Computational Fluid Dynamics

Introduction to numerical methods in CFD II

Lecture 17

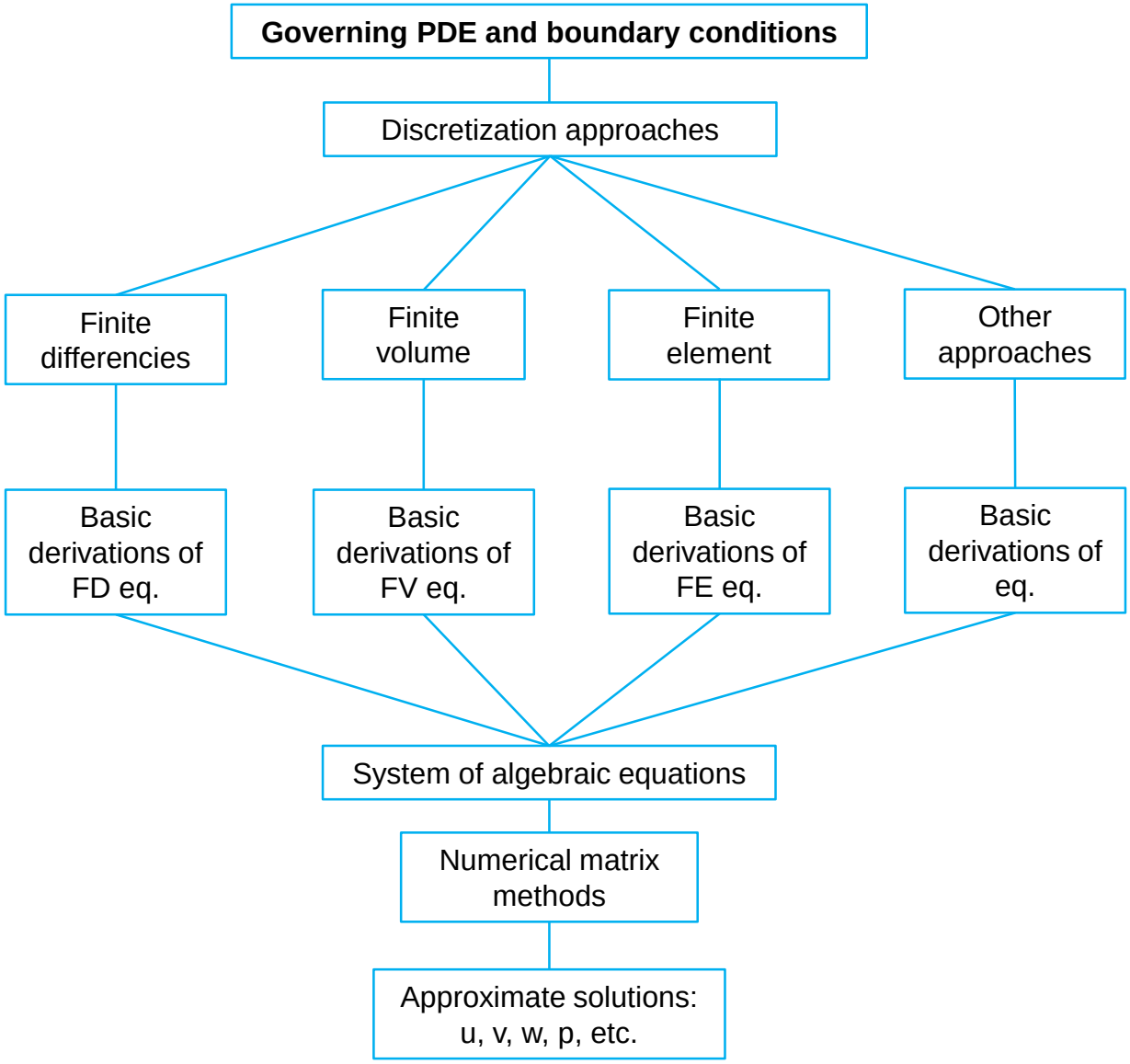
Krister Wiklund
Department of Physics
Umeå University

