FOAM

@PT

Guimarães
10-11
Julho
2015

Universidade do Minho
Escola de Engenharia
Dep. de Eng. de Polímeros

iPC i3N
INSTITUTE FOR POLYMERS AND COMPOSITES

bluecaPe
Computer Applications in Science & Engineering

I Encontro em Portugal de utilizadores da tecnologia Open∇FOAM

Introduction to OpenFOAM

L.L. Ferrás, C. Fernandes, J.M. Nóbrega
Institute for Polymers and Composites (i3N), University of Minho

Programa Geral

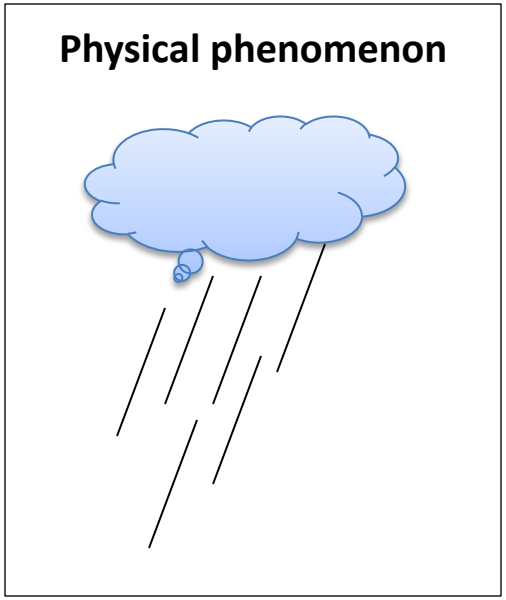
	10 Julho (sexta-feira)		11 Julho (sábado)
8:45 9:00	Sessão de Abertura		
9:00 10:00	Curso Básico (Sessões B1 e B2) **B1. Introdução ao OpenFOAM 9:00-10:30**	Curso Avançado (Sessões A1 e A2)	OpenFOAM: History, current capabilities and future perspectives (Hrvoje Jasak)
10:00 12:30	**B2. Geração de malha com o blockMesh 11:00-12:30**		Apresentações de trabalhos desenvolvidos em OpenFOAM
12:30 14:00	Almoço (Restaurante Vila Flor)		
14:00 17:30	Curso Básico (Sessões B3 e B4)	Curso Avançado (Sessões A3 e A4)	Reunião Geral do grupo português de utilizadores de OpenFOAM

What is OpenFOAM (open Field Operation and Manipulation)?

OpenFOAM is a powerful tool that allows the numerical solution of differential equations (usually found in continuum mechanics).

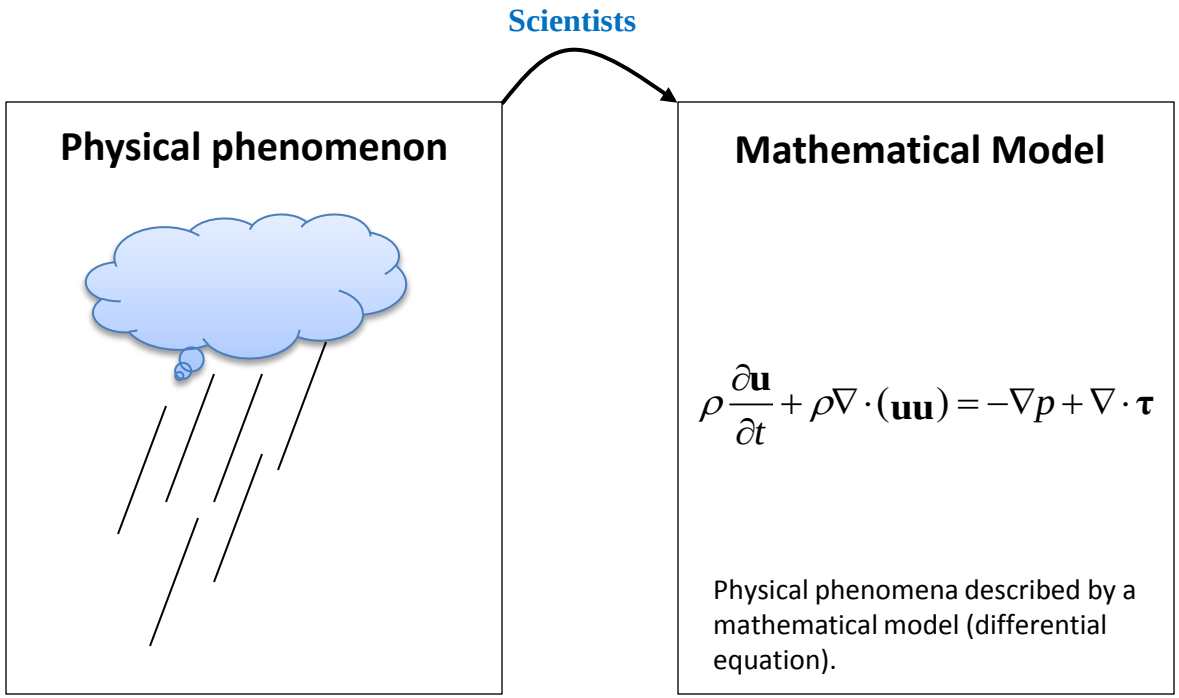
What is OpenFOAM (open Field Operation and Manipulation)?

OpenFOAM is a powerful tool that allows the numerical solution of differential equations (usually found in continuum mechanics).



What is OpenFOAM (open Field Operation and Manipulation)?

OpenFOAM is a powerful tool that allows the numerical solution of differential equations (usually found in continuum mechanics).



What is OpenFOAM (open Field Operation and Manipulation)?

OpenFOAM is a powerful tool that allows the numerical solution of differential equations (usually found in continuum mechanics).

Scientists

OpenFOAM

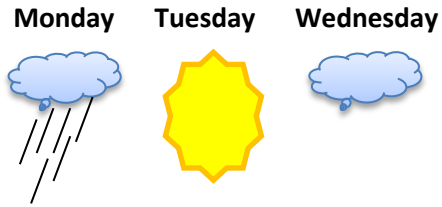
Physical phenomenon

Mathematical Model

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \nabla \cdot (\mathbf{uu}) = -\nabla p + \nabla \cdot \boldsymbol{\tau}$$

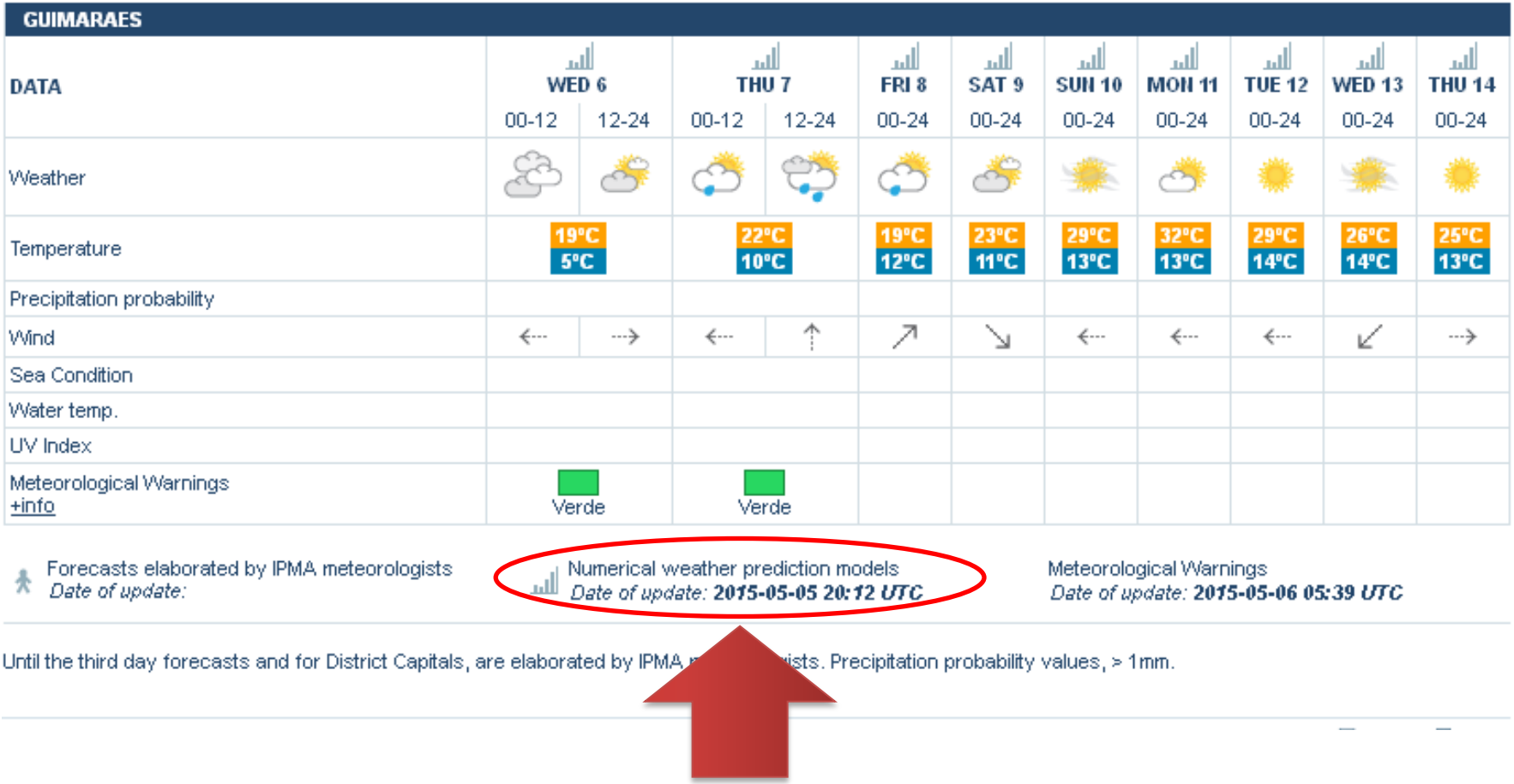
Physical phenomena described by a mathematical model (differential equation).

Numerical Solution



Prediction of the evolution of physical phenomena by numerical solution of the differential equations.

https://www.ipma.pt/



GUIMARAES

DATA	WED 6		THU 7		FRI 8	SAT 9	SUN 10	MON 11	TUE 12	WED 13	THU 14
	00-12	12-24	00-12	12-24	00-24	00-24	00-24	00-24	00-24	00-24	00-24
Weather											
Temperature	19°C 5°C		22°C 10°C		19°C 12°C	23°C 11°C	29°C 13°C	32°C 13°C	29°C 14°C	26°C 14°C	25°C 13°C
Precipitation probability											
Wind	←	→	←	↑	↗	↘	←	←	←	↙	→
Sea Condition											
Water temp.											
UV Index											
Meteorological Warnings +info	Verde		Verde								

Forecasts elaborated by IPMA meteorologists
Date of update:

Numerical weather prediction models
Date of update: **2015-05-05 20:12 UTC**

Meteorological Warnings
Date of update: **2015-05-06 05:39 UTC**

Until the third day forecasts and for District Capitals, are elaborated by IPMA [illegible]ists. Precipitation probability values, > 1mm.

What is OpenFOAM (open Field Operation and Manipulation)?

What is OpenFOAM (open Field Operation and Manipulation)?

- **OpenFOAM** is an open source C++ library, used primarily to create executables, known as applications. The applications fall into two categories: solvers, that are each designed to solve a specific problem in continuum mechanics; and utilities, that are designed to perform tasks that involve data manipulation.

What is OpenFOAM (open Field Operation and Manipulation)?

- **OpenFOAM** is an open source C++ library, used primarily to create executables, known as applications. The applications fall into two categories: solvers, that are each designed to solve a specific problem in continuum mechanics; and utilities, that are designed to perform tasks that involve data manipulation.

- New **solvers** and **utilities** can be created by its users with some pre-requisite knowledge of the underlying method, physics and programming techniques involved.

What is OpenFOAM (open Field Operation and Manipulation)?

What is OpenFOAM (open Field Operation and Manipulation)?

- The discretization of the equation is based on the Finite Volume Method;

What is OpenFOAM (open Field Operation and Manipulation)?

- The discretization of the equation is based on the Finite Volume Method;
- Collocated polyhedral unstructured meshes;

What is OpenFOAM (open Field Operation and Manipulation)?

- The discretization of the equation is based on the Finite Volume Method;
- Collocated polyhedral unstructured meshes;
- It uses second order schemes for the approximation of the different operators, but, many schemes are available, including high order schemes;

What is OpenFOAM (open Field Operation and Manipulation)?

- The discretization of the equation is based on the Finite Volume Method;
- Collocated polyhedral unstructured meshes;
- It uses second order schemes for the approximation of the different operators, but, many schemes are available, including high order schemes;
- Parallel computation are easy to perform (reduction of the computational time);

What is OpenFOAM (open Field Operation and Manipulation)?

- The discretization of the equation is based on the Finite Volume Method;
- Collocated polyhedral unstructured meshes;
- It uses second order schemes for the approximation of the different operators, but, many schemes are available, including high order schemes;
- Parallel computation are easy to perform (reduction of the computational time);
- You can create simple meshes with the mesh generator that comes with OpenFOAM. Also, you can convert to the OpenFOAM format, meshes created with another software (check the user guide for the available formats);

What is OpenFOAM (open Field Operation and Manipulation)?

- The discretization of the equation is based on the Finite Volume Method;
- Collocated polyhedral unstructured meshes;
- It uses second order schemes for the approximation of the different operators, but, many schemes are available, including high order schemes;
- Parallel computation are easy to perform (reduction of the computational time);
- You can create simple meshes with the mesh generator that comes with OpenFOAM. Also, you can convert to the OpenFOAM format, meshes created with another software (check the user guide for the available formats);
- It comes with several utilities;

What is OpenFOAM (open Field Operation and Manipulation)?

- The discretization of the equation is based on the Finite Volume Method;
- Collocated polyhedral unstructured meshes;
- It uses second order schemes for the approximation of the different operators, but, many schemes are available, including high order schemes;
- Parallel computation are easy to perform (reduction of the computational time);
- You can create simple meshes with the mesh generator that comes with OpenFOAM. Also, you can convert to the OpenFOAM format, meshes created with another software (check the user guide for the available formats);
- It comes with several utilities;
- All components implemented in library form for easy re-use.

What is OpenFOAM (open Field Operation and Manipulation)?

‘Basic’ CFD codes

laplacianFoam Solves a simple Laplace equation, e.g. for thermal diffusion in a solid;

potentialFoam Simple potential flow solver which can be used to generate starting fields for full Navier-Stokes codes;

scalarTransportFoam Solves a transport equation for a passive scalar

What is OpenFOAM (open Field Operation and Manipulation)?

Incompressible flow **DNS and LES**

What is OpenFOAM (open Field Operation and Manipulation)?

Incompressible flow

DNS and LES

Compressible flow

Particle-tracking flows

What is OpenFOAM (open Field Operation and Manipulation)?

Incompressible flow	**DNS and LES**
Compressible flow	**Particle-tracking flows**
Multiphase flow	**Heat transfer**

What is OpenFOAM (open Field Operation and Manipulation)?

Incompressible flow	**DNS and LES**
Compressible flow	**Particle-tracking flows**
Multiphase flow	**Heat transfer**
Combustion	**…**

What is OpenFOAM (open Field Operation and Manipulation)?

- Users can change the existing solvers, and, use them as the start point for the creation of a new solver.
- The complete source of the code is available!
- Users can write a complex solver with only few lines.

Simulations with OpenFOAM

https://www.youtube.com/watch?v=1PHvJMy45rE&list=PL6BfJWHH53_pLUOzU8uuzXgvrhR6uBAR6
Multiphase flow based on OpenFOAM's volume of fluid method, simulation performed by DHCAE Tools GmbH.
For more information, please contact info@dhcae-tools.de

https://www.youtube.com/watch?v=SCQL57wHVq4&index=4&list=PL6BfJWHH53_pNtHuG0JAEGsW7WzteQbsL
CFD simulation of a supersonic jet occuring in a direct injection **combustion engine**. Computation has been done with OpenFoam at GDTech in Belgium. LES turbulence model has been used.

https://www.youtube.com/watch?v=D6iuVr9V6os
2D CFD simulation of a bullet at Mach number 1.6 done with OpenFoam. GDTech – Belgium.

https://www.youtube.com/watch?v=Q6XyTrhaGxc&list=PL6BfJWHH53_pNtHuG0JAEGsW7WzteQbsL&index=6
Multiphase Flow Simulation of Vertical Slot Fishway performed with HELYX®

https://www.youtube.com/watch?v=OH0l5CkY_aY
CFD analysis of a planing hull in OpenFOAM. TotalSim.

https://www.youtube.com/watch?v=Ky7Ksdf9ihs
Floating wind turbine platform

https://www.youtube.com/watch?v=UiuX7IwJ3tA&index=31&list=PL6BfJWHH53_pq7o-VCQvBdyDu3yUq7YoT
Flow on the Sharp-Crested Rectangular Weir

https://www.youtube.com/watch?v=FhkGIa8J9b0
Tanker aerodynamics - MkI

Equation mimicking in OpenFOAM

The equation,

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot \left(\phi \mathbf{u} \right) - \nabla \cdot \mu \nabla \mathbf{u} = -\nabla p$$

can be written in **OpenFOAM** as follows,

Equation mimicking in OpenFOAM

The equation,

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot \left(\phi \mathbf{u} \right) - \nabla \cdot \mu \nabla \mathbf{u} = -\nabla p$$

can be written in **OpenFOAM** as follows,

```
solve
(
  fvm::ddt(rho,U) + fvm::div(phi,U) - fvm::laplacian(mu,U) == - fvc::grad(p)
);
```

Different versions of OpenFOAM

Different versions of OpenFOAM

Forks

- Caelus-CML
- ENGYS' own builds of OpenFOAM
- foam-extend
- FreeFOAM
- iconCFD
- RapidCFD

Variants

- Windows
- Mac

http://openfoamwiki.net/index.php/**Forks_and_Variants**

Different versions of OpenFOAM

Different versions of OpenFOAM

The OpenFOAM Foundation

http://www.openfoam.org/

OpenFOAM Foundation releases OpenFOAM® 2.3.1
10th December 2014

Old Versions of OpenFOAM
Version 2.3.0 download
Version 2.2.2 download
Version 2.2.1 download
Version 2.2.0 download
.
.
.

Different versions of OpenFOAM

The OpenFOAM Foundation

http://www.openfoam.org/

OpenFOAM Foundation releases OpenFOAM® 2.3.1
10th December 2014

Old Versions of OpenFOAM
Version 2.3.0 download
Version 2.2.2 download
Version 2.2.1 download
Version 2.2.0 download
.
.
.

The OpenFOAM® Extend Project

http://www.extend-project.de/

foam-extend-3.1 "Zagreb" is released The Extend Project is proud to announce the next release of the community fork of OpenFOAM®: foam-extend-3.1

Different versions of OpenFOAM

https://openfoamwiki.net/index.php/Installation

2 Installing OpenFOAM

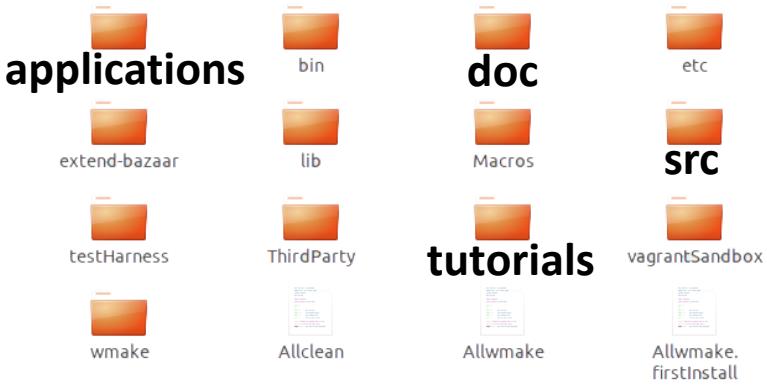
This is the tree of the categories and respective sub-categories about installing OpenFOAM: [+] Installing OpenFOAM on Linux
[+] Installing OpenFOAM on Mac OS
[+] Installing OpenFOAM on Windows
Installation
Installation/Linux
Installation/Mac OS
Installation/Windows

Is this Wiki affiliated with OpenCFD/ESI ?

No. This Wiki is a community effort. Its aim is to give people a place where they can collect/publish information about OpenFOAM and related projects.

