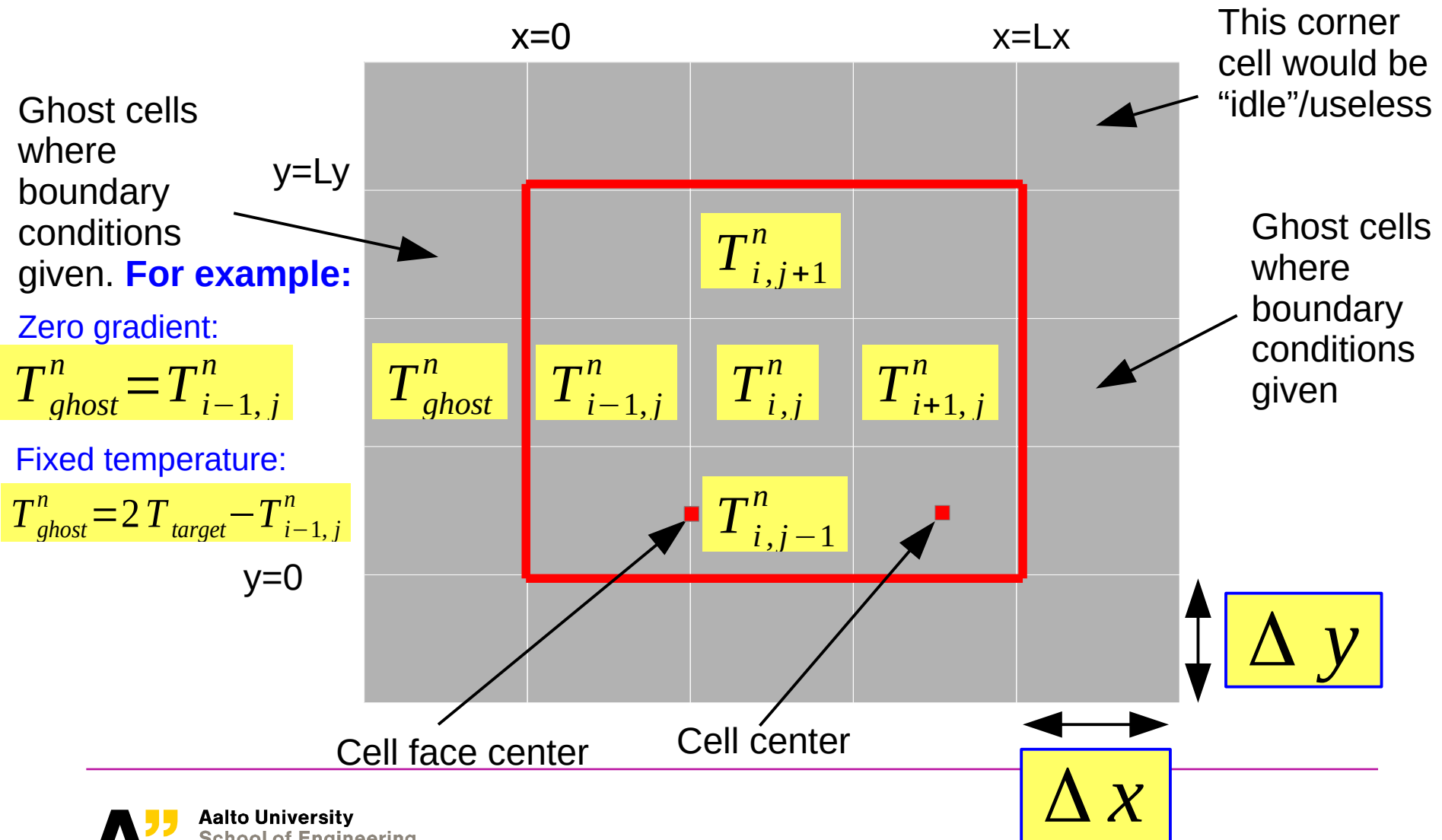
Which is equal to the “delta” form:

$$\Delta T_{i,j}^n = \Delta t \alpha \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} + \Delta t \alpha \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2}$$

Where:

$$\Delta T_{i,j}^n = T_{i,j}^{n+1} - T_{i,j}^n$$

Visualization of 2d Cartesian Grid



Summary of the Numerical Solution Scheme for 2d Heat Equation

1) Set boundary conditions (BC's) to the ghost cells using T from step n .