SUMMARY OF LECTURE:

Introduction to numerical methods in CFD II

- Be aware of differences between DVM ,LBM and FEM methods
- Be aware of differencies in spatial discretization between FEM & FV
- Be aware of main steps in the formulation of a discrete representation (matrix eq.) of a PDE problem using FEM
- Be aware of what the order of an element in FEM means
- Be aware of advantages and disadvantages of first and second order
- Be aware of the effect of mesh-refinement and element order increase

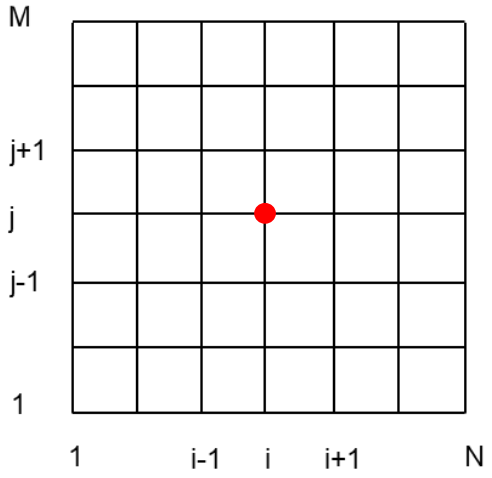
Recap


Governing PDE and boundary conditions
Discretization approaches
Finite differencies
Finite volume
Finite element
Other approaches
Basic derivations of FD eq.
Basic derivations of FV eq.
Basic derivations of FE eq.
Basic derivations of eq.
System of algebraic equations
Numerical matrix methods
Approximate solutions: u, v, w, p, etc.

Finite Difference Method

- Oldest method, used for example by L. F. Richardson (1910)
- Courant, Fredrichson and Lewy (1928) derived stability criteria for explicit time stepping (CFL-criteria)

Spatial discretization



$$\begin{cases} \phi_{i+1,j} = \phi_{i,j} + \left(\frac{\partial \phi}{\partial x}\right)_{i,j} \Delta x + \left(\frac{\partial^2 \phi}{\partial x^2}\right)_{i,j} \frac{\Delta x^2}{2} + \ldots \\ \phi_{i-1,j} = \phi_{i,j} - \left(\frac{\partial \phi}{\partial x}\right)_{i,j} \Delta x + \left(\frac{\partial^2 \phi}{\partial x^2}\right)_{i,j} \frac{\Delta x^2}{2} - \ldots \end{cases}$$

Truncation error

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x} + O(\Delta x)$$

$$\left(\frac{\partial \phi}{\partial y}\right)_{i,j} = \frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta y} + O(\Delta x)$$

Forward differences

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x} + O(\Delta x)$$ Forward difference

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x} + O(\Delta x)$$ Backward difference

Useful at the boundaries

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} + O(\Delta x^2)$$ Central difference

More accurate.
If applied at boundaries ”ghost points” are needed

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

Time discretization

$$\left(\frac{\partial \phi}{\partial t}\right)_{i,j} = \frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\Delta t} + O(\Delta t)$$ Forward difference

Basic derivation of FD-equations

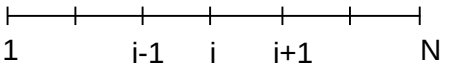
Example 1: Steady 1D diffusion

$$\frac{\partial}{\partial x}\left(\Gamma\frac{\partial \phi}{\partial x}\right)+q_\phi=0$$