Structure of OpenFOAM

Structure of OpenFOAM



The **OpenFOAM** source code comprises of four main components:

src: the core OpenFOAM source code;

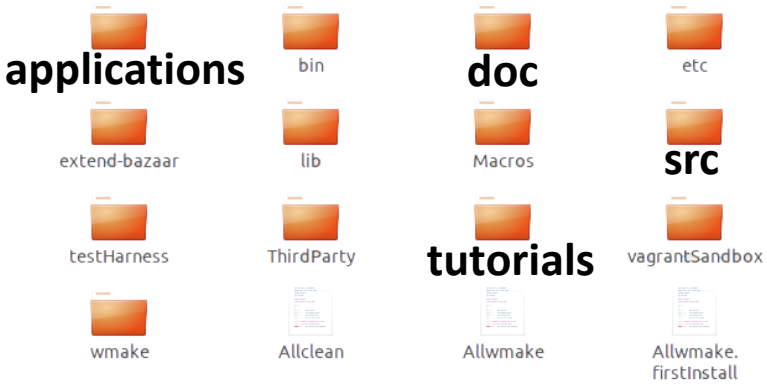
applications: collections of library functionality wrapped up into applications, such as solvers and utilities;

tutorials: a suite of test cases that highlight a broad cross-section of OpenFOAM's capabilities;

doc: supporting documentation.

Structure of OpenFOAM

```
»cd $FOAM_INST_DIR
```



The **OpenFOAM** source code comprises of four main components:

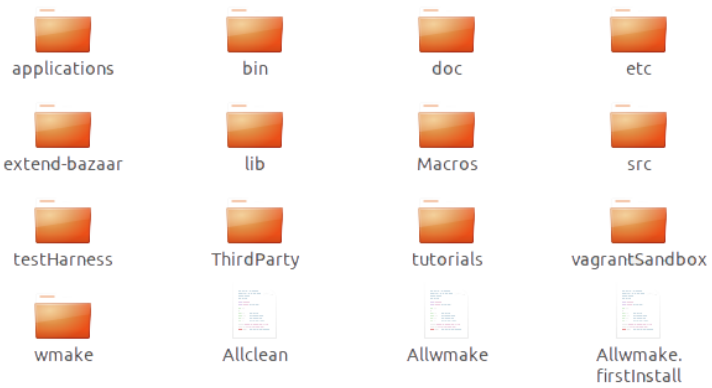
src: the core OpenFOAM source code;

applications: collections of library functionality wrapped up into applications, such as solvers and utilities;

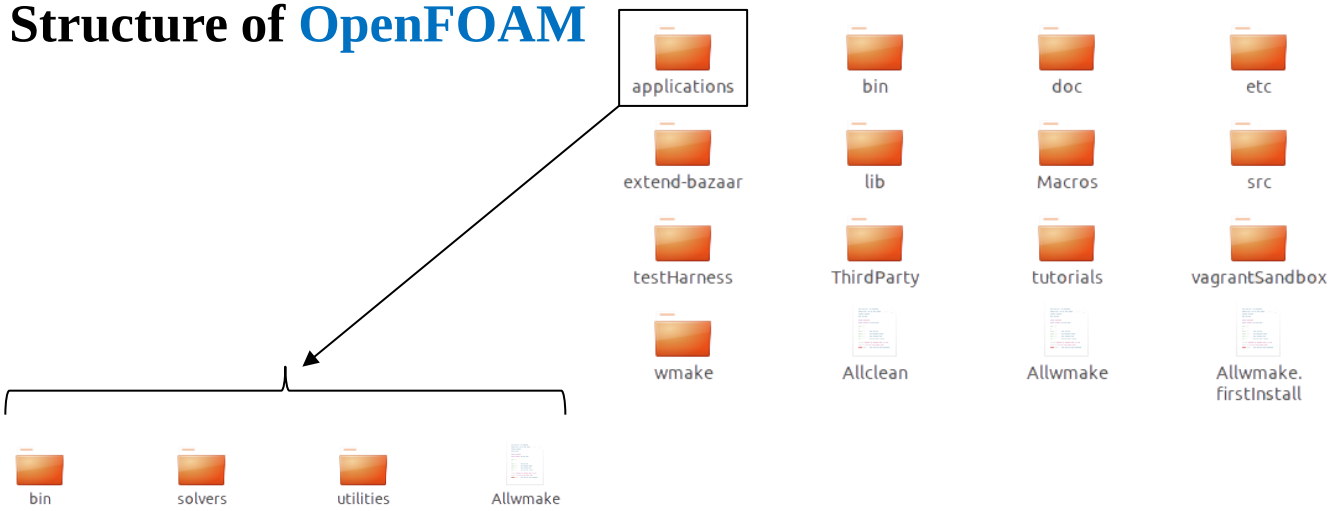
tutorials: a suite of test cases that highlight a broad cross-section of OpenFOAM's capabilities;

doc: supporting documentation.

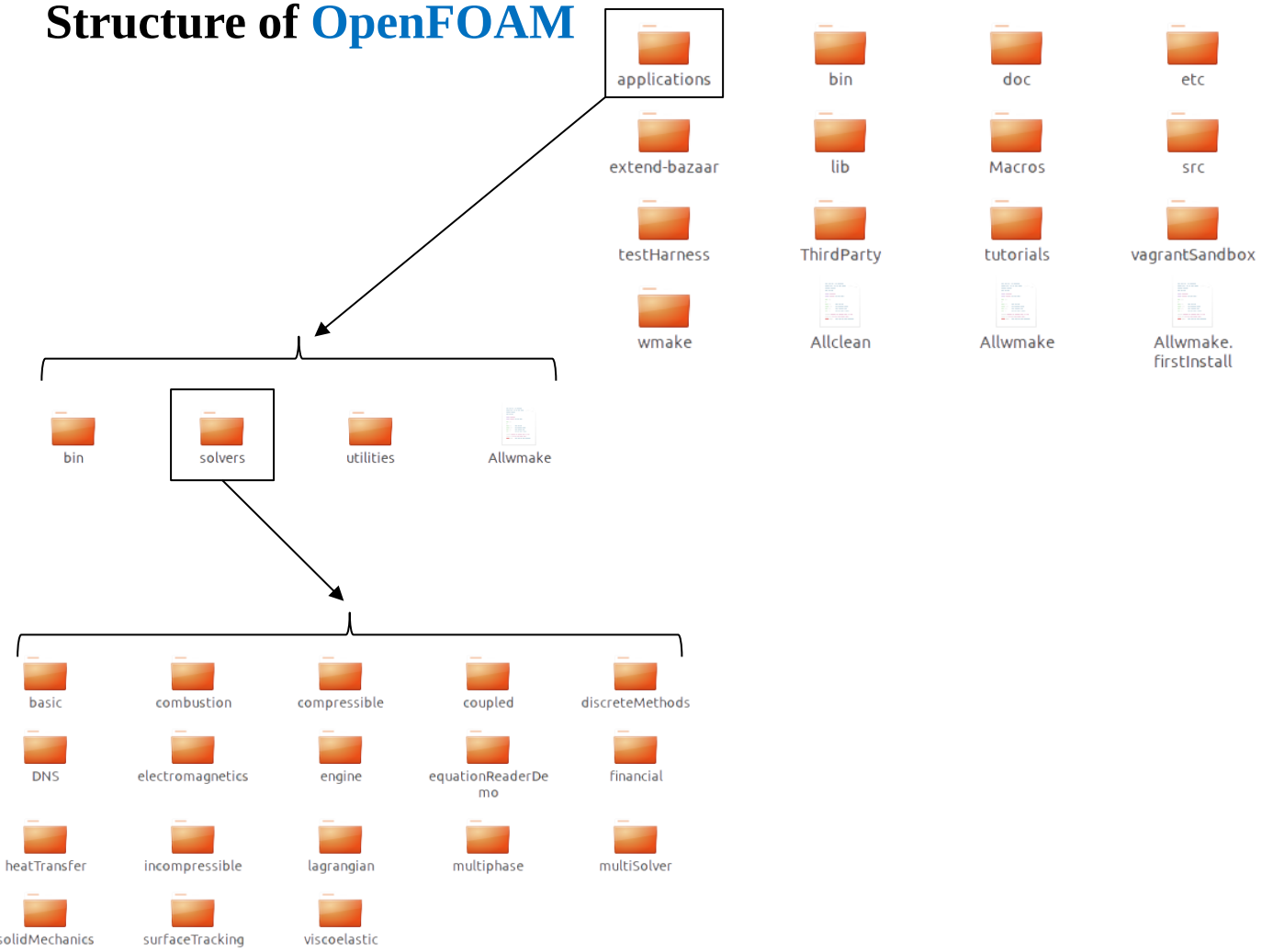
Structure of OpenFOAM



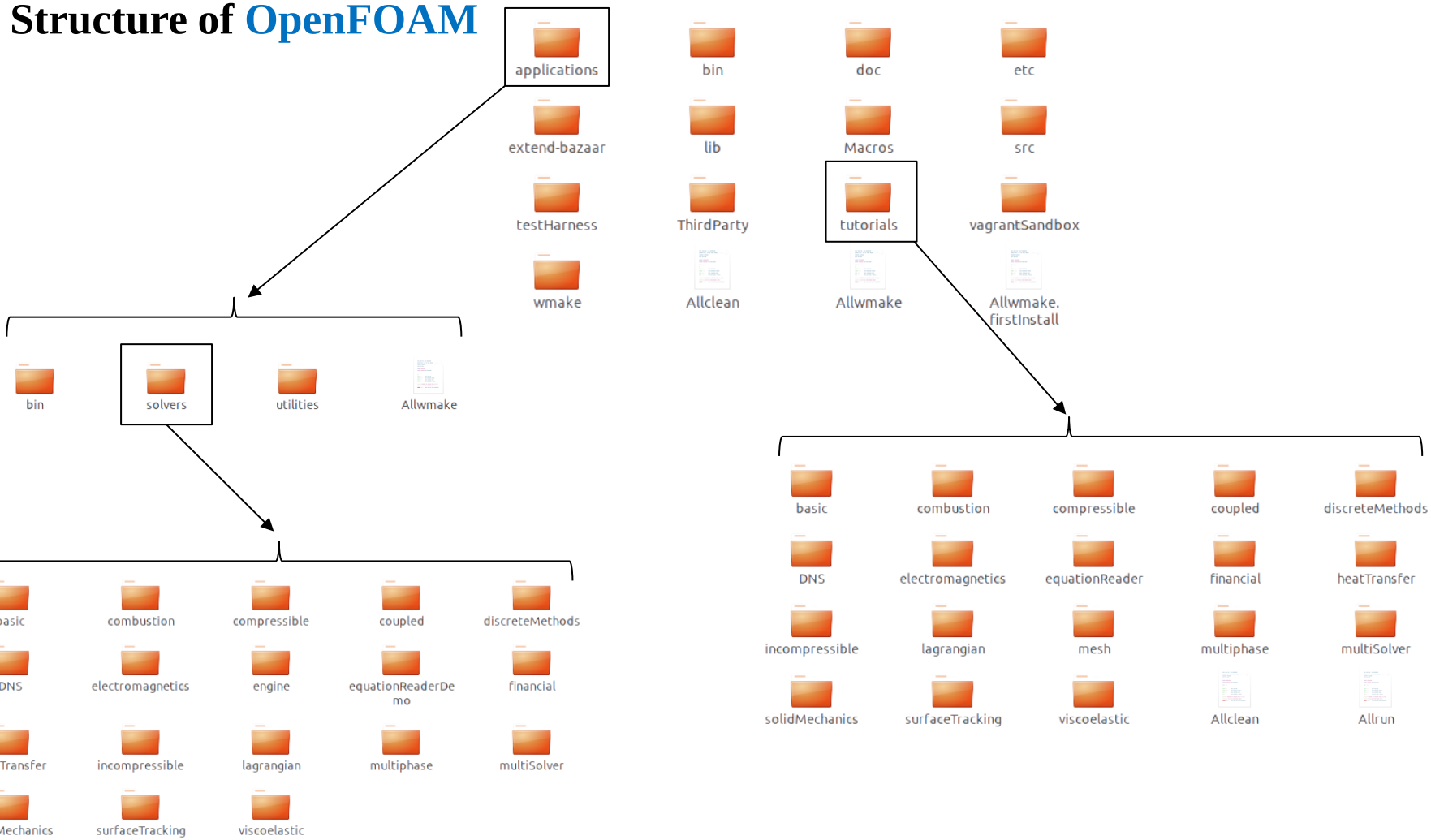
Structure of OpenFOAM



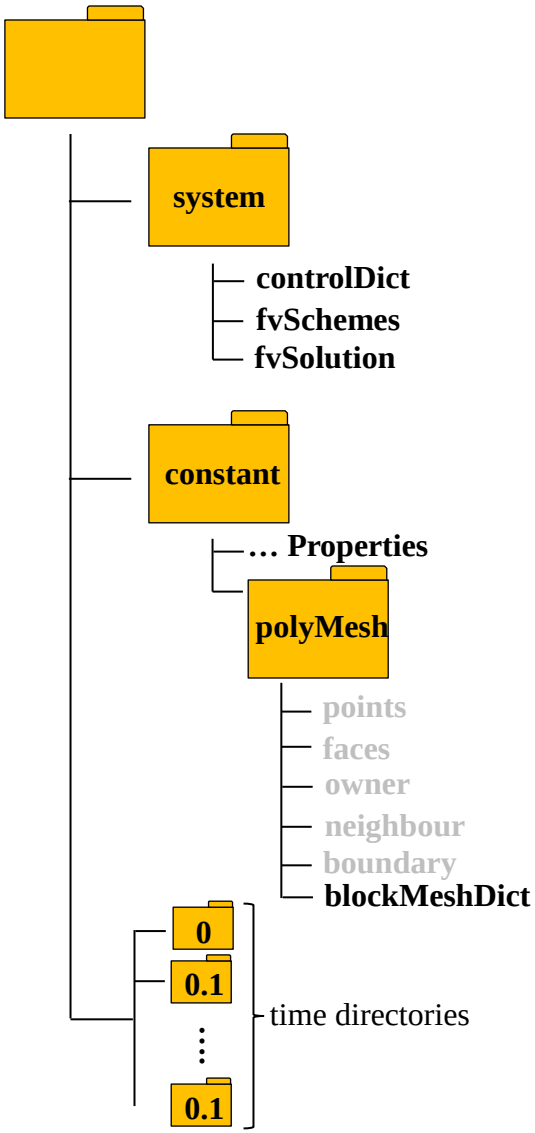
Structure of OpenFOAM



Structure of OpenFOAM



Case directory in OpenFOAM



Solvers?

fvSchemes?

polyMesh?

⋮

Q: How to understand and get acquainted with all these concepts?

Solvers?

fvSchemes?

polyMesh?

⋮

Q: How to understand and get acquainted with all these concepts?

A: First lets take a look at a really simple example “lid driven cavity”

Solvers?

fvSchemes?

polyMesh?

⋮

Q: How to understand and get acquainted with all these concepts?

A: First lets take a look at a really simple example “lid driven cavity”

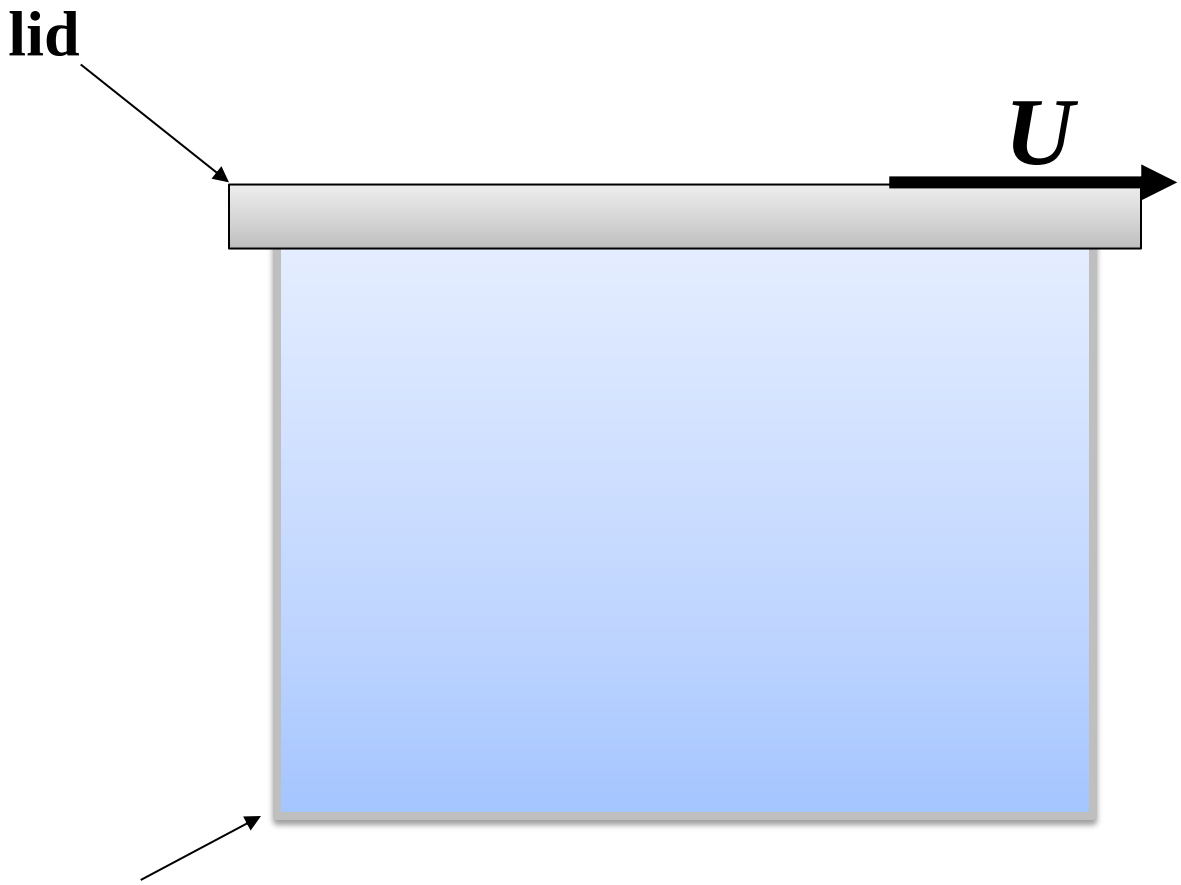
B: Lets work a problem by ourselves “1D problem – heat transfer”

Also…

http://www.openfoam.com/
http://openfoamwiki.net/index.php/Main_Page
http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2009/
http://sourceforge.net/projects/openfoam-extend/
http://www.openfoamworkshop.org/

OpenFOAM user guide!

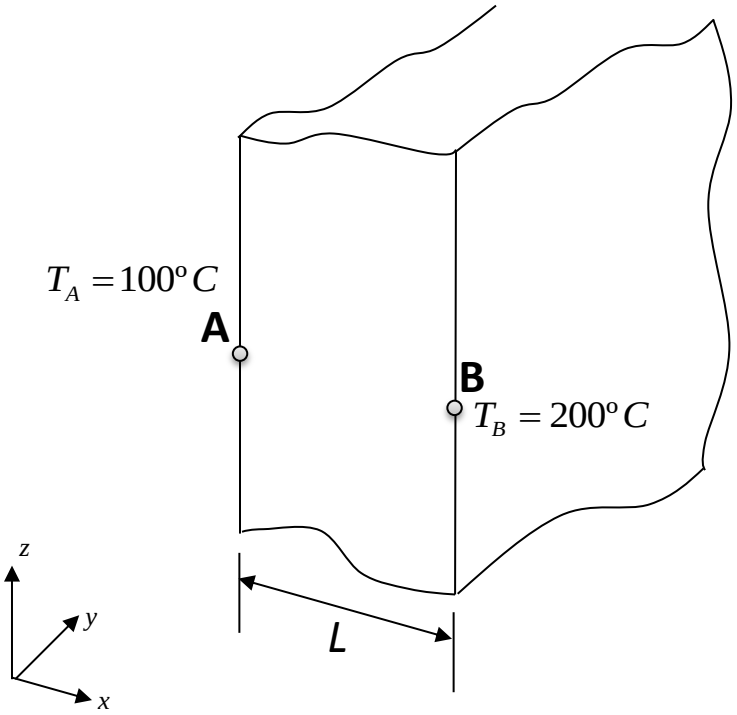
Lid driven cavity


lid
U
Container with fluid

- blockMesh
- checkMesh
- icoFoam
- paraFoam

Simple 1D Problem – heat transfer

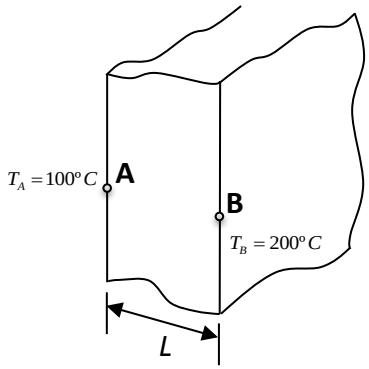
Heat conduction in a large plate, thickness ***L*=2 cm**, thermal conductivity ***k*= 0.5 W/mK**, uniform internal heat generation ***q*=1000 W/m³**. Faces A and B are kept at a constant temperature.