$T_{i,j}^n$ (Known and hence BC update possible)

2) Update new temperature at timestep $n+1$ in the internal cells

$$T_{i,j}^{n+1} = T_{i,j}^n + \Delta t \alpha \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} + \Delta t \alpha \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2}$$

3) Update time according to $t = t + dt$



$T_{i,j}^{n+1}$

$t_{n+1} = t_n + \Delta t$

4) Go back to 1)

Also This Scheme is Extremely Short to Program in Matlab

Program: /Examples2d/HeatDiffusion2d.m

Execution: >> **HeatDiffusion2d**

What it does: Solves 2d heat equation in Cartesian grid, various BC's possible. It is also possible to simulate materials with variable heat diffusivity to simulate conduction in e.g. “layered” materials.

The “heart” of this 2d code is the computation of **dT** in the .m file **computedT.m**

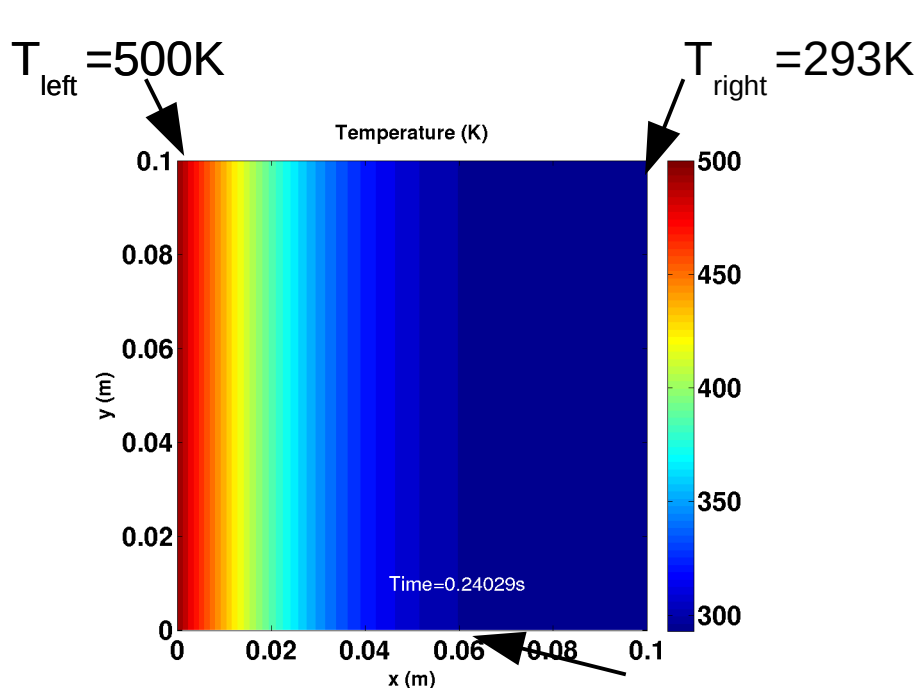
```
dT=dt*DivDiv(T,nu,east,west,north,south,inx,iny,dx,dy);
```

- In short the “cryptic” function **DivDiv.m** is needed to evaluate divergence of the diffusive flux (conservatively) i.e. $\nabla \cdot \alpha \nabla T$
- The contents of the function **DivDiv.m** looks lengthy but it is so only to account for the fact that the code assumes *diffusivity* (x,y) - dependent.

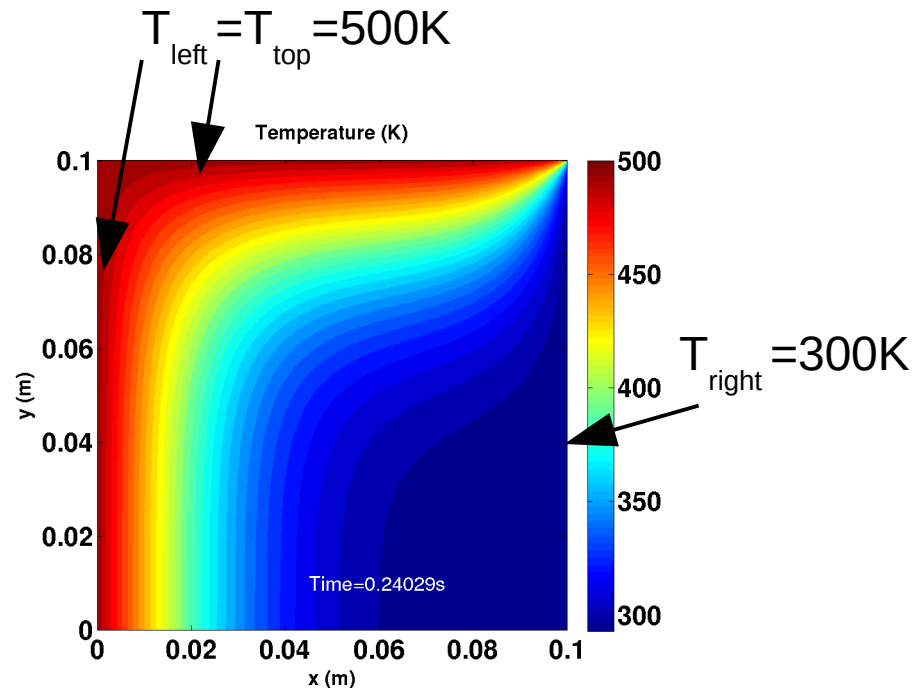
The function solveTemperature.m applies the 4th order Runge-Kutta method (for better stability than Euler) to evaluate dT several times and finally accumulates them together to end up in essentially the same outcome as Euler method.