Diffusion coefficient

Source

$$\Gamma\frac{\partial^2\phi}{\partial x^2}=-q_\phi$$



BC: Assume Φ known at i=1 and i=N

$$\left(\frac{\partial^2\phi}{\partial x^2}\right)_i=\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}+O(\Delta x^2)$$

$$\Rightarrow \quad \phi_{i+1}-2\phi_i+\phi_{i-1}=-\frac{\Delta x^2}{\Gamma}q_i$$

$$i=2:\quad \phi_3-2\phi_2+\phi_1=-\frac{\Delta x^2}{\Gamma}q_2$$

$$i=3:\quad \phi_4-2\phi_3+\phi_2=-\frac{\Delta x^2}{\Gamma}q_3$$

$$i=4:\ \ldots.$$

$$\vdots$$

$$i=N-1:$$

$$\Rightarrow \quad \begin{pmatrix} -2 & 1 & 0 & . \\ 1 & -2 & 1 & . \\ 0 & 1 & -2 & . \\ . & . & . & -2 \end{pmatrix}\begin{pmatrix} \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{N-1} \end{pmatrix}=\begin{pmatrix} -\frac{\Delta x^2}{\Gamma}q_2-\phi_1 \\ -\frac{\Delta x^2}{\Gamma}q_3 \\ \vdots \\ -\frac{\Delta x^2}{\Gamma}q_{N-1}-\phi_N \end{pmatrix}$$

Solving the matrix system: Direct vs Iterative methods

Direct solvers

- Finds an approximation of the solution by matrix factorization (LU-decomposition)
- Robust
- Expensive in memory
- CPU intensive
- Use for small & medium size problems

Examples

- SPOOLES * (Slowest, least memory)
- MUMPS * (Cluster computing)
- PARDISO * (Fastest)
- UMFPACK

* Available in Comsol

Iterative solvers

- Finds solution from initial guess + iterations, approaches the solution gradually
- Oscillatory behaviour in error during iterations indicate that the problem is not properly setup
- Significant less memory usage than direct solvers
- Need more tuning and does not always work, different physics require different settings

Examples

- Jacobi
- Gauss-Seidel
- Succesive overrelaxion (SOR)
- Conjugate gradient (CG) *
- BiCGStab *
- GMRES *
- FGMRES *

Good tutorials: Google ”Walter Frei Comsol blogg”

Finite Volume Method

- Most CFD-software use it, so it is well developed
- First well-documented application was by Evans and Harlow (1957)
- Used for the 2D-Euler eq. by McDonald (1971), MacCormack and Paullay (1973)
- Advantages vs FD:
  - Can be used on any type of grid
  - Momentum and energy always conserved
- Disadvantages vs FD:
  - Differencing approximations higher than 2:nd order more difficult in 3D

Summary of the control volume approach

1. Divide the domain into finite control volumes (CV)

2. Integrate relevant PDEs over CVs and apply divergence theorem

$$\frac{\partial}{\partial x}\left(\Gamma\frac{\partial \phi}{\partial x}\right)+q_\phi=0 \quad \Rightarrow \quad \int_S \Gamma\frac{\partial \phi}{\partial x}\mathbf{n}\cdot\mathbf{dS}+\int_{CV} q_\phi dV=0$$

3. Make approximations on surface and volume integrations — Not discussed here

4. Make approximations on field variable interpolation — Not discussed here

5. Collect terms and state a matrix problem — Not discussed here

Example: Convection-diffusion equation

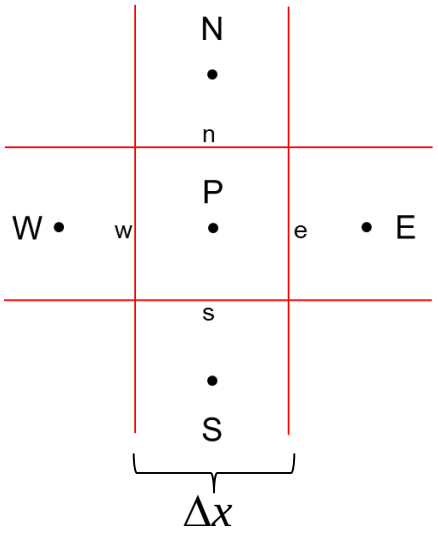
$$\rho\frac{\partial\phi}{\partial t}+\rho\mathbf{u}\cdot\nabla\phi=\nabla\cdot\left(\Gamma\nabla\phi\right)+q_\phi$$

$$\rho\left(\frac{\partial}{\partial t}+\mathbf{u}\cdot\nabla\right)\phi=\Gamma\nabla^2\phi+q_\phi$$

$$\frac{\partial\left(\rho\phi\right)}{\partial t}+\nabla\cdot\left(\rho\mathbf{u}\phi\right)=\nabla\cdot\left(\Gamma\nabla\phi\right)+q_\phi$$