Heat conduction in a large plate, thickness L=2 cm, thermal conductivity *k= 0.5 W/mK*, uniform internal heat generation *q=1E6 W/m^3*. Faces A and B are kept at a constant temperature.

Heat conduction in a large plate, thickness L=2 cm, thermal conductivity *k= 0.5 W/mK*, uniform internal heat generation *q=1E6 W/m^3*. Faces A and B are kept at a constant temperature.

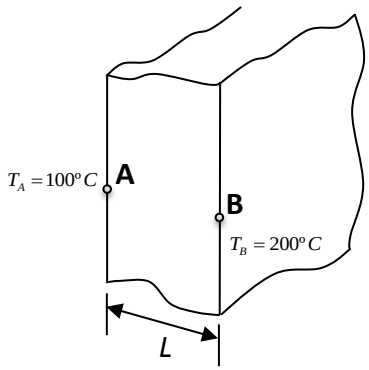
Physical phenomena



Heat conduction in a large plate, thickness L=2 cm, thermal conductivity *k= 0.5 W/mK*, uniform internal heat generation *q=1E6 W/m^3*. Faces A and B are kept at a constant temperature.

Scientists

Physical phenomena



Mathematical Model

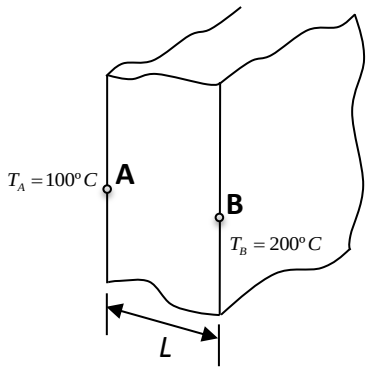
$$\frac{d}{dx}\left(k\frac{dT}{dx}\right)+q=0$$

Heat conduction in a large plate, thickness L=2 cm, thermal conductivity *k= 0.5 W/mK*, uniform internal heat generation *q=1E6 W/m^3*. Faces A and B are kept at a constant temperature.

Scientists

OpenFOAM

Physical phenomena



Mathematical Model

$$\frac{d}{dx}\left(k\frac{dT}{dx}\right)+q=0$$

Numerical Solution

$T = ??!!!$

Theory:

Analytical solution:

$$T = \left[\frac{T_B - T_A}{L} + \frac{q}{2k}(L - x) \right] x + T_A$$

For this particular case we know the exact solution, meaning that, we know how temperature varies from A to B.

Theory:

Analytical solution:

$$T = \left[\frac{T_B - T_A}{L} + \frac{q}{2k}(L - x) \right] x + T_A$$

For this particular case we know the exact solution, meaning that, we know how temperature varies from A to B.

↓

OpenFOAM does not know that!!!

Theory:

Analytical solution:

$$T = \left[\frac{T_B - T_A}{L} + \frac{q}{2k}(L - x) \right] x + T_A$$

For this particular case we know the exact solution, meaning that, we know how temperature varies from A to B.

↓

OpenFOAM does not know that!!!

$$\frac{d}{dx}\left(k \frac{dT}{dx} \right) + q = 0 \longrightarrow$$

So, I want the numerical solution for this equation!
How does **OpenFOAM** operates?!

Theory:

$$\int \frac{d}{dx}\left(k\frac{dT}{dx}\right)dx + \int q\,dx = 0$$

OpenFOAM uses the **integral version** of governing equations**.**

Then, **Gauss theorem** is used to reduce the domain of integration by one dimension

Theory:

$$\int \frac{d}{dx}\left(k\frac{dT}{dx}\right)dx + \int q\,dx = 0$$

OpenFOAM uses the **integral version** of governing equations**.**

Then, **Gauss theorem** is used to reduce the domain of integration by one dimension

I do not remember **Gauss theorem** …

Well, in 1D, **Gauss theorem** is equivalent to the **Fundamental theorem of Calculus**

$$\int_a^b F'(x)\,dx = F(b) - F(a)$$

Theory:

$$\int_a^b \frac{d}{dx}\left(k\frac{dT}{dx}\right)dx + \int_a^b q\,dx = 0$$

Fundamental theorem of Calculus (FTC)

$$\int_a^b F'(x)\,dx = F(b) - F(a)$$

Theory:

$$\underbrace{\int_a^b \frac{d}{dx}\left(k\frac{dT}{dx}\right)dx}_{\text{(FTC)}} + \int_a^b q\,dx = 0$$

$$\left(k\frac{dT}{dx}\right)_b - \left(k\frac{dT}{dx}\right)_a$$

Fundamental theorem of Calculus (FTC)

$$\int_a^b F'(x)\,dx = F(b) - F(a)$$

Theory:

Fundamental theorem of Calculus (FTC)

$$\int_a^b F'(x)\,dx = F(b) - F(a)$$

$$\underbrace{\int_a^b \frac{d}{dx}\left(k\frac{dT}{dx}\right)dx}_{\text{(FTC)}} + \underbrace{\int_a^b q\,dx}_{q\int_a^b 1\,dx} = 0$$

$$\left(k\frac{dT}{dx}\right)_b - \left(k\frac{dT}{dx}\right)_a$$

$$q(b-a)$$

Theory:

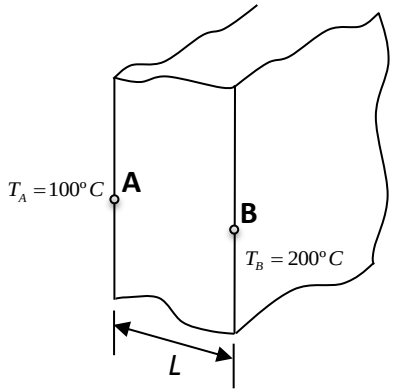
Fundamental theorem of Calculus (FTC)

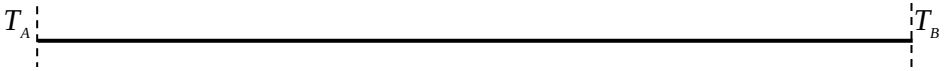
$$\int_a^b F'(x)\,dx = F(b) - F(a)$$

$$\underbrace{\int_a^b \frac{d}{dx}\left(k\frac{dT}{dx}\right)dx}_{\text{(FTC)}} + \underbrace{\int_a^b q\,dx}_{q\int_a^b 1\,dx} = 0$$

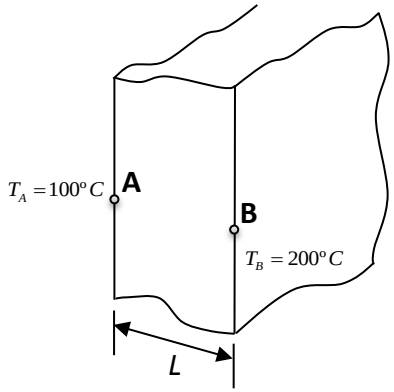
$$\left(k\frac{dT}{dx}\right)_b - \left(k\frac{dT}{dx}\right)_a + q(b-a) = 0$$

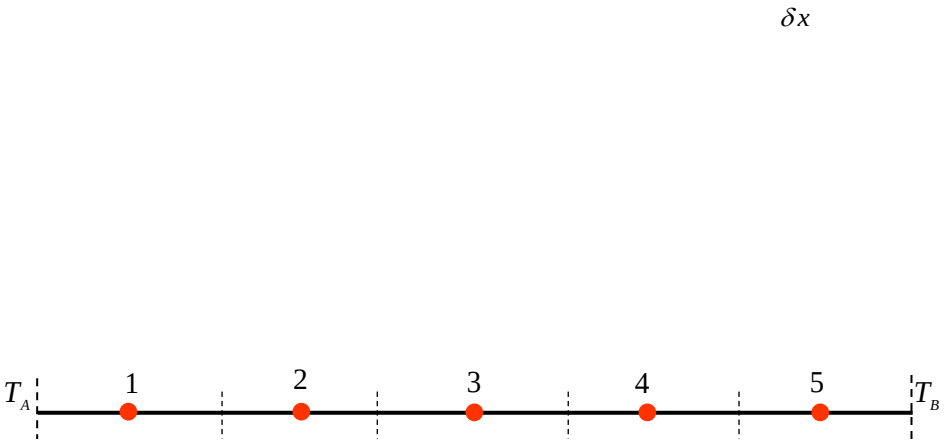
Theory: creating the mesh



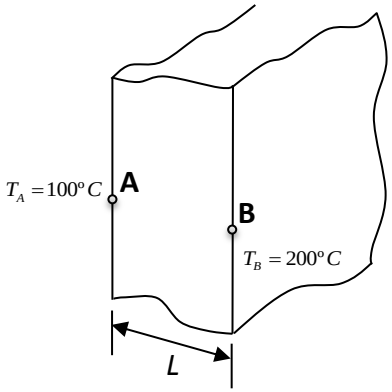


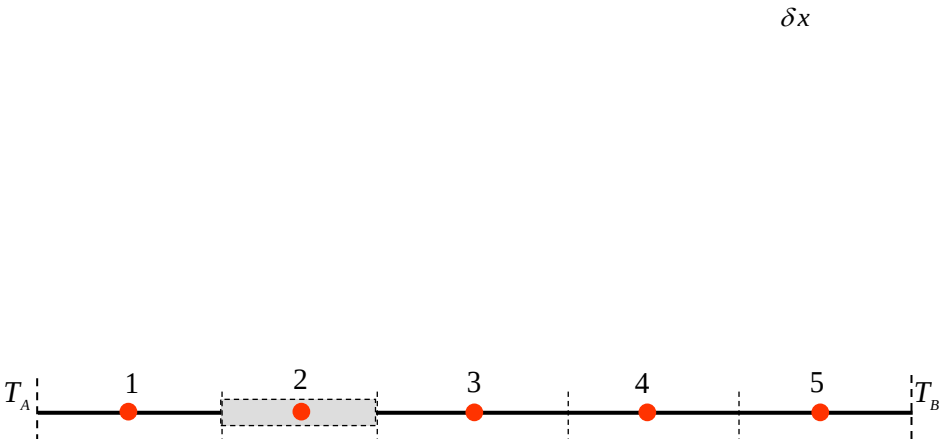
Theory: creating the mesh



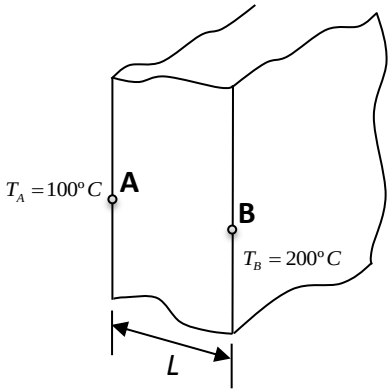


Theory: creating the mesh

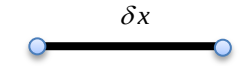


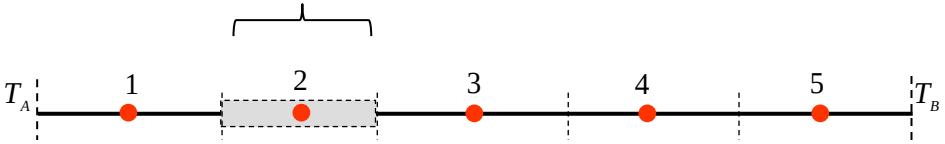


Theory: creating the mesh

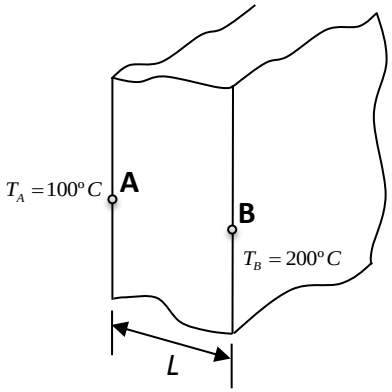


$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$

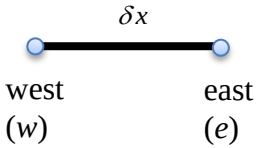


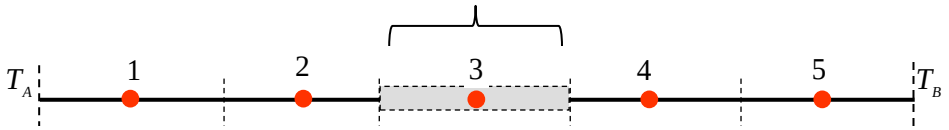


Theory: creating the mesh

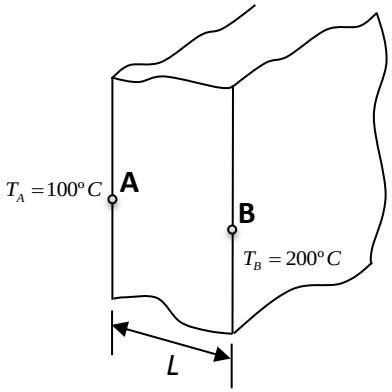


$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$

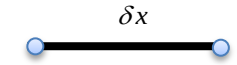


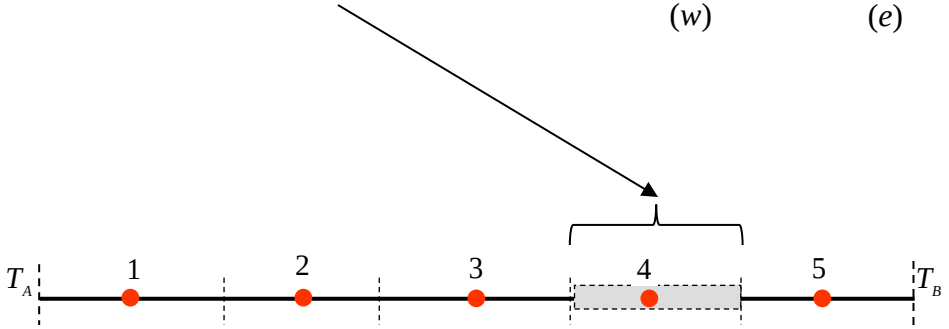


Theory: creating the mesh

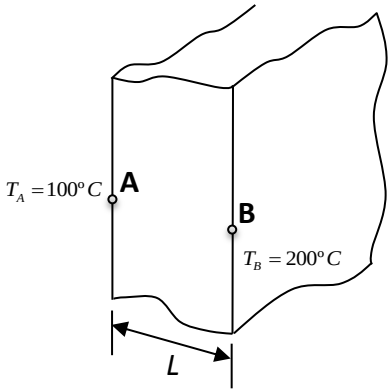


$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$

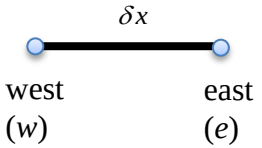


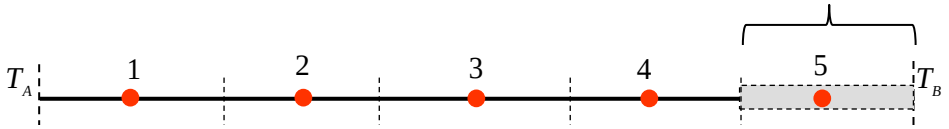


Theory: creating the mesh

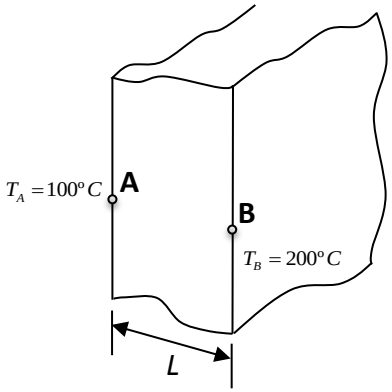


$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$

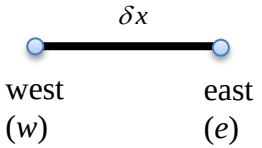


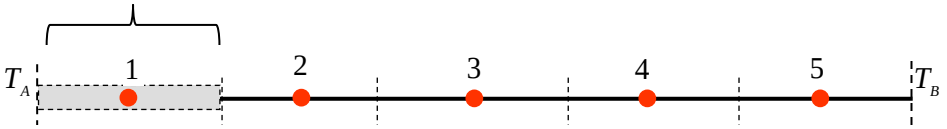


Theory: creating the mesh

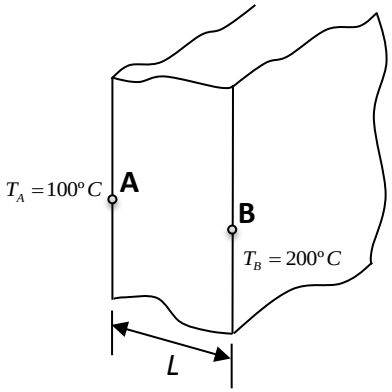


$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$

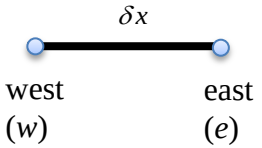


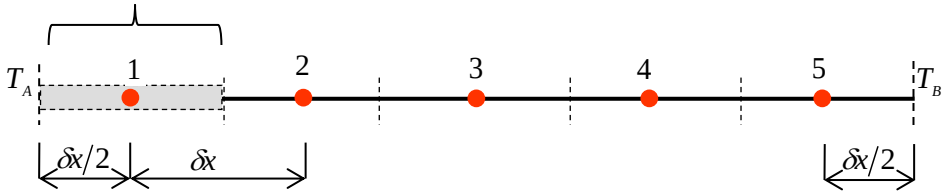


Theory: creating the mesh

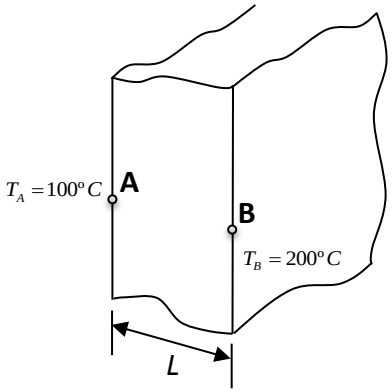


$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$

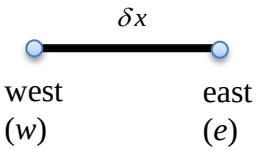


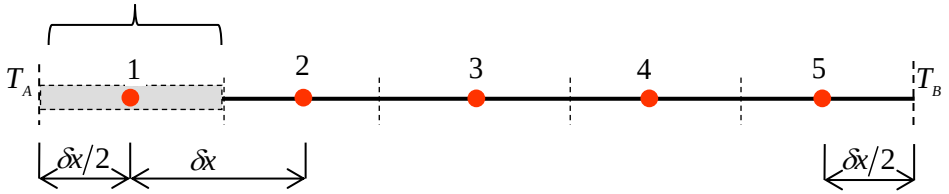


Theory: creating the mesh



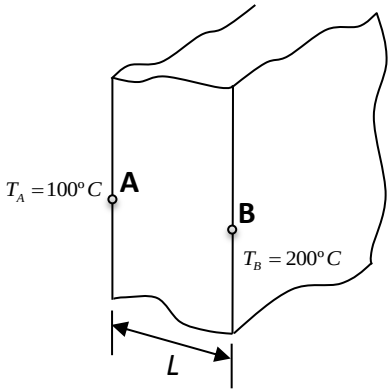
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$



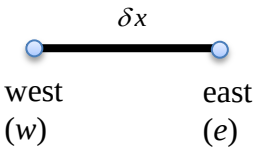


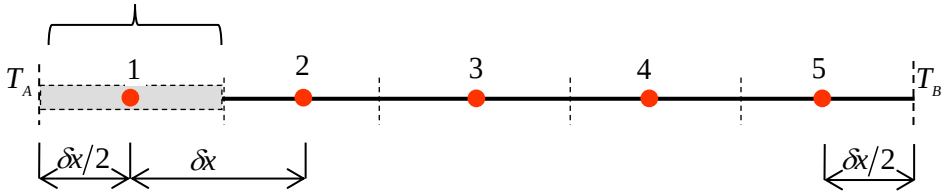
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell1}$$