$$T_{new} = T_{old} + \Delta T$$

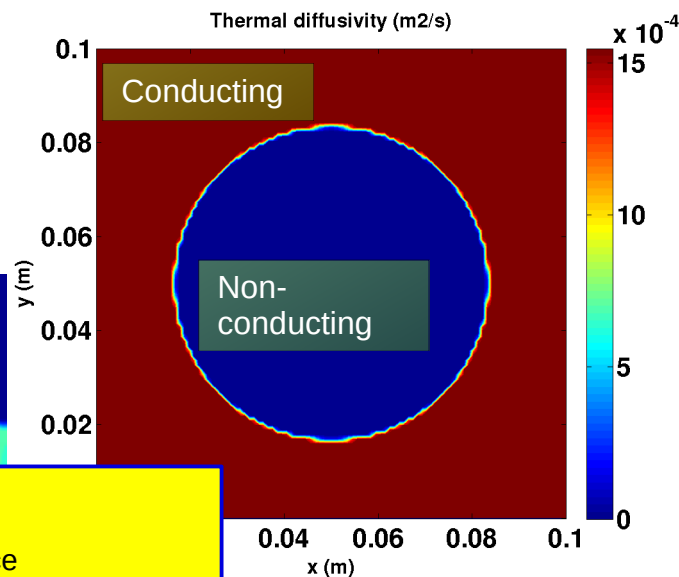
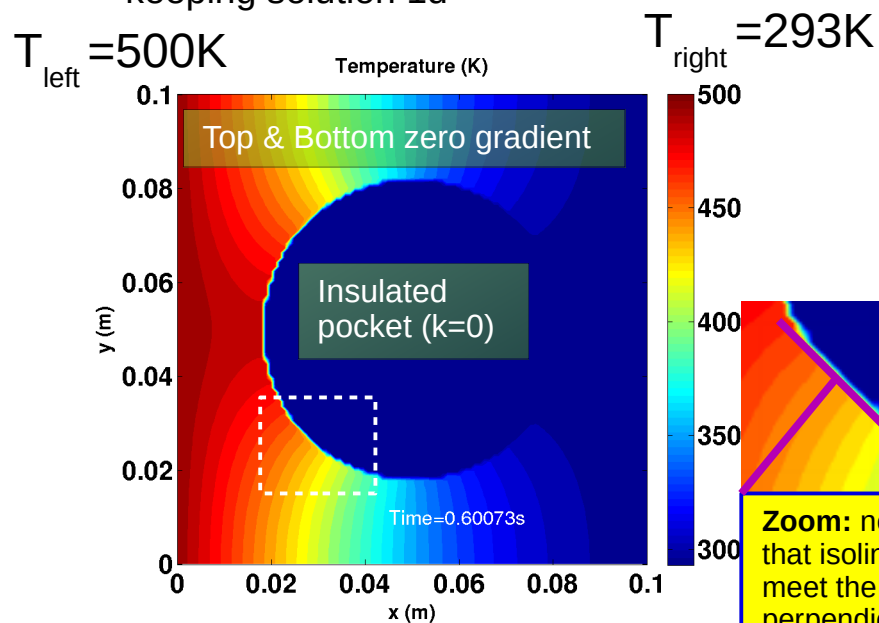




Bottom and top walls are zero gradient
keeping solution 1d



Bottom wall is zero temperature gradient
i.e. thermally insulated



Thank you for your attention!



Aalto University
School of Engineering