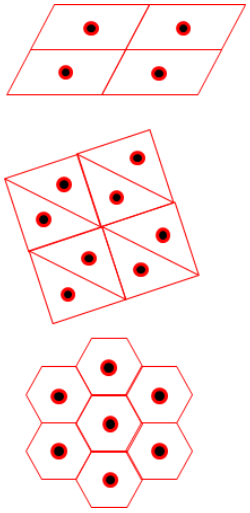
$$\left(\frac{\partial\rho}{\partial t}+\nabla\cdot\left(\rho\mathbf{u}\right)=0\right)$$

Step 1: Spatial discretization (2D uniform cell centered approach)



Common notation

- The field variable is determined in the center of each CV (N, S, W, E)
- Each surface around CV at point P is labeled by n,s,w,e



Step 2: Integration over control volumes (consider here steady case)

$$\nabla \cdot \left(\rho \mathbf{u} \phi \right) = \nabla \cdot \left(\Gamma \nabla \phi \right) + q_\phi \quad \Rightarrow \quad \int_{CV} \nabla \cdot \left(\rho \mathbf{u} \phi - \Gamma \nabla \phi \right) dV = \int_{CV} q_\phi dV$$

Divergence theorem ⇒ $\int_S f dS = \int_{CV} q_\phi dV$

Local conservation law

Flux through control surface

$$f \equiv \left(\rho \mathbf{u} \phi - \Gamma \nabla \phi \right) \cdot \mathbf{n}$$