Theory: creating the mesh



$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$

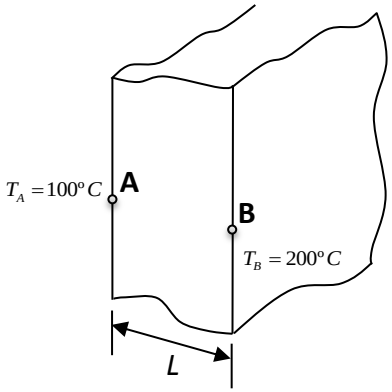




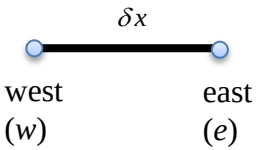
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1$$

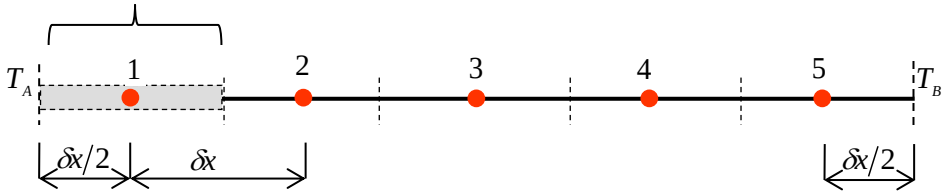
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2$$

Theory: creating the mesh



$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$



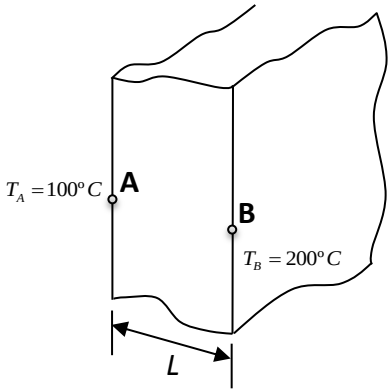


$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1$$

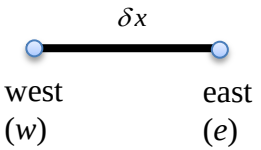
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2$$

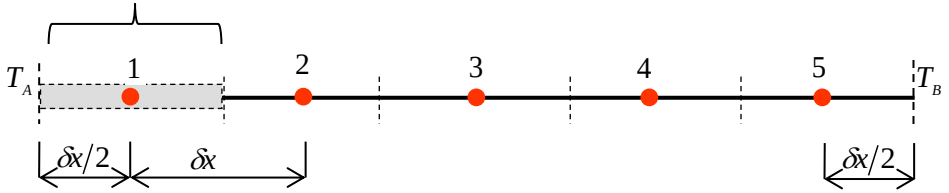
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}3$$

Theory: creating the mesh



$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$





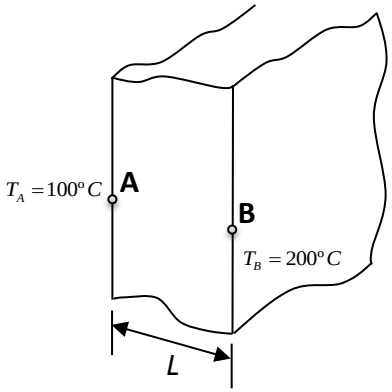
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1$$

$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2$$

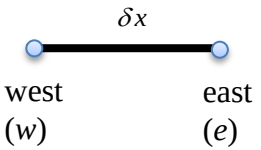
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}3$$

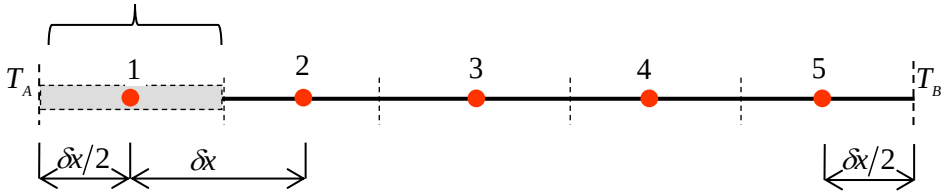
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}4$$

Theory: creating the mesh



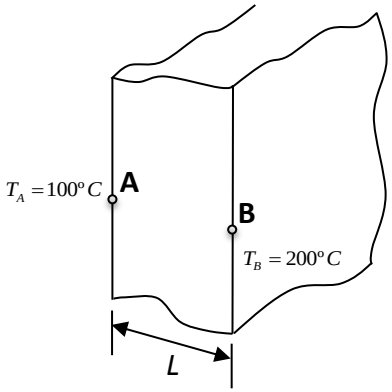
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$



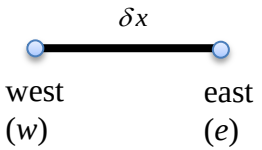


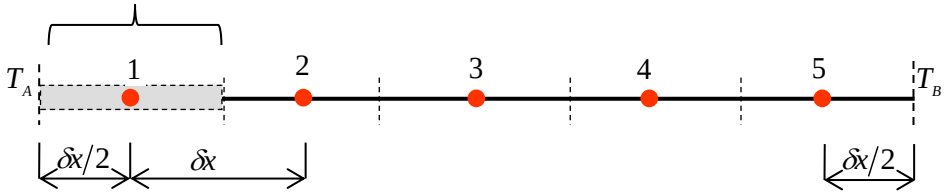
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1$$
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2$$
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}3$$
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}4$$
$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}5$$

Theory: creating the mesh



$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0$$



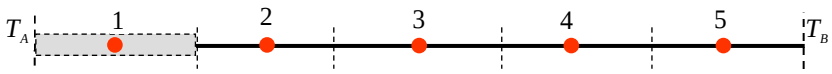


System of 5 equations

$$\begin{cases}
\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1 \\
\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2 \\
\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}3 \\
\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}4 \\
\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}5
\end{cases}$$

System of 5 equations

$$\begin{cases} \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}3 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}4 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}5 \end{cases}$$

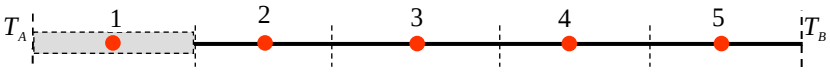


How to approximate the derivatives?

How to define the mesh in OpenFOAM?

System of 5 equations

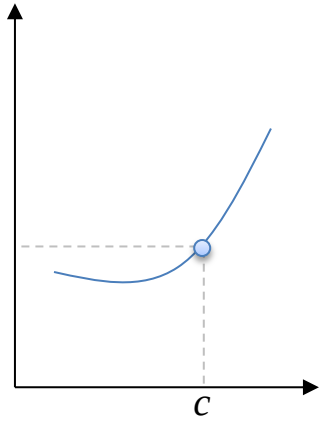
$$\begin{cases} \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}3 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}4 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}5 \end{cases}$$



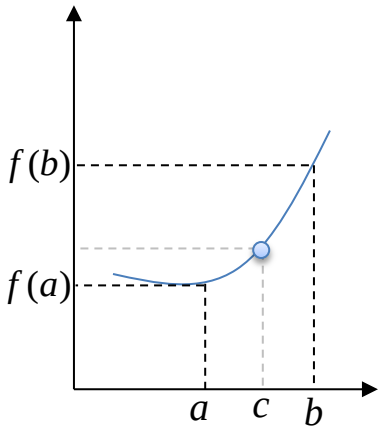
How to approximate the derivatives?

How to define the mesh in OpenFOAM?

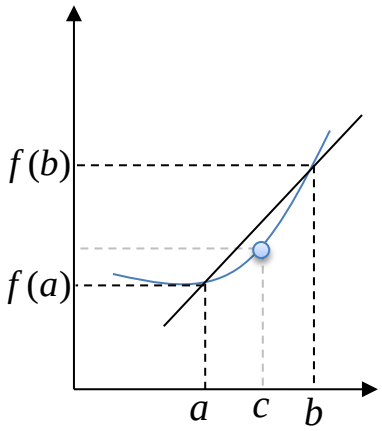
How to approximate the derivatives?



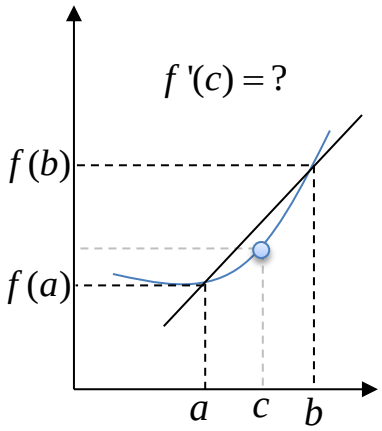
How to approximate the derivatives?



How to approximate the derivatives?

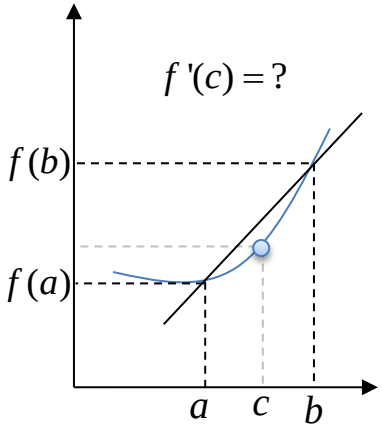


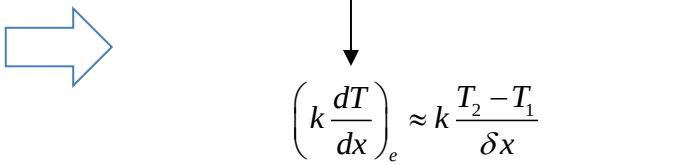
How to approximate the derivatives?



$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

How to approximate the derivatives?

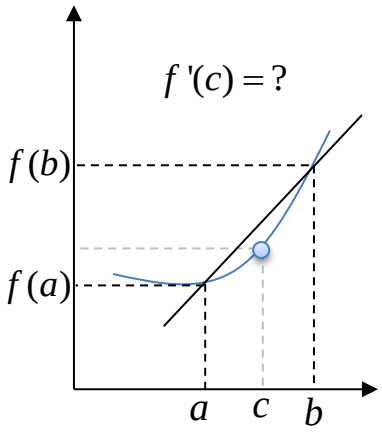




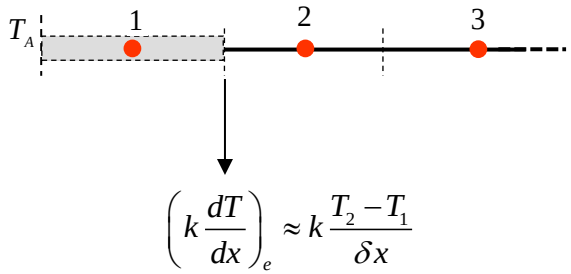
$$\left(k\frac{dT}{dx}\right)_e \approx k\frac{T_2 - T_1}{\delta x}$$

$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

How to approximate the derivatives?



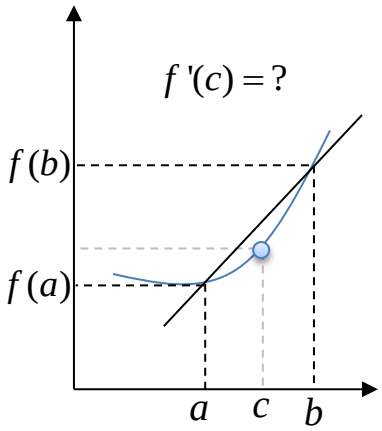
$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$



System of linear equations

$$\begin{cases} k\dfrac{T_2 - T_1}{\delta x} - k\dfrac{T_1 - T_A}{\delta x/2} + q\delta x = 0 \quad \text{cell}1 \\ k\dfrac{T_3 - T_2}{\delta x} - k\dfrac{T_2 - T_1}{\delta x} + q\delta x = 0 \quad \text{cell}2 \\ k\dfrac{T_4 - T_3}{\delta x} - k\dfrac{T_3 - T_2}{\delta x} + q\delta x = 0 \quad \text{cell}3 \\ k\dfrac{T_5 - T_4}{\delta x} - k\dfrac{T_4 - T_3}{\delta x} + q\delta x = 0 \quad \text{cell}4 \\ k\dfrac{T_B - T_5}{\delta x/2} - k\dfrac{T_5 - T_4}{\delta x} + q\delta x = 0 \quad \text{cell}5 \end{cases}$$

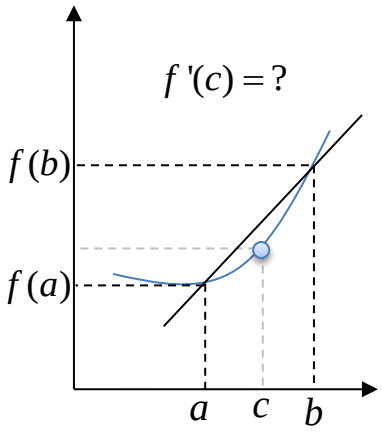
How to approximate the derivatives?



$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

Is this the only possible approximation for the derivative?

How to approximate the derivatives?

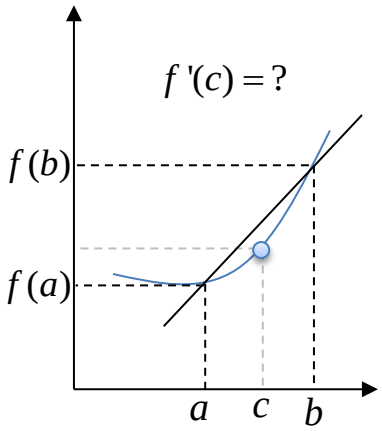


$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

Is this the only possible approximation for the derivative?

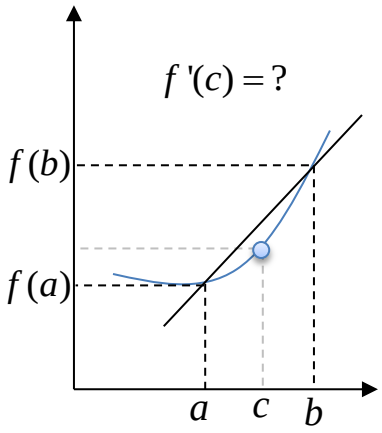
No!!!

How to approximate the derivatives?



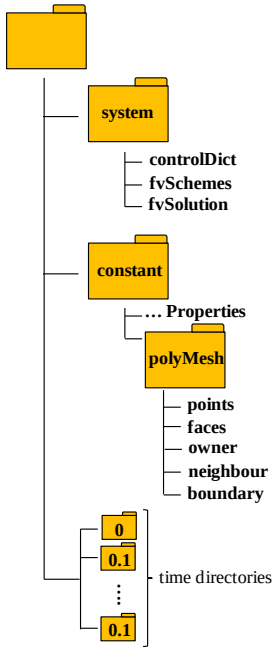
$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

How to approximate the derivatives?

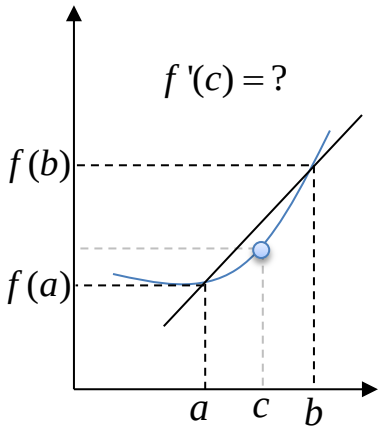


$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

Remember…

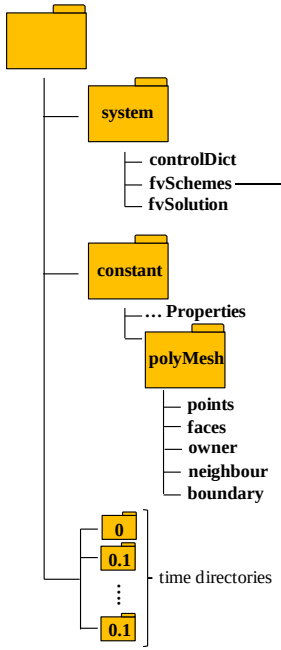


How to approximate the derivatives?



$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

Remember…

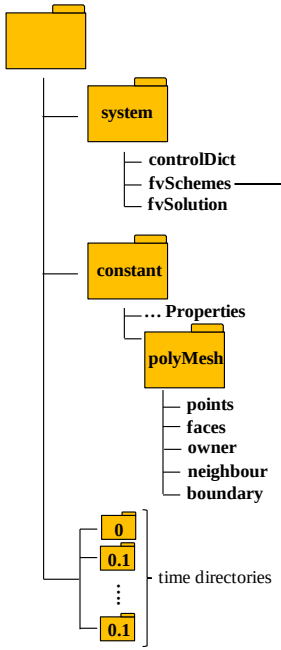


fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      / F ield          | OpenFOAM Extend Project: Open Source CFD        |
|  \\    / O peration       | Version:  1.6-ext                               |
|   \\  / A nd              | Web:      www.extend-project.de                 |
|    \\/  M anipulation     |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}
divSchemes
{
    default             none;
    div(phi,U)          Gauss upwind;
    div(phi,C)          Gauss Gamma01 0.8;
}
laplacianSchemes
{
    default                 none;
    laplacian(etaPEff,U)        Gauss linear corrected;
    laplacian(etaPEff+etaS,U)   Gauss linear corrected;
    laplacian((1|A(U)),p)       Gauss linear corrected;
}
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    p;
}
```

fvSchemes

Remember…

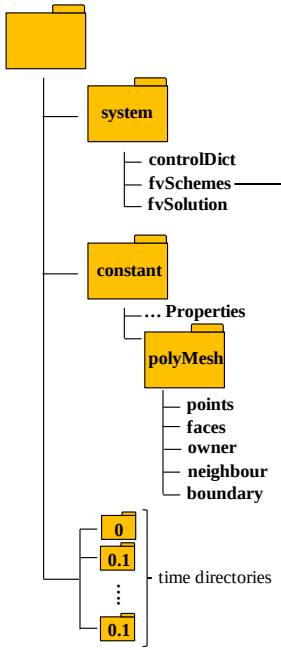



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM Extend Project: Open Source CFD        |
|  \\    /   O peration     | Version:  1.6-ext                               |
|   \\  /    A nd           | Web:      www.extend-project.de                 |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}
divSchemes
{
    default             none;
    div(phi,U)          Gauss upwind;
    div(phi,C)          Gauss Gamma01 0.8;
}
laplacianSchemes
{
    default                 none;
    laplacian(etaPEff,U)        Gauss linear corrected;
    laplacian(etaPEff+etaS,U)   Gauss linear corrected;
    laplacian((1|A(U)),p)       Gauss linear corrected;
}
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    p;
}
```


```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM Extend Project: Open Source CFD        |
|  \\    /   O peration     | Version:  1.6-ext                               |
|   \\  /    A nd           | Web:      www.extend-project.de                 |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

fvSchemes

Remember…

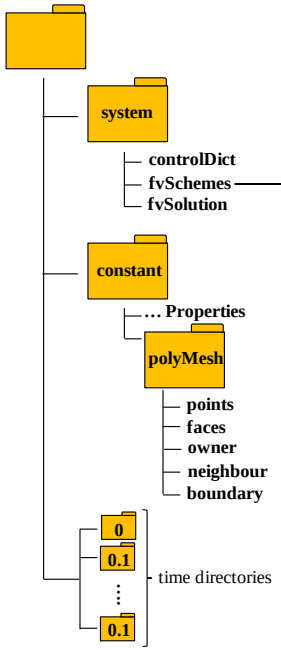



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM Extend Project: Open Source CFD        |
|  \\    /   O peration     | Version:  1.6-ext                               |
|   \\  /    A nd           | Web:      www.extend-project.de                 |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}
divSchemes
{
    default             none;
    div(phi,U)          Gauss upwind;
    div(phi,C)          Gauss Gamma01 0.8;
}
laplacianSchemes
{
    default                 none;
    laplacian(etaPEff,U)        Gauss linear corrected;
    laplacian(etaPEff+etaS,U)   Gauss linear corrected;
    laplacian((1|A(U)),p)       Gauss linear corrected;
}
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    p;
}
```

```
ddtSchemes                                  → ∂/∂t (...)
{
    default         Euler;
}
gradSchemes                                 → ∇p
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}
divSchemes                                  → ∇·(uu)
{
    default             none;
    div(phi,U)          Gauss upwind;
    div(phi,C)          Gauss Gamma01 0.8;
}
laplacianSchemes                            → ∇·(k∇T)
{
    default                 none;
    laplacian(etaPEff,U)        Gauss linear corrected;
    laplacian(etaPEff+etaS,U)   Gauss linear corrected;
    laplacian((1|A(U)),p)       Gauss linear corrected;
```

fvSchemes

Remember…

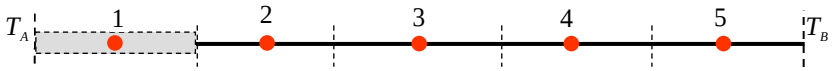



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM Extend Project: Open Source CFD        |
|  \\    /   O peration     | Version:  1.6-ext                               |
|   \\  /    A nd           | Web:      www.extend-project.de                 |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}
divSchemes
{
    default             none;
    div(phi,U)          Gauss upwind;
    div(phi,C)          Gauss Gamma01 0.8;
}
laplacianSchemes
{
    default                 none;
    laplacian(etaPEff,U)        Gauss linear corrected;
    laplacian(etaPEff+etaS,U)   Gauss linear corrected;
    laplacian((1|A(U)),p)       Gauss linear corrected;
}
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    p;
}
```

```
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    p;
}
```

System of 5 equations

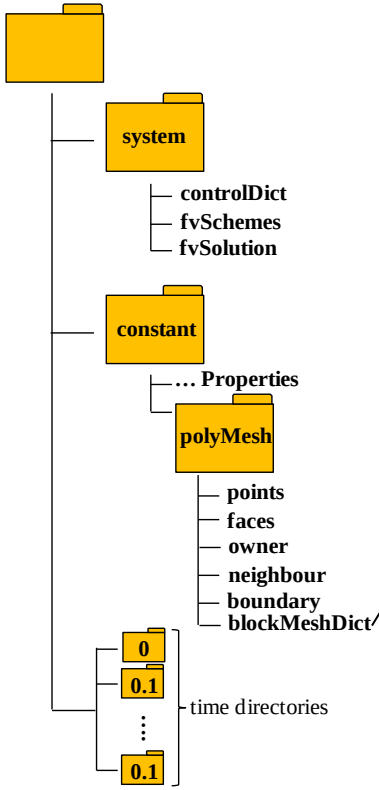
$$\begin{cases} \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}1 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}2 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}3 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}4 \\ \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + q\delta x = 0 \quad \text{cell}5 \end{cases}$$



How to approximate the derivatives?

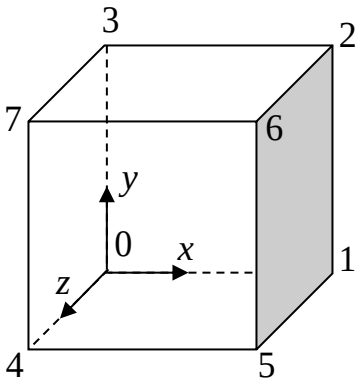
How to define the mesh in OpenFOAM?

How to define the mesh in OpenFOAM?

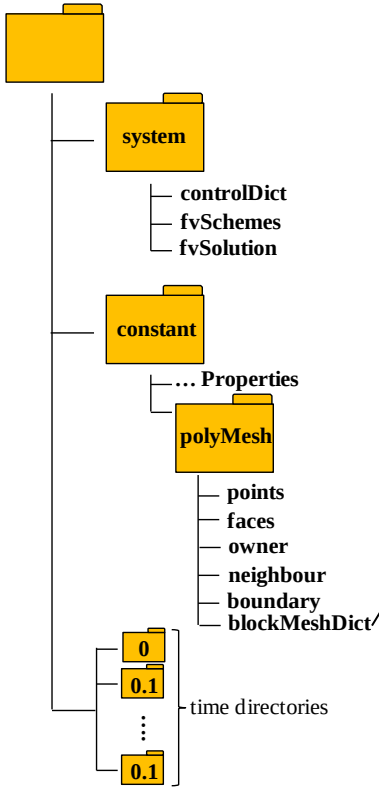


blockMeshDict

```
convertToMeters 1;
vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);
blocks
(hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1);
edges
(     );
boundary
(
    tempA
    {
        type inlet;
        faces
        (    (4 7 3 0)    );
    }
    tempB
    {
        type outlet;
        faces
        (   (1 2 6 5)   );
    }
    frontAndBack
    {
        type empty;
        faces
        (    (0 3 2 1)
             (4 5 6 7)
             (2 3 7 6)
             (0 1 5 4)   );
    }
);
```



How to define the mesh in OpenFOAM?

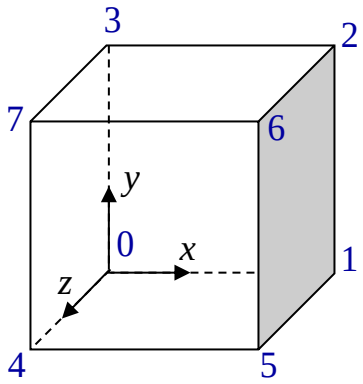


blockMeshDict

convertToMeters 1;

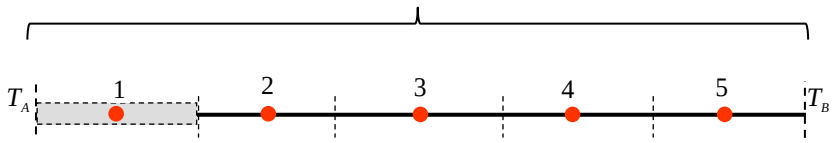
vertices

```
(
    (0 0 0)      //0
    (1 0 0)      //1
    (1 1 0)      //2
    (0 1 0)      //3
    (0 0 0.1)    //4
    (1 0 0.1)    //5
    (1 1 0.1)    //6
    (0 1 0.1)    //7
);
```

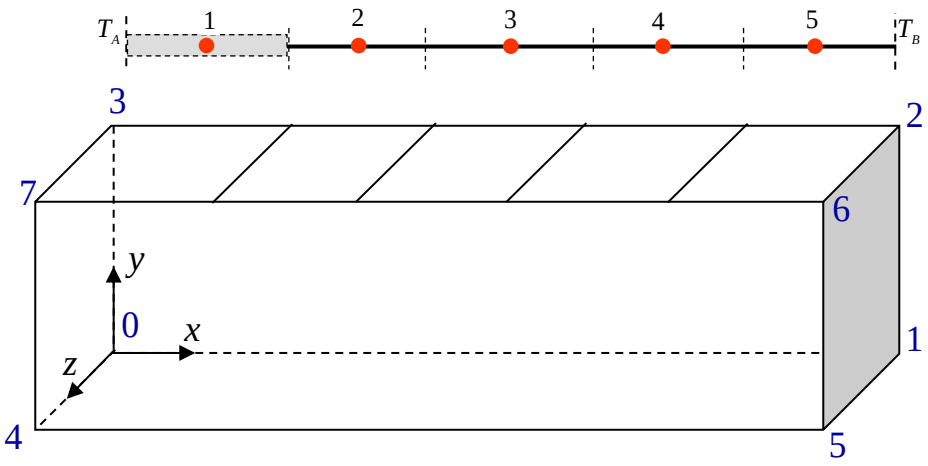


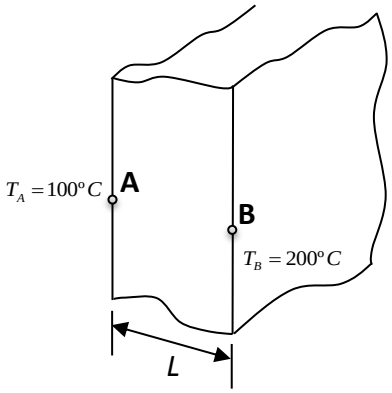
blocks

(hex (0 1 2 3 4 5 6 7) (**5** 1 1) simpleGrading (1 1 1);



How to define the **mesh** in OpenFOAM?

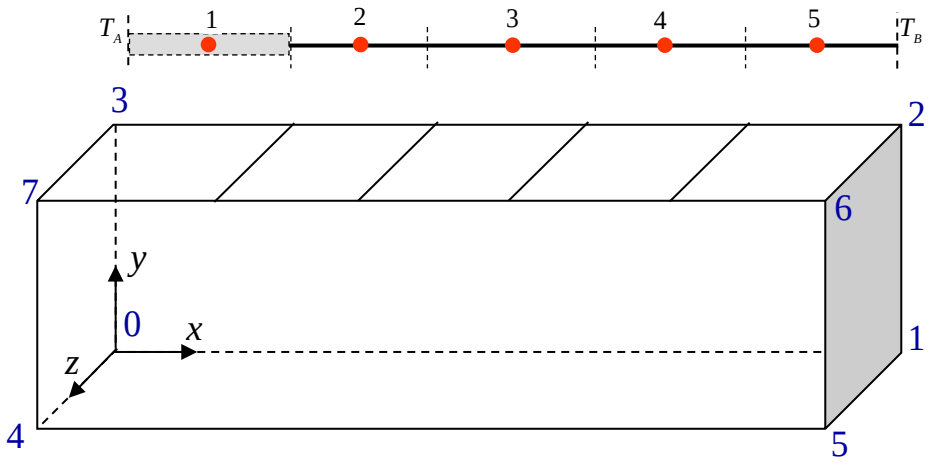


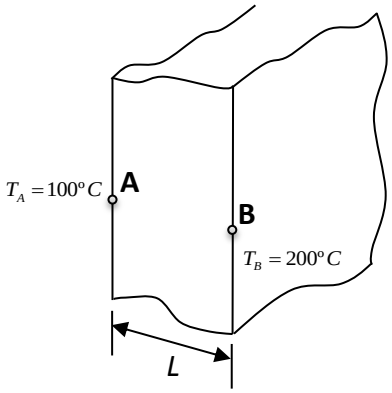


How to define the mesh in OpenFOAM?

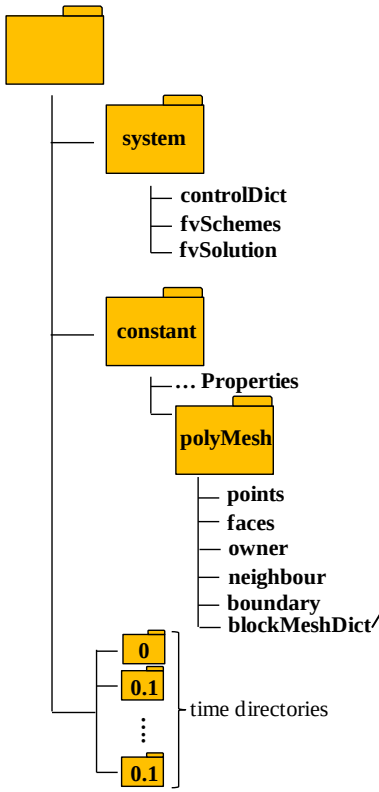
blockMeshDict

```
convertToMeters 1;
vertices
(
    (0 0 0)              //0
    (0.02 0 0)           //1
    (0.01 0.001 0)       //2
    (0 0.001 0)          //3
    (0 0 0.001)          //4
    (0.02 0 0.001)       //5
    (0.01 0.001 0.001)   //6
    (0 0.001 0.001)      //7
);
```



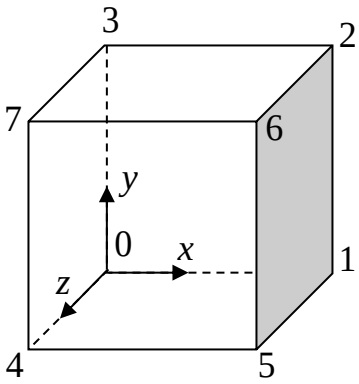


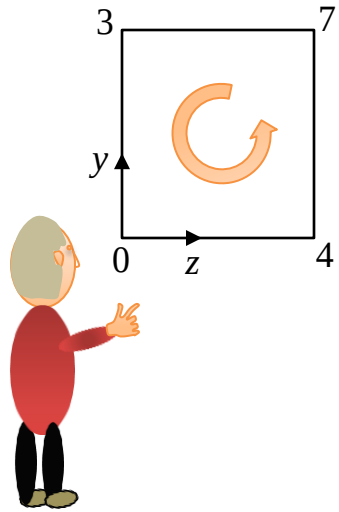
How to define the mesh in OpenFOAM?



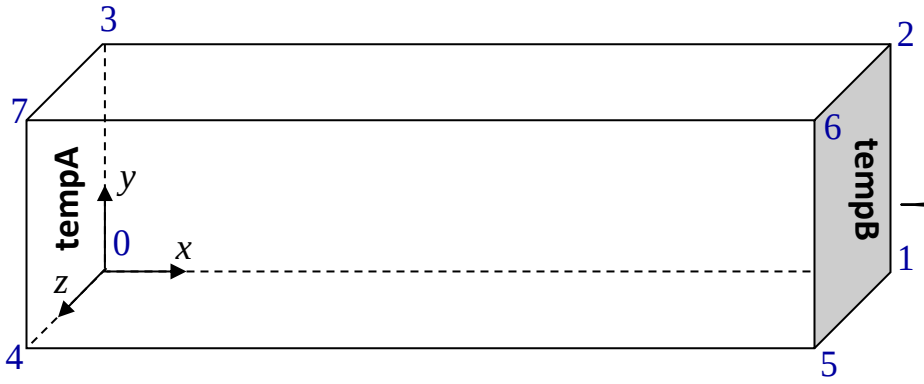
blockMeshDict

```
boundary
(
    tempA
    {
        type inlet;
        faces
        (    (4 7 3 0)    );
    }
    tempB
    {
        type outlet;
        faces
        (    (1 2 6 5)   );
    }
    frontAndBack
    {
        type empty;
        faces
        (    (0 3 2 1)
             (4 5 6 7)
             (2 3 7 6)
             (0 1 5 4)   );
    }
);
```



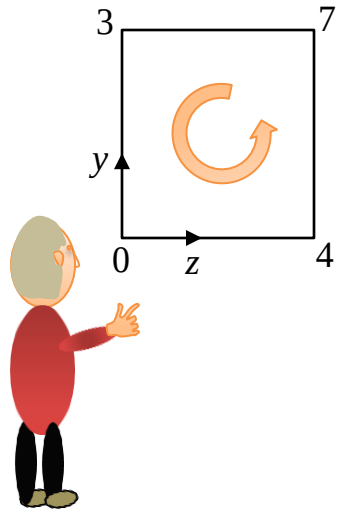


How to define the mesh in OpenFOAM?

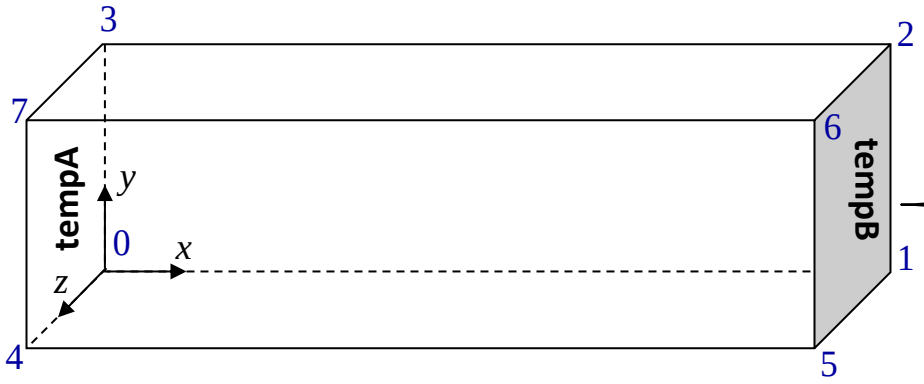


blockMeshDict

```
boundary
(
    tempA
    {
        type patch;
        faces
        (    (4 7 3 0)    );
    }
    tempB
    {
        type patch;
        faces
        (    (1 2 6 5)   );
    }
    frontAndBack
    {
        type empty;
        faces
        (    (0 3 2 1)
             (4 5 6 7)
             (2 3 7 6)
             (0 1 5 4)   );
    }
);
```



How to define the mesh in OpenFOAM?

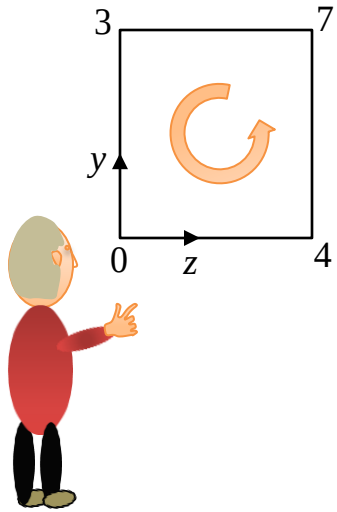


All other boundaries are empty!!!

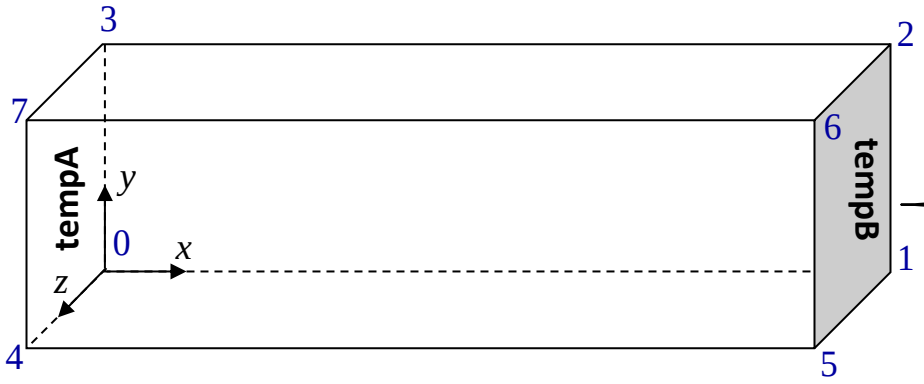
Why??!

blockMeshDict

```
boundary
(
    tempA
    {
        type patch;
        faces
        (    (4 7 3 0)    );
    }
    tempB
    {
        type patch;
        faces
        (    (1 2 6 5)   );
    }
    frontAndBack
    {
        type empty;
        faces
        (    (0 3 2 1)
             (4 5 6 7)
             (2 3 7 6)
             (0 1 5 4)   );
    }
);
```



How to define the mesh in OpenFOAM?



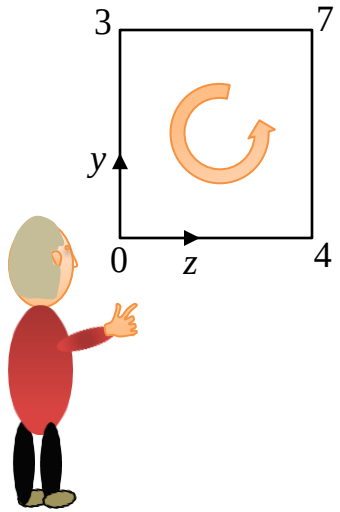
All other boundaries are **empty!!!**

Why??!

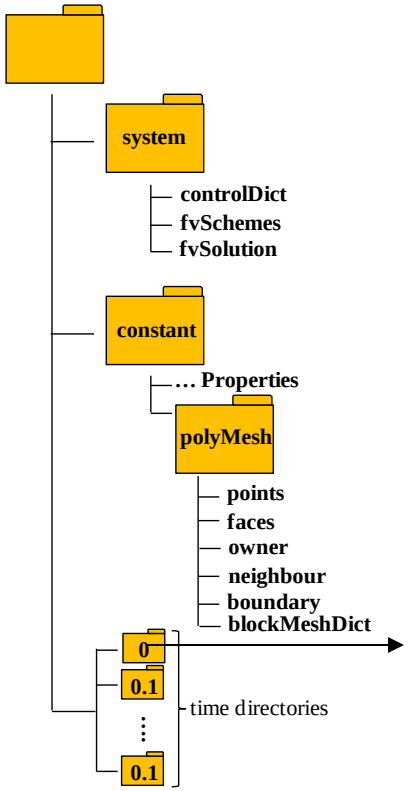
1D

blockMeshDict

```
boundary
(
    tempA
    {
        type patch;
        faces
        (    (4 7 3 0)    );
    }
    tempB
    {
        type patch;
        faces
        (    (1 2 6 5)   );
    }
    frontAndBack
    {
        type empty;
        faces
        (    (0 3 2 1)
             (4 5 6 7)
             (2 3 7 6)
             (0 1 5 4)   );
    }
);
```



Boundary conditions?



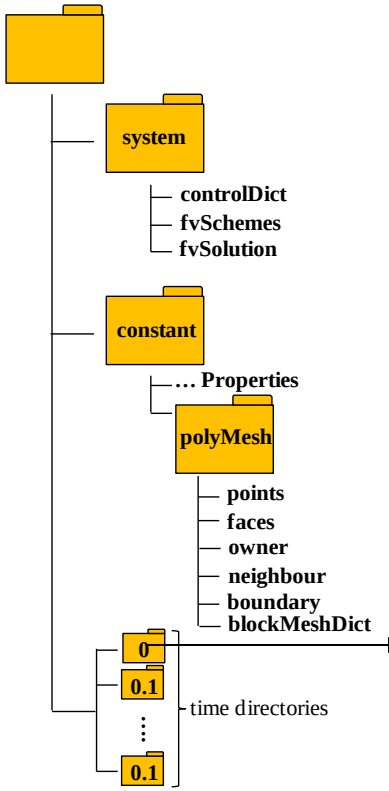
T

0 folder

T