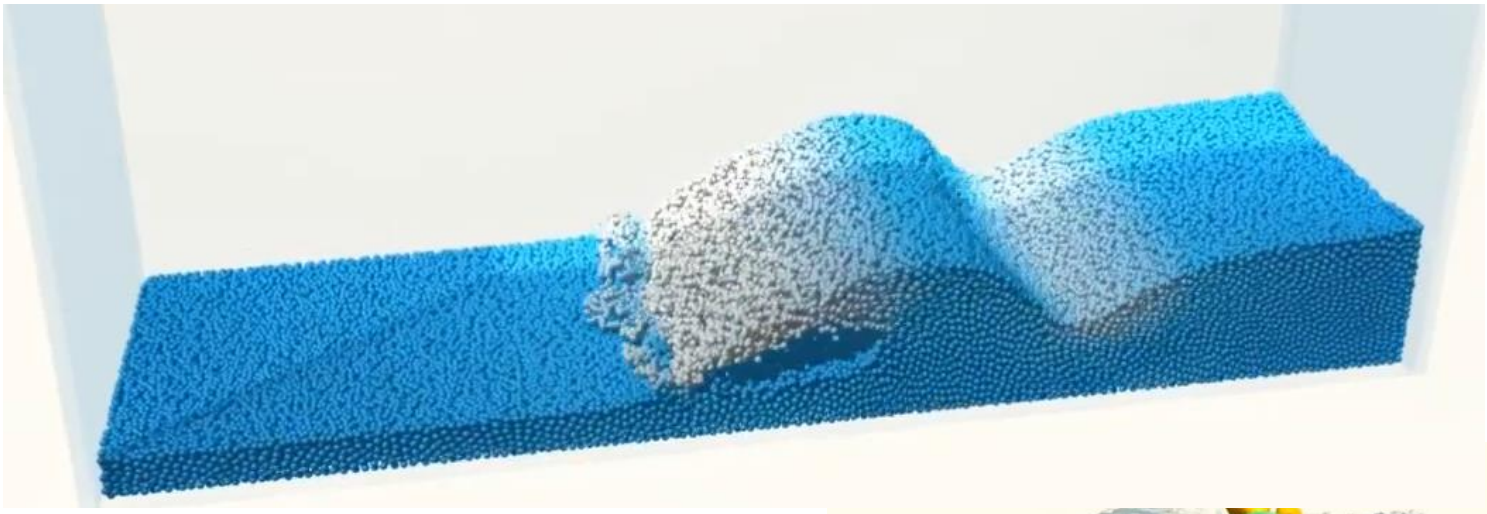
Convective Diffusive

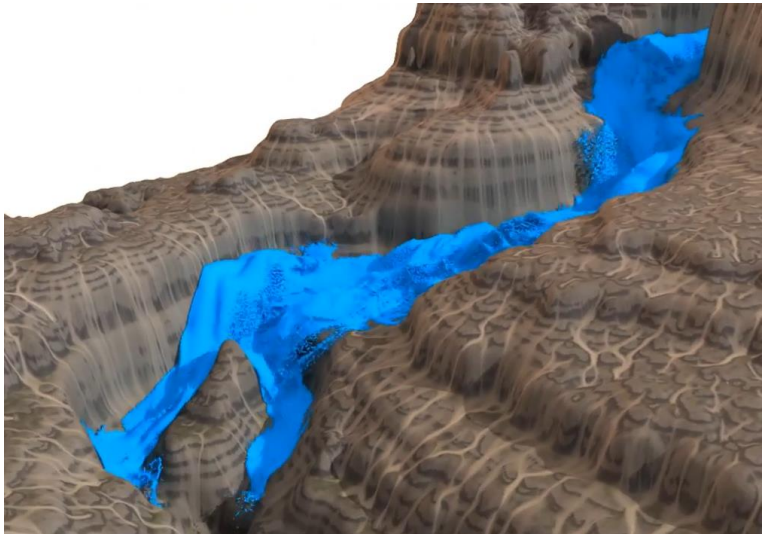
We split the surface integral into a sum of the contribution from each part of the control volume

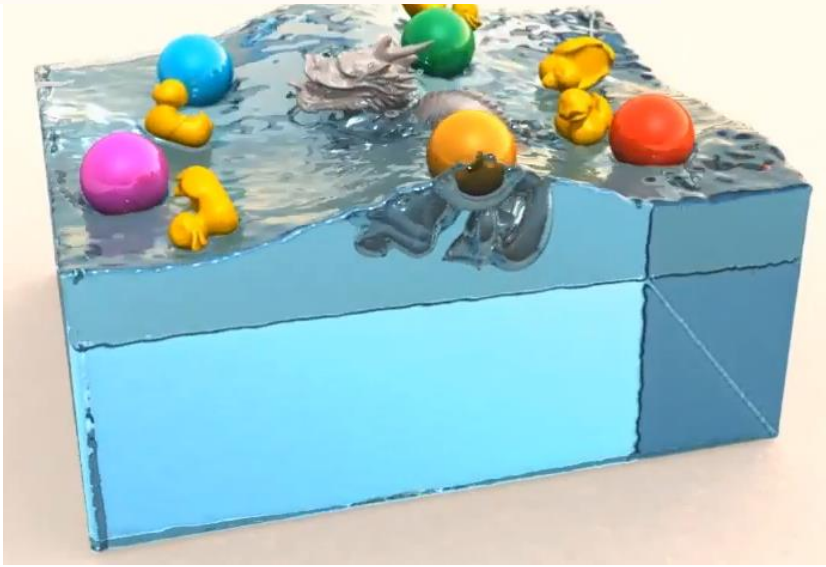
$$\int_S f dS = \sum_k \int_{S_k} f dS$$