```
dimensions      [0 0 0 1 0 0 0];

internalField   uniform 0;

boundaryField
{
    tempA
    {
        type            fixedValue;
        value           uniform 100;
    }

    tempB
    {
        type            fixedValue;
        value           uniform 200;
    }

    frontAndBack
    {
        type            empty;
    }

}
```

Boundary conditions?



T

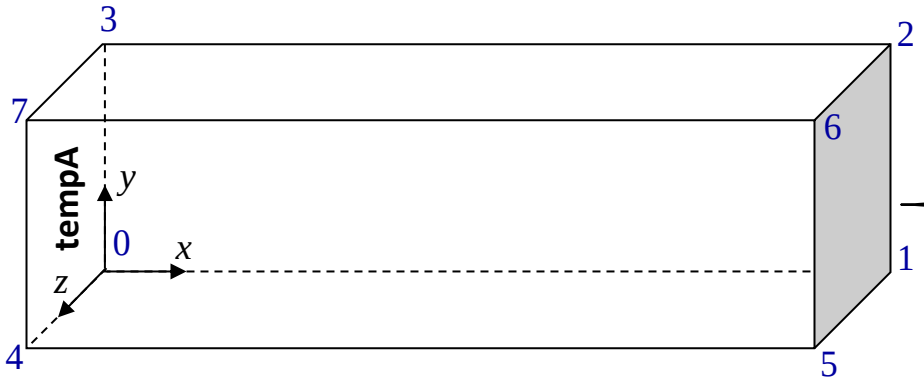
0 folder

T

dimensions **[0 0 0 1 0 0 0];**

No.	Property	SI unit
1	Mass	kilogram (kg)
2	Length	metre (m)
3	Time	second (s)
4	Temperature	Kelvin (K))
5	Quantity	mole (mol)
6	Current	ampere (A)
7	Luminous intensity	candela (cd)

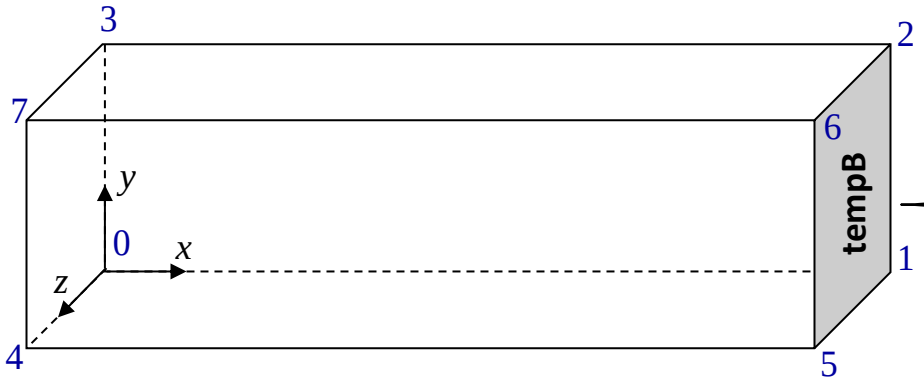
Boundary conditions?



T

```
dimensions      [0 0 0 1 0 0 0];

internalField   uniform 0;

boundaryField
{
    tempA
    {
        type            fixedValue;
        value           uniform 100;
    }

    tempB
    {
        type            fixedValue;
        value           uniform 200;
    }

    frontAndBack
    {
        type            empty;
    }

}
```

Boundary conditions?



T

```
dimensions      [0 0 0 1 0 0 0];

internalField   uniform 0;

boundaryField
{
    tempA
    {
        type            fixedValue;
        value           uniform 100;
    }

    tempB
    {
        type            fixedValue;
        value           uniform 200;
    }

    frontAndBack
    {
        type            empty;
    }

}
```

Where were we?

We want to solve this system of linear equations

$$
\begin{cases}
k_e \dfrac{T_2 - T_1}{\delta x} - k_w \dfrac{T_1 - T_A}{\delta x / 2} + q\delta x = 0 \\
k_e \dfrac{T_3 - T_2}{\delta x} - k_w \dfrac{T_2 - T_1}{\delta x} + q\delta x = 0 \\
k_e \dfrac{T_4 - T_3}{\delta x} - k_w \dfrac{T_3 - T_2}{\delta x} + q\delta x = 0 \\
k_e \dfrac{T_5 - T_4}{\delta x} - k_w \dfrac{T_4 - T_3}{\delta x} + q\delta x = 0 \\
k_e \dfrac{T_B - T_5}{\delta x / 2} - k_w \dfrac{T_5 - T_4}{\delta x} + q\delta x = 0
\end{cases}
$$

$$
\underbrace{\begin{bmatrix}
375 & -125 & 0 & 0 & 0 \\
-125 & 250 & -125 & 0 & 0 \\
0 & -125 & 250 & -125 & 0 \\
0 & 0 & -125 & 250 & -125 \\
0 & 0 & 0 & -125 & 375
\end{bmatrix}}_{\mathbf{A}}
\underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}}
=
\underbrace{\begin{bmatrix} 4 + 250T_A \\ 4 \\ 4 \\ 4 \\ 4 + 250T_B \end{bmatrix}}_{\mathbf{B}}
$$

How can we solve the system of equations?

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

a) Just guess a solution!?

b) Solve with paper & pencil!?

c) Compute the solution directly!?

d) Compute the solution using an iterative procedure!?

How can we solve the system of equations?

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

a) Just guess a solution!?

b) Solve with paper & pencil!? It works for this case!!!

c) Compute the solution directly!?

d) Compute the solution using an iterative procedure!?

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 110 \\ 130 \\ 150 \\ 170 \\ 190 \end{bmatrix}$$

How can we solve the system of equations?

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

c) Compute the solution directly!?

d) Compute the solution using an iterative procedure!?

Compute the solution directly

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

Solution by Cramer's Rule

This method is rather inefficient and relatively difficult to program!

Requires at least $3n^3$ operations!!!

Compute the solution using an iterative procedure

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

This methods are more efficient and provide answers to a linear system of equations in considerably fewer steps, but at a level of accuracy that will depend on the number of times the algorithm is applied

Requires $O(n^2)$ 0perations for each iteration

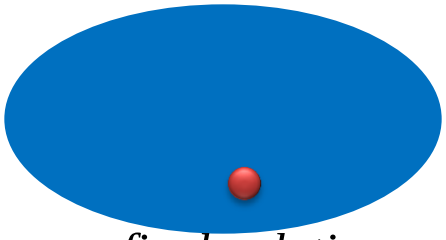
Compute the solution using an iterative procedure

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

For large systems of equations is better to use iterative methods!!!

Compute the solution using an iterative procedure

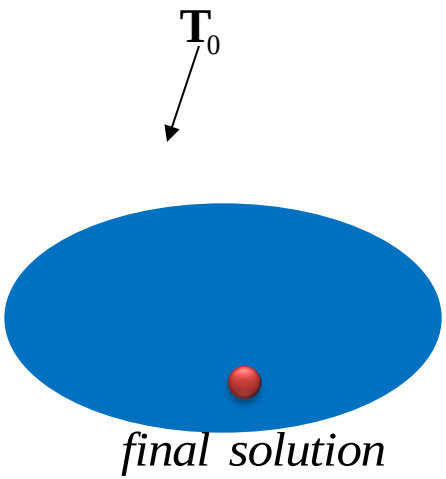
$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$



final solution

Compute the solution using an iterative procedure

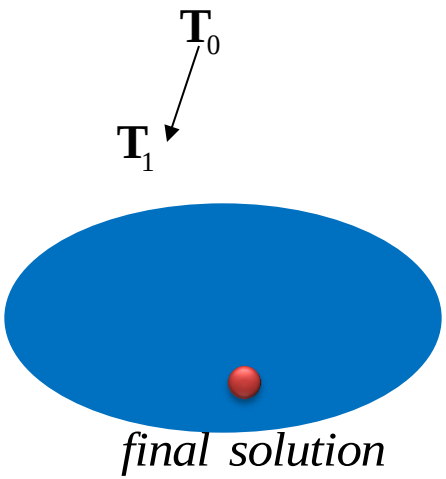
$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

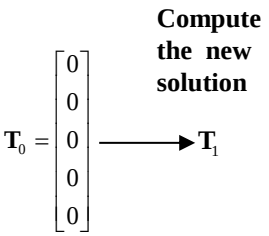


$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Compute the solution using an iterative procedure

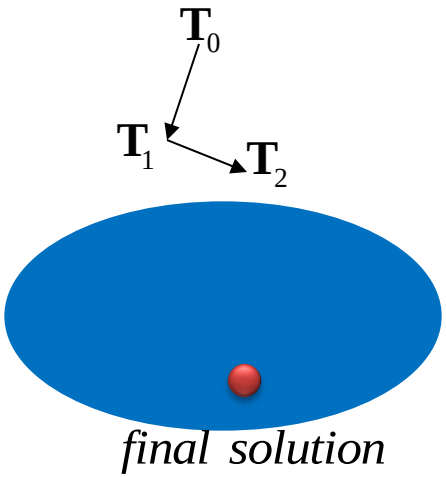
$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$





Compute the solution using an iterative procedure

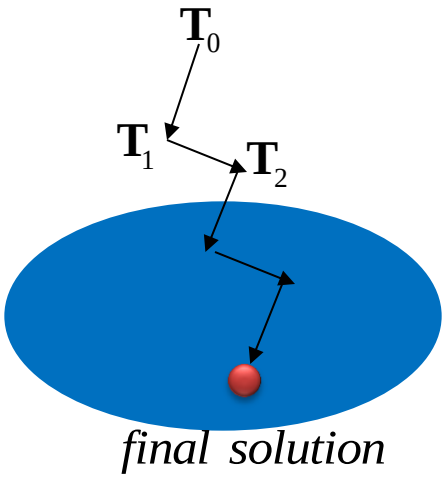
$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

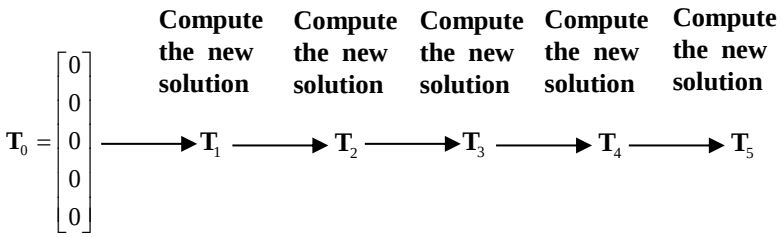


$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ → **Compute the new solution** $\mathbf{T}_1$ → **Compute the new solution** $\mathbf{T}_2$

Compute the solution using an iterative procedure

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$





Compute the solution using an iterative procedure

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

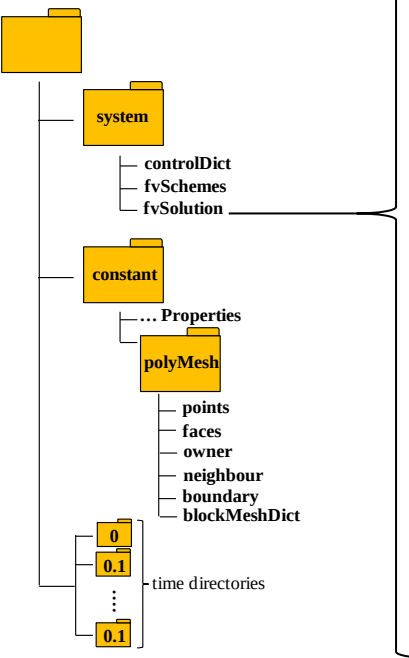
Compute the solution using an iterative procedure

$$\underbrace{\begin{bmatrix} 375 & -125 & 0 & 0 & 0 \\ -125 & 250 & -125 & 0 & 0 \\ 0 & -125 & 250 & -125 & 0 \\ 0 & 0 & -125 & 250 & -125 \\ 0 & 0 & 0 & -125 & 375 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\mathbf{T}} = \underbrace{\begin{bmatrix} 4+250T_A \\ 4 \\ 4 \\ 4 \\ 4+250T_B \end{bmatrix}}_{\mathbf{B}}$$

PCG (preconditioned conjugate gradient method)

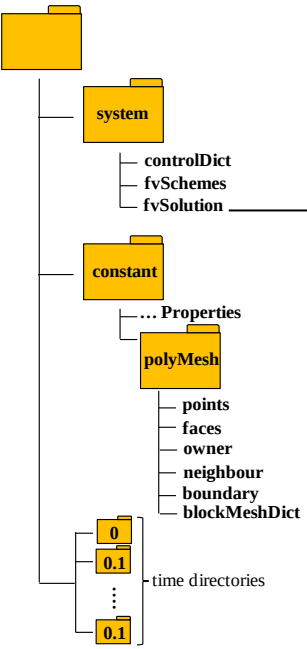
The CG method reaches the exact solution of $Ax=b$ in at most ***n*** steps for any initial guess!

fvSolution




```
 object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
    T
    {
        solver                  PCG;
        preconditioner          DIC
        tolerance               1e-07;
        relTol                  0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 2;
}
```

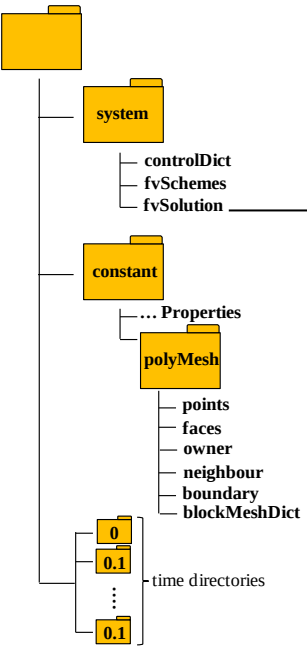
fvSolution




```
 object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
    T
    {
        solver          PCG;
        preconditioner  DIC
        tolerance       1e-07;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 2;
}
```

iterative method

fvSolution

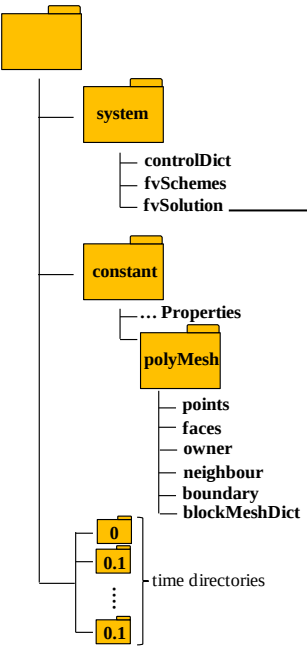



```
 object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
    T
    {
        solver          PCG;
        preconditioner  DIC
        tolerance       1e-07;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 2;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

fvSolution




```
 object     fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver          PCG;
      preconditioner  DIC
      tolerance       1e-07;
      relTol          0;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 2;
}
```

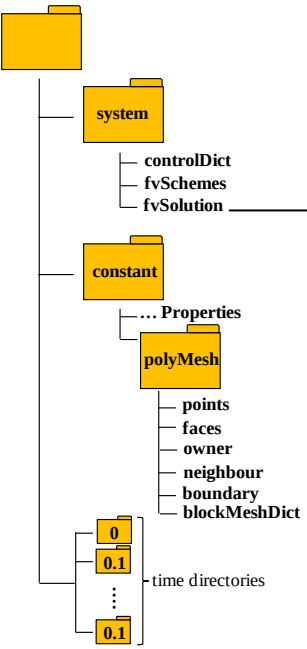
iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{T}_1 \rightarrow \mathbf{T}_2 \rightarrow \mathbf{T}_3 \rightarrow \mathbf{T}_4 \rightarrow \mathbf{T}_5$$

fvSolution




```
 object     fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver          PCG;
      preconditioner  DIC
      tolerance       1e-07;
      relTol          0;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 2;
}
```

iterative method

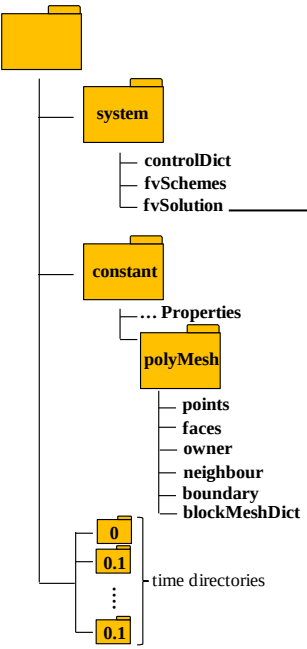
Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{T}_1 \rightarrow \mathbf{T}_2 \rightarrow \mathbf{T}_3 \rightarrow \mathbf{T}_4 \rightarrow \mathbf{T}_5$$

$$r = \left\| \mathbf{A}\mathbf{T}_0 - \mathbf{B} \right\|$$

fvSolution




```
 object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
    T
    {
        solver          PCG;
        preconditioner  DIC
        tolerance       1e-07;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 2;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{T}_1 \rightarrow \mathbf{T}_2 \rightarrow \mathbf{T}_3 \rightarrow \mathbf{T}_4 \rightarrow \mathbf{T}_5$$

$$r = \left\| \mathbf{A}\mathbf{T}_0 - \mathbf{B} \right\| \qquad r = \left\| \mathbf{A}\mathbf{T}_3 - \mathbf{B} \right\| = 1e-8 < 1e-7$$

fvSolution

```
 object     fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver          PCG;
      preconditioner  DIC
      tolerance       1e-07;
      relTol          0.1;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 2;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

relTol is the relative tolerance between the initial and the final residual

fvSolution

```
 object     fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver          PCG;
      preconditioner  DIC
      tolerance       1e-07;
      relTol          0.1;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 2;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

relTol is the relative tolerance between the initial and the final residual

$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{T}_1 \rightarrow \mathbf{T}_2 \rightarrow \mathbf{T}_3 \rightarrow \mathbf{T}_4 \rightarrow \mathbf{T}_5$$

fvSolution

```
 object     fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver          PCG;
      preconditioner  DIC
      tolerance       1e-07;
      relTol          0.1;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 2;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

relTol is the relative tolerance between the initial and the final residual

$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{T}_1 \rightarrow \mathbf{T}_2 \rightarrow \mathbf{T}_3 \rightarrow \mathbf{T}_4 \rightarrow \mathbf{T}_5$$

$$r = \left\| \mathbf{A}\mathbf{T}_0 - \mathbf{B} \right\| = 0.55$$

fvSolution

```
 object     fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver          PCG;
      preconditioner  DIC
      tolerance       1e-07;
      relTol          0.1;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 2;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

relTol is the relative tolerance between the initial and the final residual

$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{T}_1 \rightarrow \mathbf{T}_2 \rightarrow \mathbf{T}_3 \rightarrow \mathbf{T}_4 \rightarrow \mathbf{T}_5$$

$$r = \left\| \mathbf{A}\mathbf{T}_0 - \mathbf{B} \right\| = 0.55 \qquad r = \left\| \mathbf{A}\mathbf{T}_3 - \mathbf{B} \right\| = 0.032$$

fvSolution

```
 object     fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver          PCG;
      preconditioner  DIC
      tolerance       1e-07;
      relTol          0.1;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 2;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

relTol is the relative tolerance between the initial and the final residual

$$\mathbf{T}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{T}_1 \rightarrow \mathbf{T}_2 \rightarrow \mathbf{T}_3 \rightarrow \mathbf{T}_4 \rightarrow \mathbf{T}_5$$

$$r = \left\| \mathbf{A}\mathbf{T}_0 - \mathbf{B} \right\| = 0.55 \qquad r = \left\| \mathbf{A}\mathbf{T}_3 - \mathbf{B} \right\| = 0.032$$

$$0.1 \times 0.55 = 0.055$$

$$0.032 < 0.055$$

fvSolution

Terminal:

```
PCG    :  Solving for T, Initial residual = 0.584235, Final residual = 1.0582e-07, No Iterations 3
```

fvSolution

Terminal:

```
PCG    :  Solving for T, Initial residual = 0.584235, Final residual = 1.0582e-07, No Iterations 3
```

solver

fvSolution

Terminal:

```
PCG    :  Solving for T, Initial residual = 0.584235, Final residual = 1.0582e-07, No Iterations 3
```

$$r = \left\| \mathbf{A}\mathbf{T}_0 - \mathbf{B} \right\|$$

solver

fvSolution

Terminal:

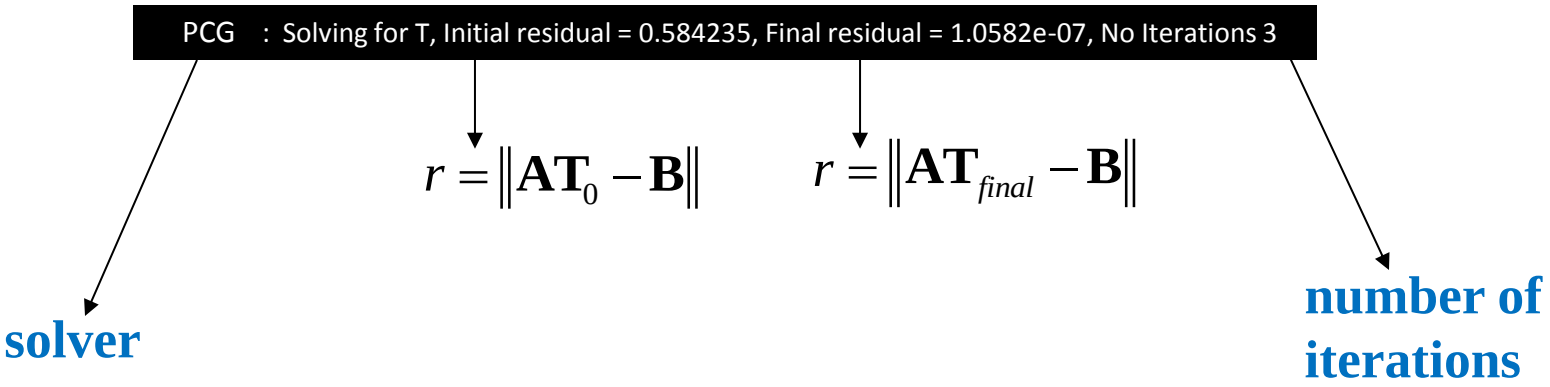
```
PCG    : Solving for T, Initial residual = 0.584235, Final residual = 1.0582e-07, No Iterations 3
```

$$r = \left\| \mathbf{AT}_0 - \mathbf{B} \right\| \qquad r = \left\| \mathbf{AT}_{final} - \mathbf{B} \right\|$$

solver

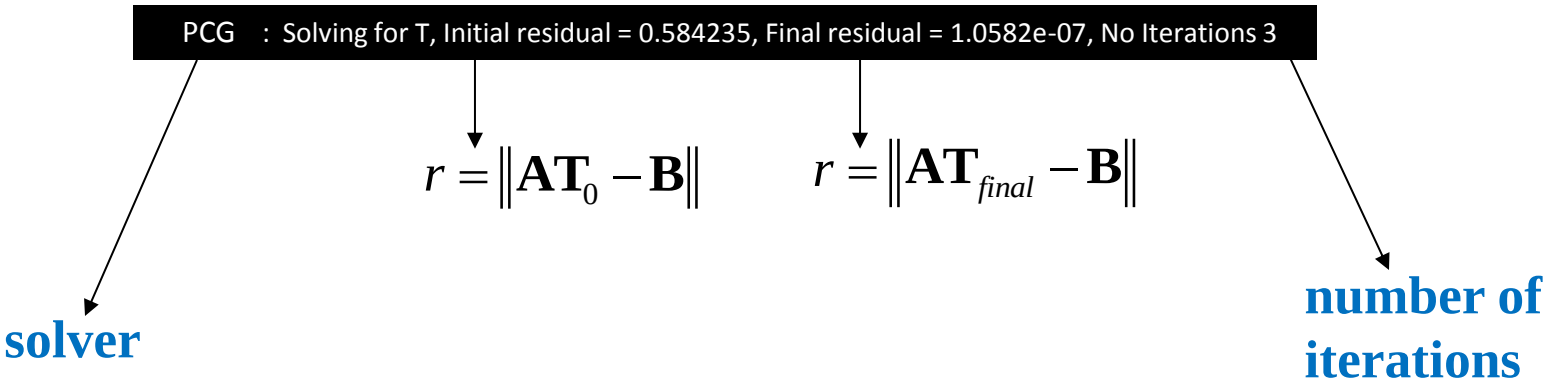
fvSolution

Terminal:



fvSolution

Terminal:

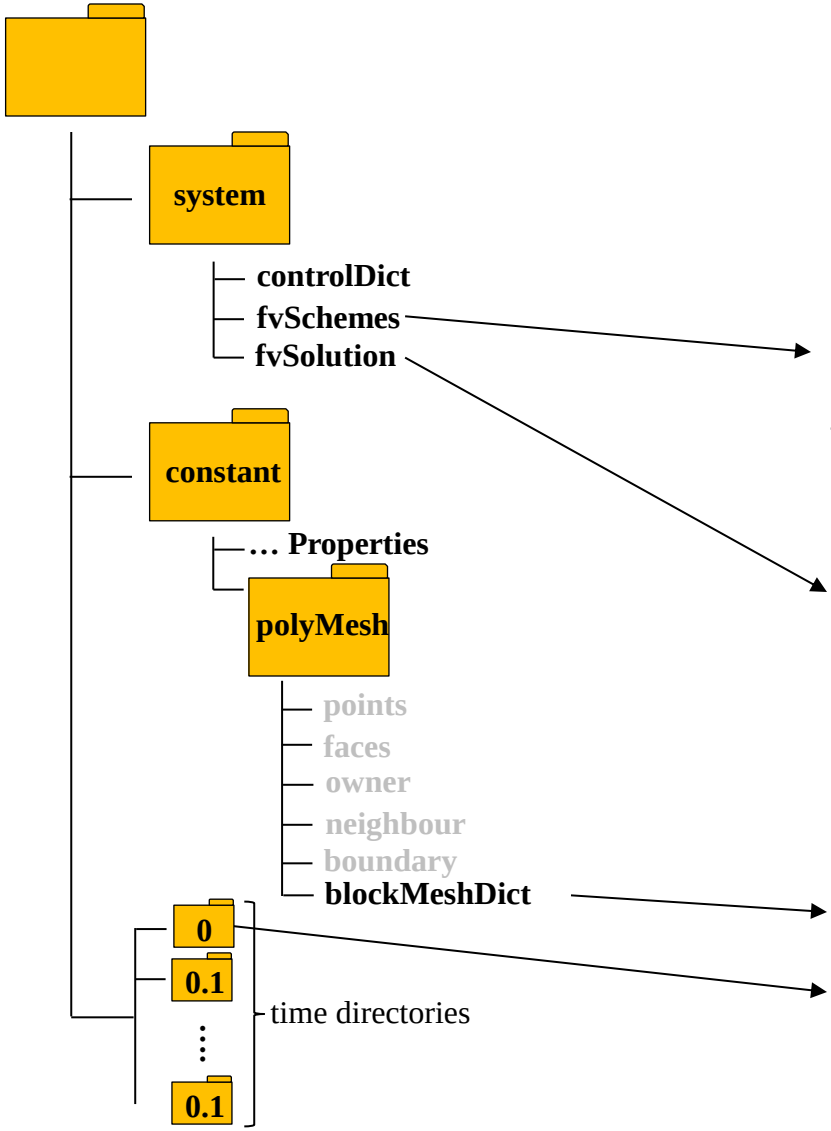


In OpenFOAM all residuals are normalized:

Confused?

We want to solve the differential equation

$$\frac{d}{dx}\left(k\frac{dT}{dx}\right)+q=0$$



Define the approximations for the operators

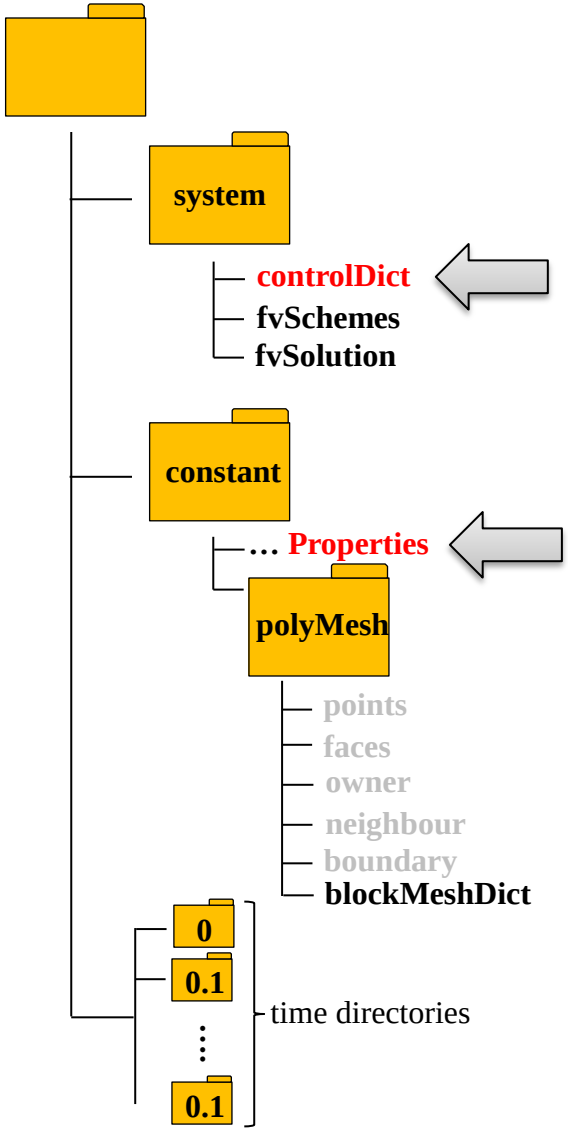
Define how to solve the system of equations

Create the mesh

Define boundary conditions

We want to solve the differential equation

$$\frac{d}{dx}\left(k\frac{dT}{dx}\right)+q=0$$



controlDict

```
application     laplacianFoamTemp;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         1;
deltaT          1;
writeControl    adjustableRunTime;
writeInterval   1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
graphFormat     raw;
runTimeModifiable yes;
adjustTimeStep on;
maxCo           0.8;
maxDeltaT       0.001;
```

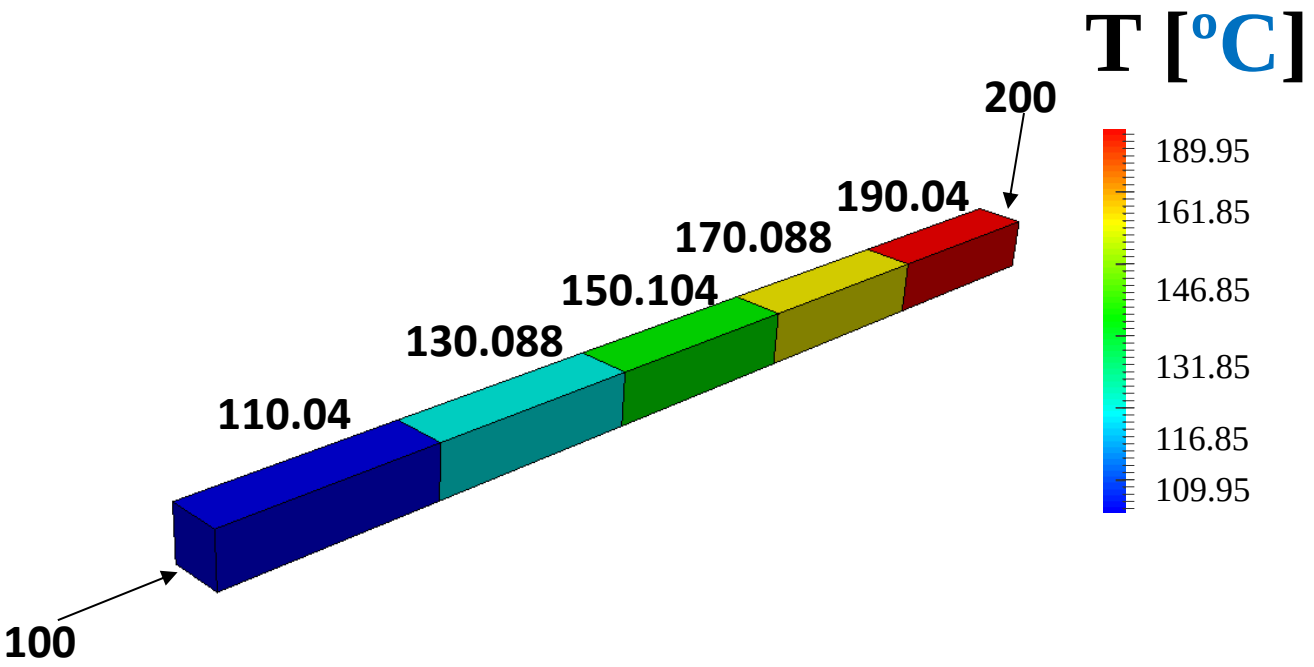
transportProperties

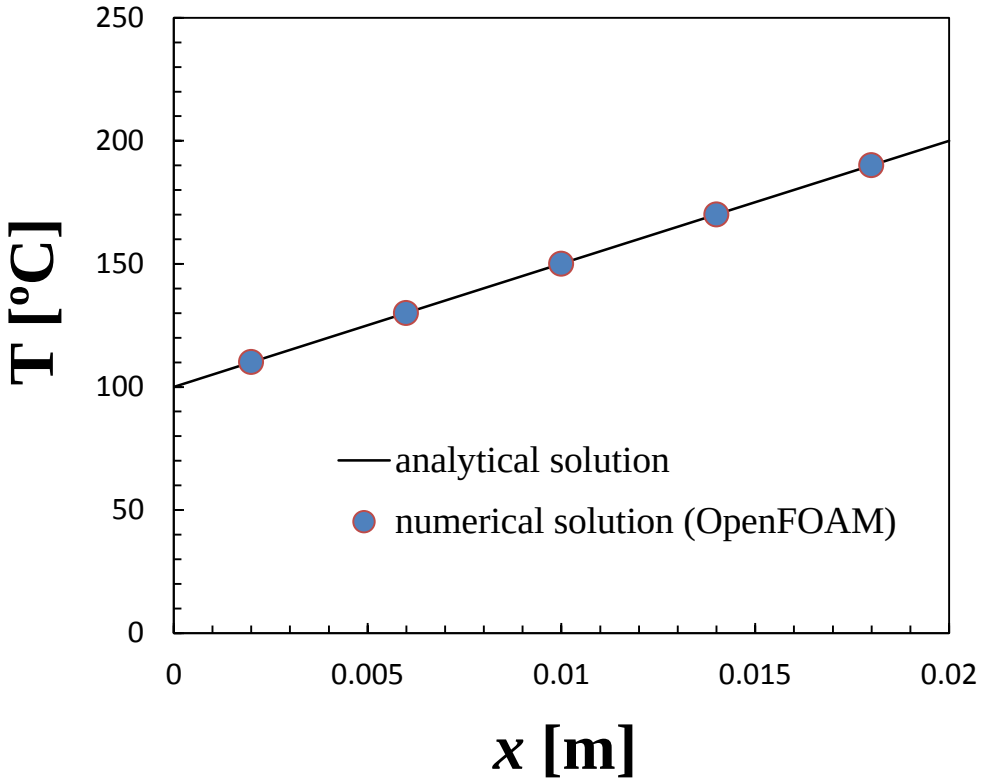
```
DT              DT [ 1 1 -3 -1 0 0 0 ] 0.5;
q               q [ 1 -1 -3 0 0 0 0 ] 1000;
```

What to do next?

```
»cd $WM_PROJECT_USER_DIR
»cd applications
»cd laplacianFoamTemp
»wclean
»wmake
```

```
»cd $WM_PROJECT_USER_DIR
»cd basicTut
»cd laplacianFoamTemp
»blockMesh
»checkMesh
»laplacianFoamTemp
»paraFoam
```


T [ºC]
200
190.04
170.088
150.104
130.088
110.04
100
189.95
161.85
146.85
131.85
116.85
109.95

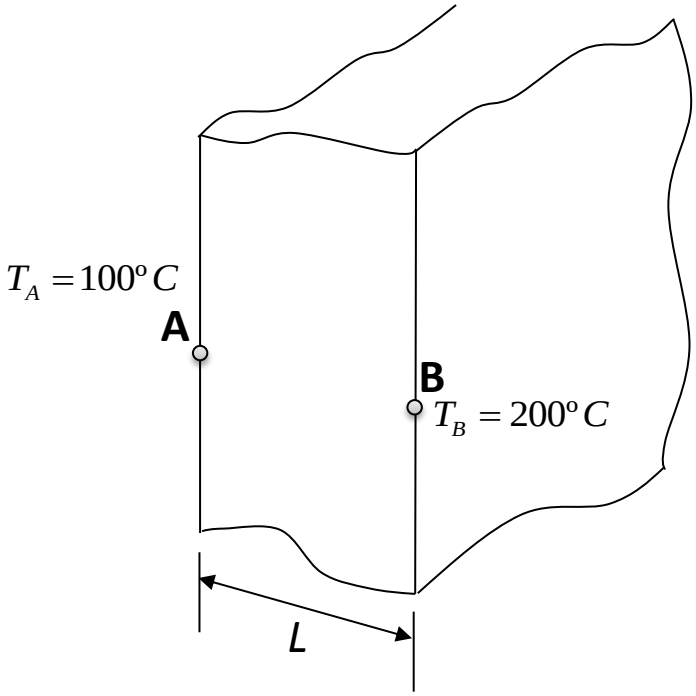

250
200
150
100
50
0
T [ºC]
0
0.005
0.01
0.015
0.02
x [m]
analytical solution
numerical solution (OpenFOAM)

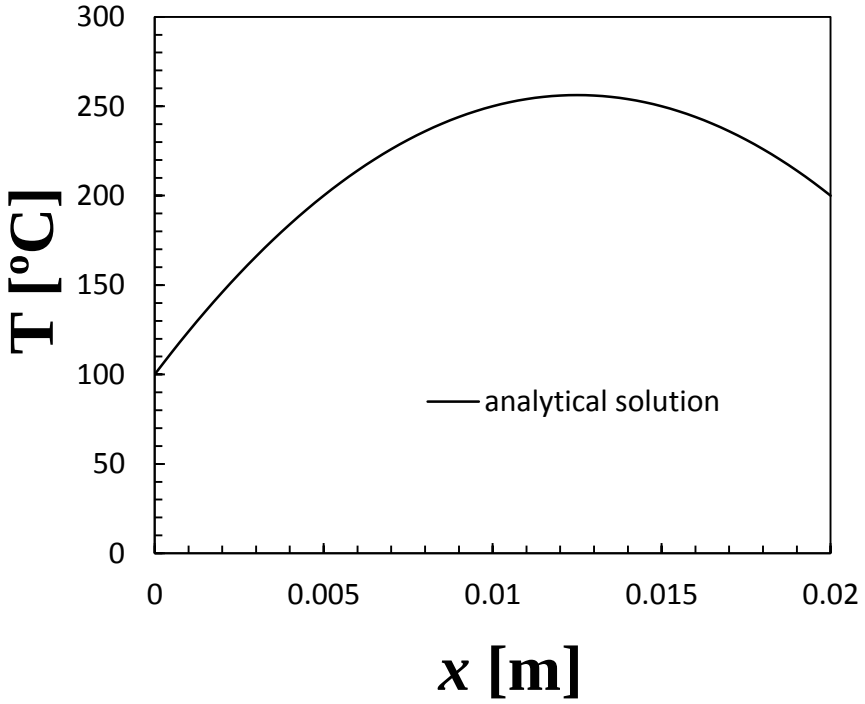
It is time for you to do it by yourselves!!