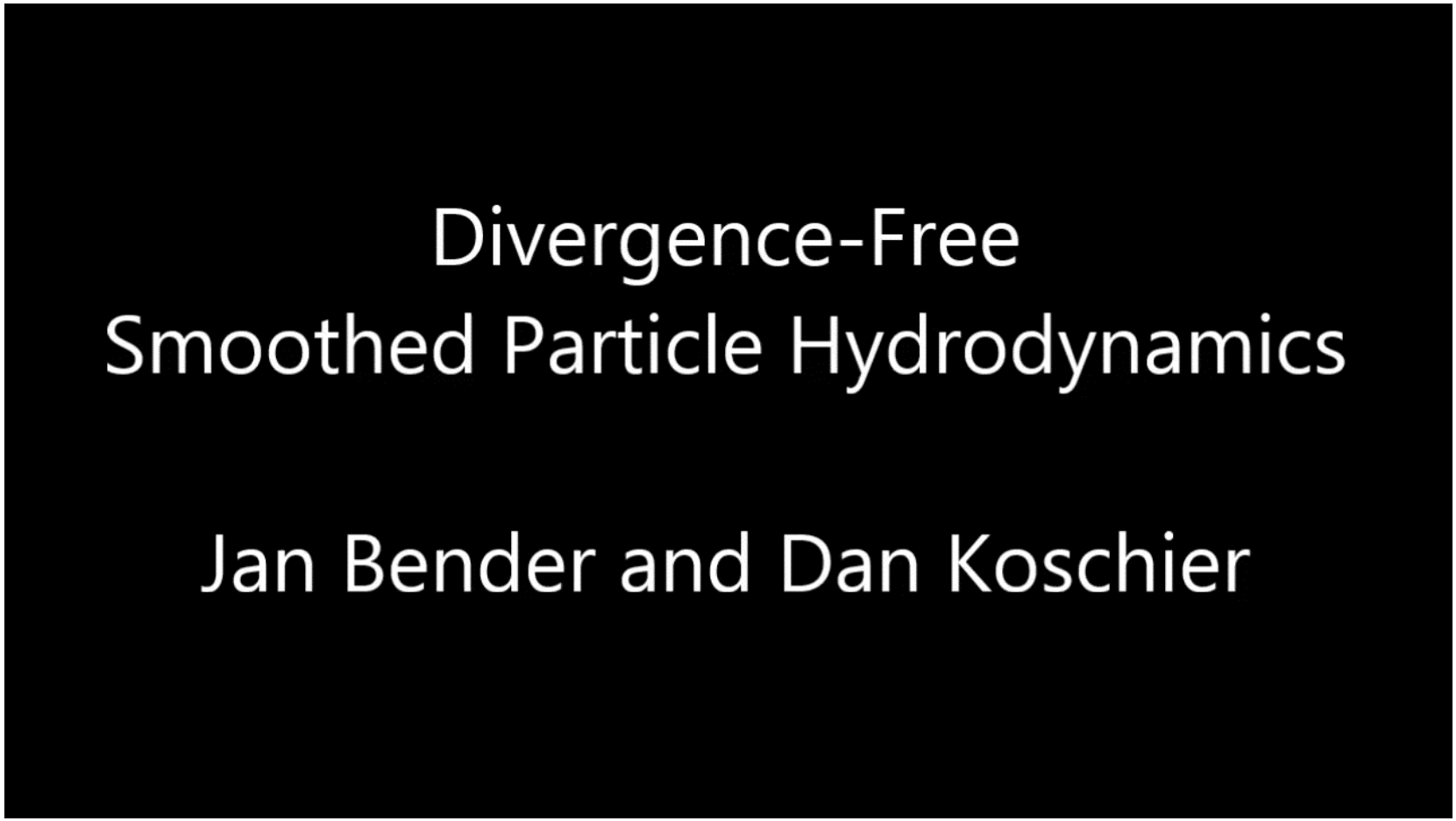
The number of surfaces depends of course on which type of mesh we use.