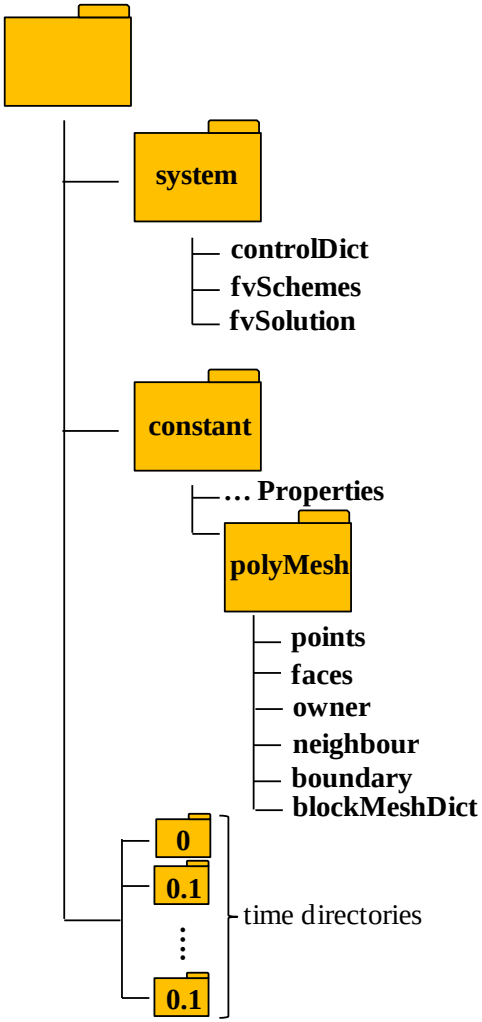
Consider the same problem with ***q*=1E6 W/m³** use only **4** cells!!!

Remember that ***L*=0.02 m**

***k*= 0.5 W/mK**

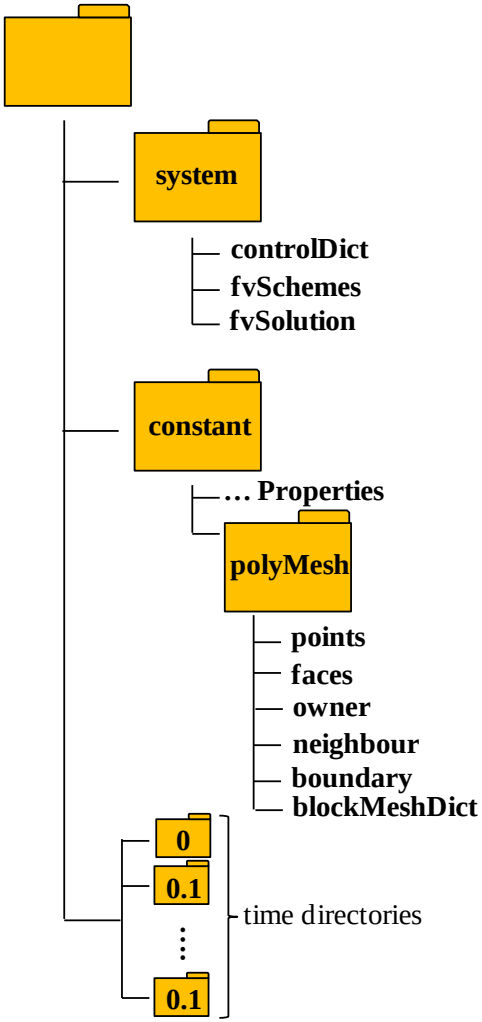






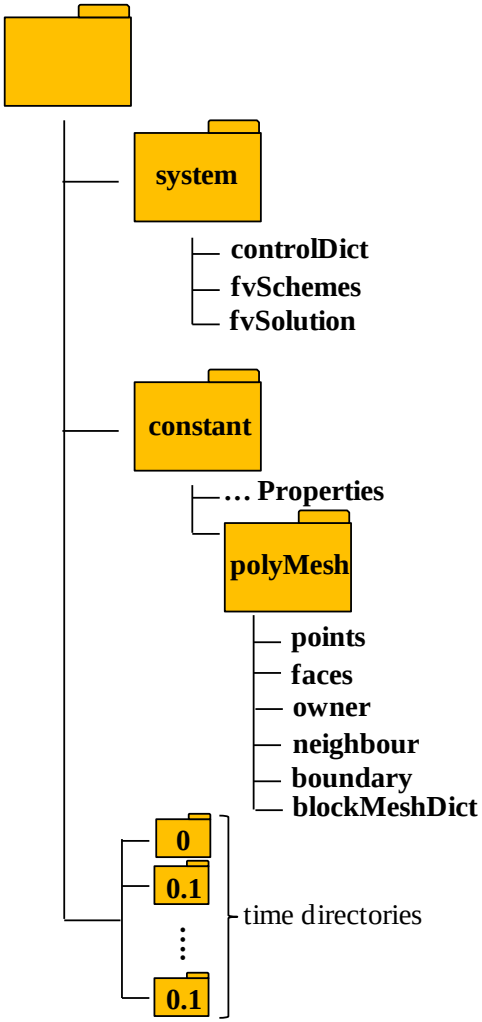
blockMeshDict

```
convertToMeters 1;
vertices
 (
    (0 0 0)
    (0.02 0 0)
    (0.01 0.001 0)
    (0 0.001 0)
    (0 0 0.001)
    (0.02 0 0.001)
    (0.01 0.001 0.001)
    (0 0.001 0.001)
);
 blocks
(hex (0 1 2 3 4 5 6 7) (4 1 1) simpleGrading (1 1 1);
```



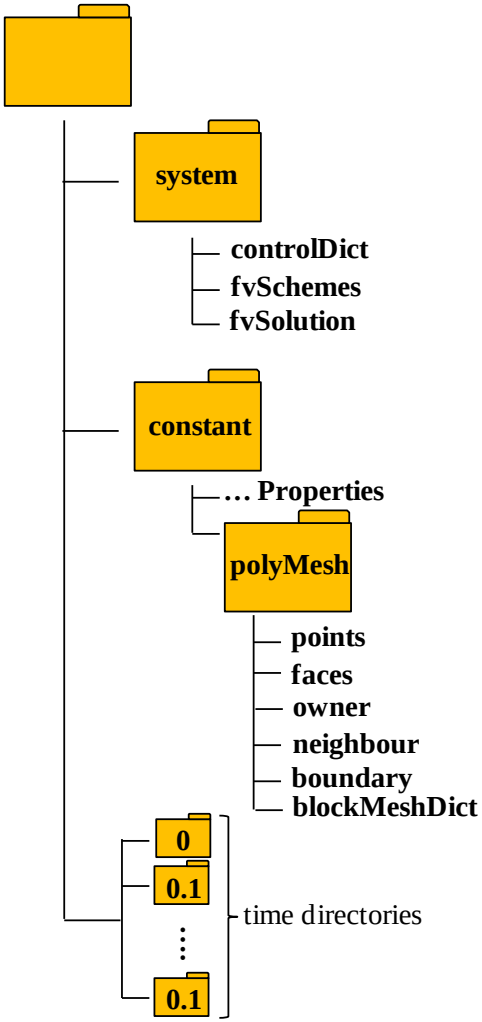
blockMeshDict

```
boundary
(
    tempA
    {
        type patch;
        faces
        (    (4 7 3 0)    );
    }
    tempB
    {
        type patch;
        faces
        (    (1 2 6 5)   );
    }
    frontAndBack
    {
        type empty;
        faces
        (    (0 3 2 1)
             (4 5 6 7)
             (2 3 7 6)
             (0 1 5 4)   );
    }
);
```



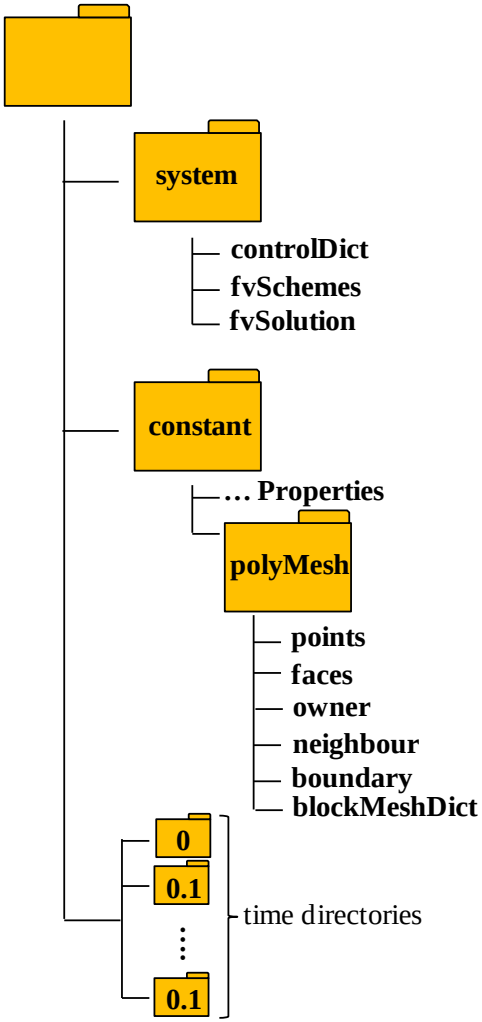
fvSchemes

No need to change (for this case)



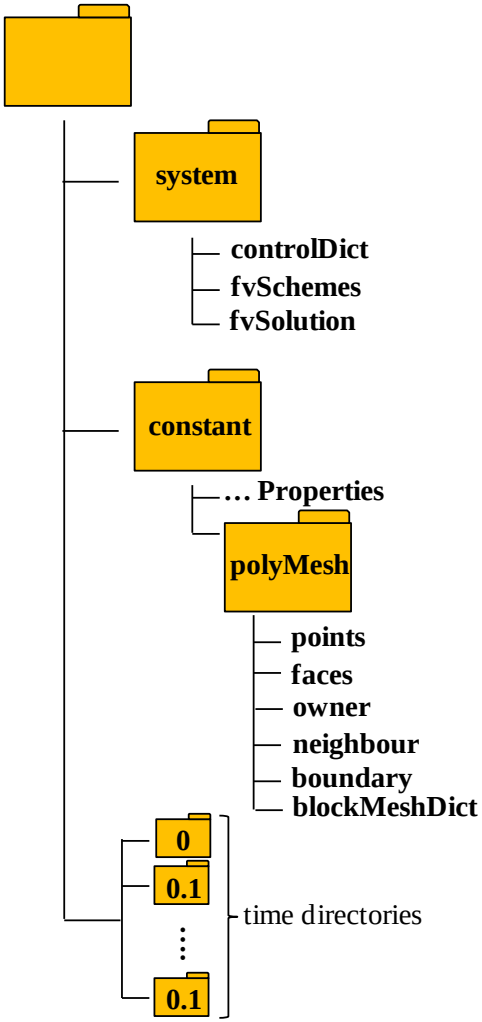
fvSolution

```
 object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
   T
   {
      solver                PCG;
      preconditioner        DIC
      tolerance             1e-07;
      relTol                0;
   }
}

SIMPLE
{
   nNonOrthogonalCorrectors 0;
}
```



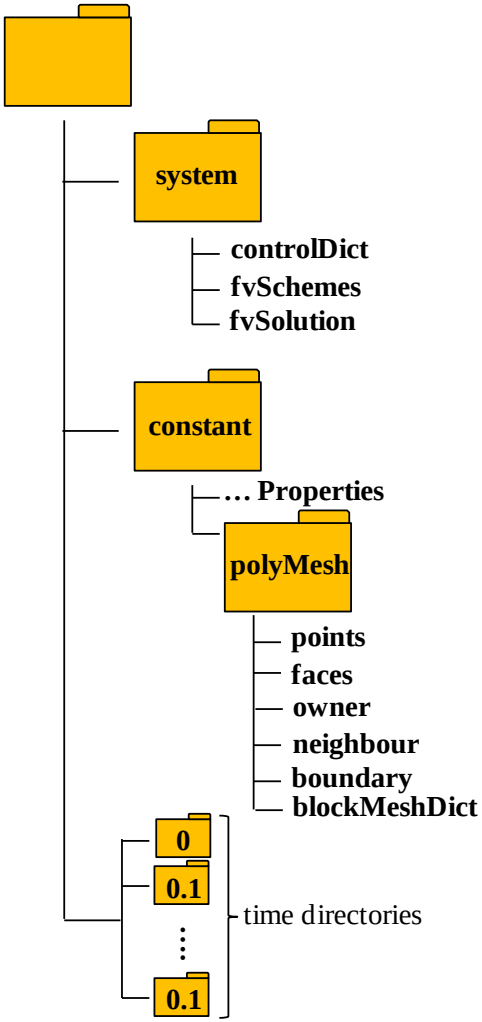
T

```
dimensions      [0 0 0 1 0 0 0];

internalField   uniform 0;

boundaryField
{
    tempA
    {
        type            fixedValue;
        value           uniform 100;
    }

    tempB
    {
        type            fixedValue;
        value           uniform 200;
    }

    frontAndBack
    {
        type            empty;
    }

}
```



transportProperties

```
application     laplacianFoamTemp;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         1;
deltaT          1;
writeControl    adjustableRunTime;
writeInterval   1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
graphFormat     raw;
runTimeModifiable yes;
adjustTimeStep on;
maxCo           0.8;
maxDeltaT       0.001;
```



transportProperties

DT DT [1 1 -3 -1 0 0 0] **0.5**;

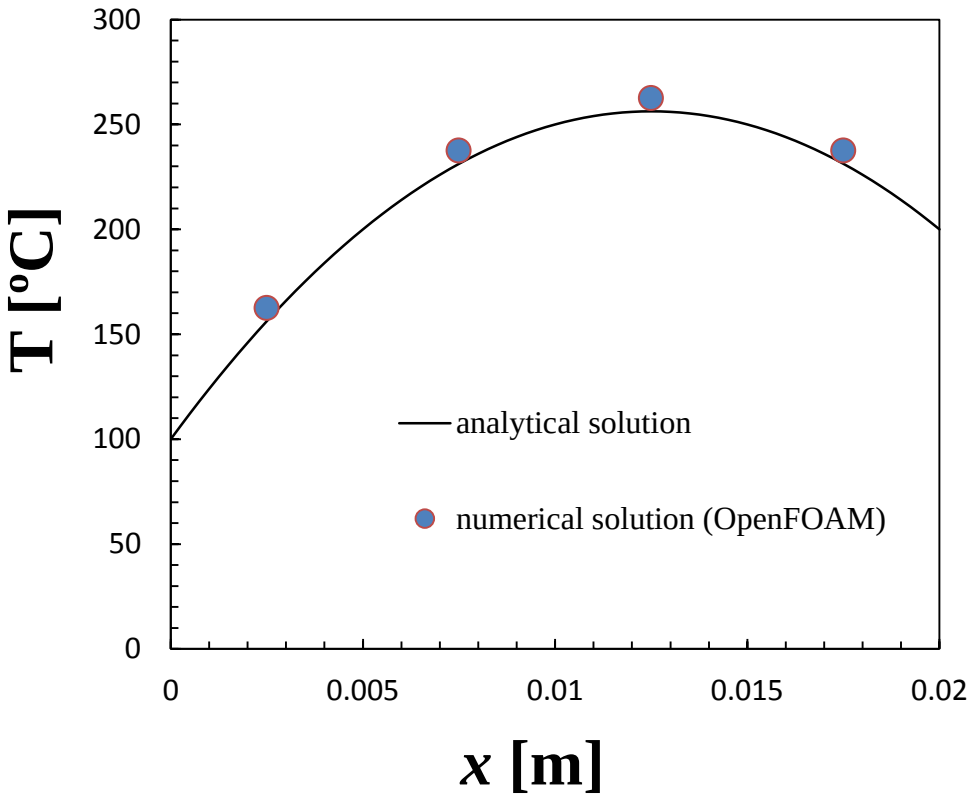
q q [1 -1 -3 0 0 0 0] **1 000 000**;

»blockMesh

»checkMesh

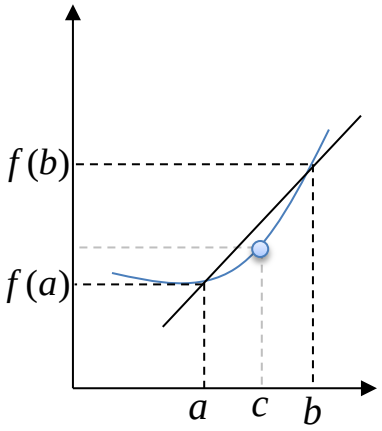
»laplacianFoamTemp

»paraFoam

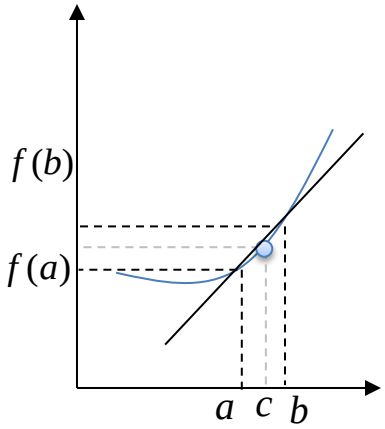


Why the results are not so good??!

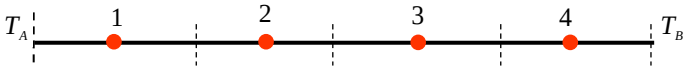
Which case provides a more accurate solution?



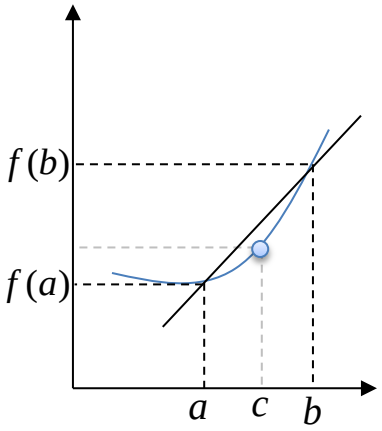
$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$



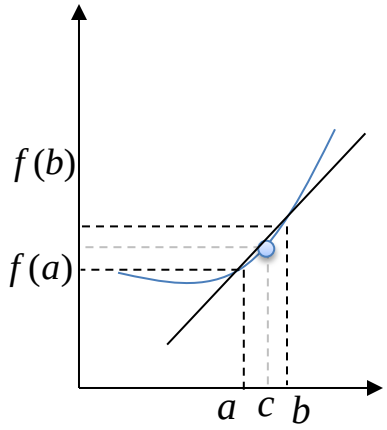
$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$



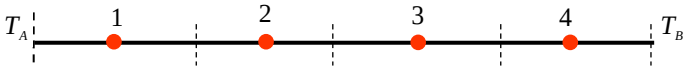
Which case provides a more accurate solution?

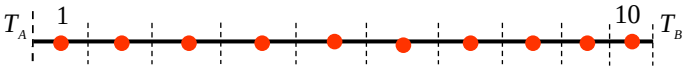


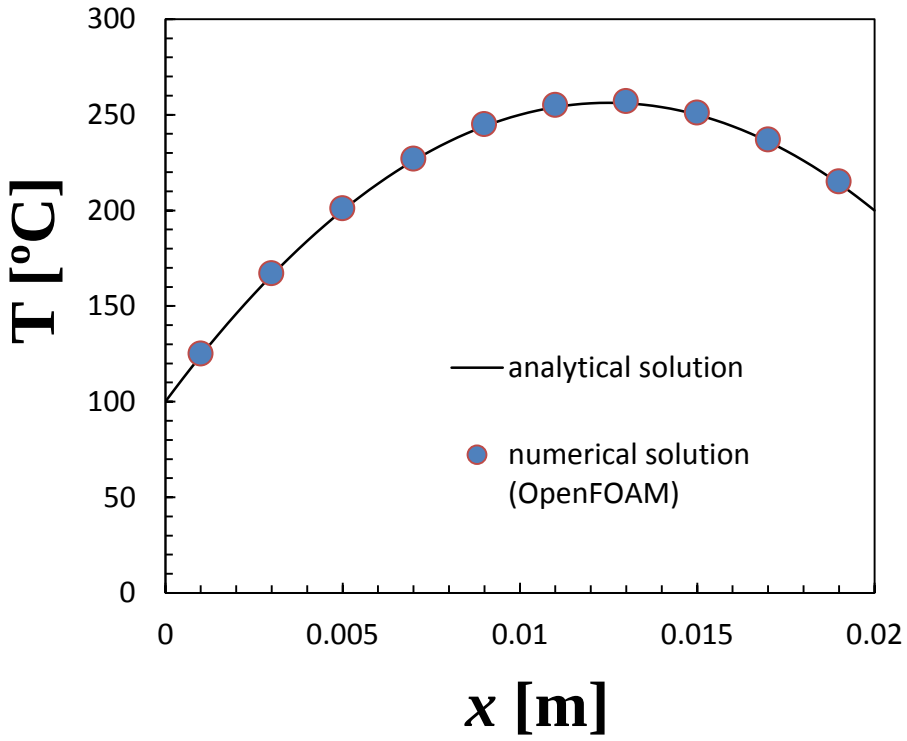
$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$



$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$






300
250
200
150
100
50
0
T [°C]
0
0.005
0.01
0.015
0.02
x [m]
analytical solution
numerical solution
(OpenFOAM)

Próximas sessões:
11:00-12:30

B2 – Geração de malha com o blockMesh

A2 – swak4Foam and pyFoam