Smoothed Particle Hydrodynamic (SPH)








Divergence-Free
Smoothed Particle Hydrodynamics
Jan Bender and Dan Koschier

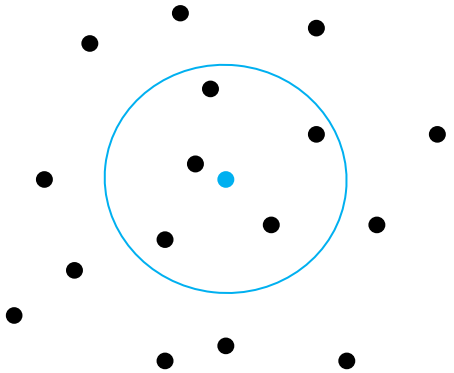
Smoothed Particle Hydrodynamics

- Introduced in astrophysics by Lucy (1977), Ginghold and Monaghan (1977)
- SPH is a mesh-free approach

MOVIE

Spatial discretization

- The fluid is represented by a particle system
- Particle properties are determined by taking an average over neighboring particles



1. Only neighbors contribute to the average
2. Near particles should contribute more than distant particles

Use a weight function to determine the average effect on the center particle (blue)

$$\langle A(r) \rangle = \int_V A(r')W(r - r')dr'$$

Some fluid property

Weight function (Kernel)

In practice we use a discrete version of the average operator:

$$\langle A \rangle_i \approx \sum_j \frac{m_j}{\rho_j} A_j W(r_{ij})$$

$$\langle \nabla A \rangle_i \approx \sum_j \frac{m_j}{\rho_j} A_j \nabla W(r_{ij})$$

$$\langle \nabla^2 A \rangle_i \approx \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W(r_{ij})$$

Example: Fluid density

$$\langle \rho \rangle_i \approx \sum_j \frac{m_j}{\rho_j} \rho_j W(r_{ij}) \approx \sum_j m_j W(r_{ij})$$

Navier-Stokes: $$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{v} + \frac{1}{\rho} F_{ext} + \mathbf{g}$$

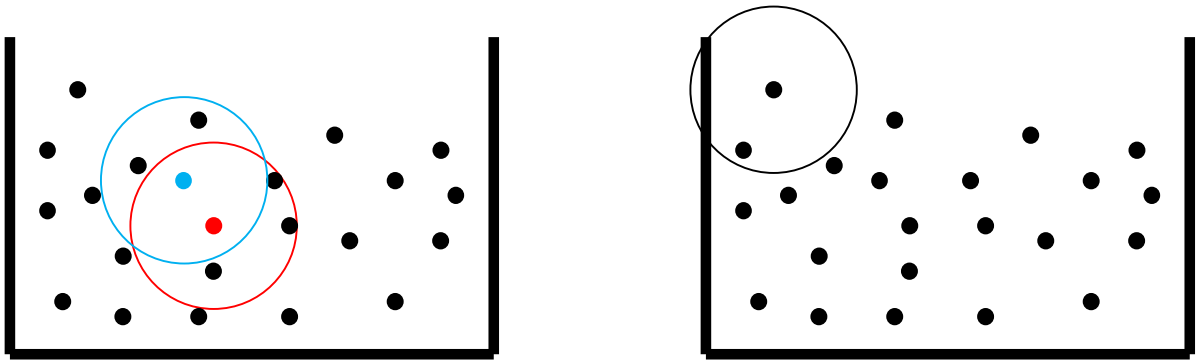
One type of implementation

$$\left\langle -\frac{1}{\rho}\nabla p \right\rangle_i \approx \sum_j P_{ij} \nabla W(r_{ij}) \qquad P_{ij} = -\frac{m_j}{\rho_j}\left(\frac{p_i}{\rho_i^{\,2}} + \frac{p_j}{\rho_j^{\,2}} \right)$$

$$\left\langle \frac{\mu}{\rho}\nabla^2 \mathbf{v} \right\rangle_i \approx \sum_j \mathbf{V}_{ij} \nabla^2 W(r_{ij}) \qquad \mathbf{V}_{ij} = \mu \frac{m_j}{\rho_j}\left(\frac{\mathbf{v}_i}{\rho_i^{\,2}} + \frac{\mathbf{v}_j}{\rho_j^{\,2}} \right)$$

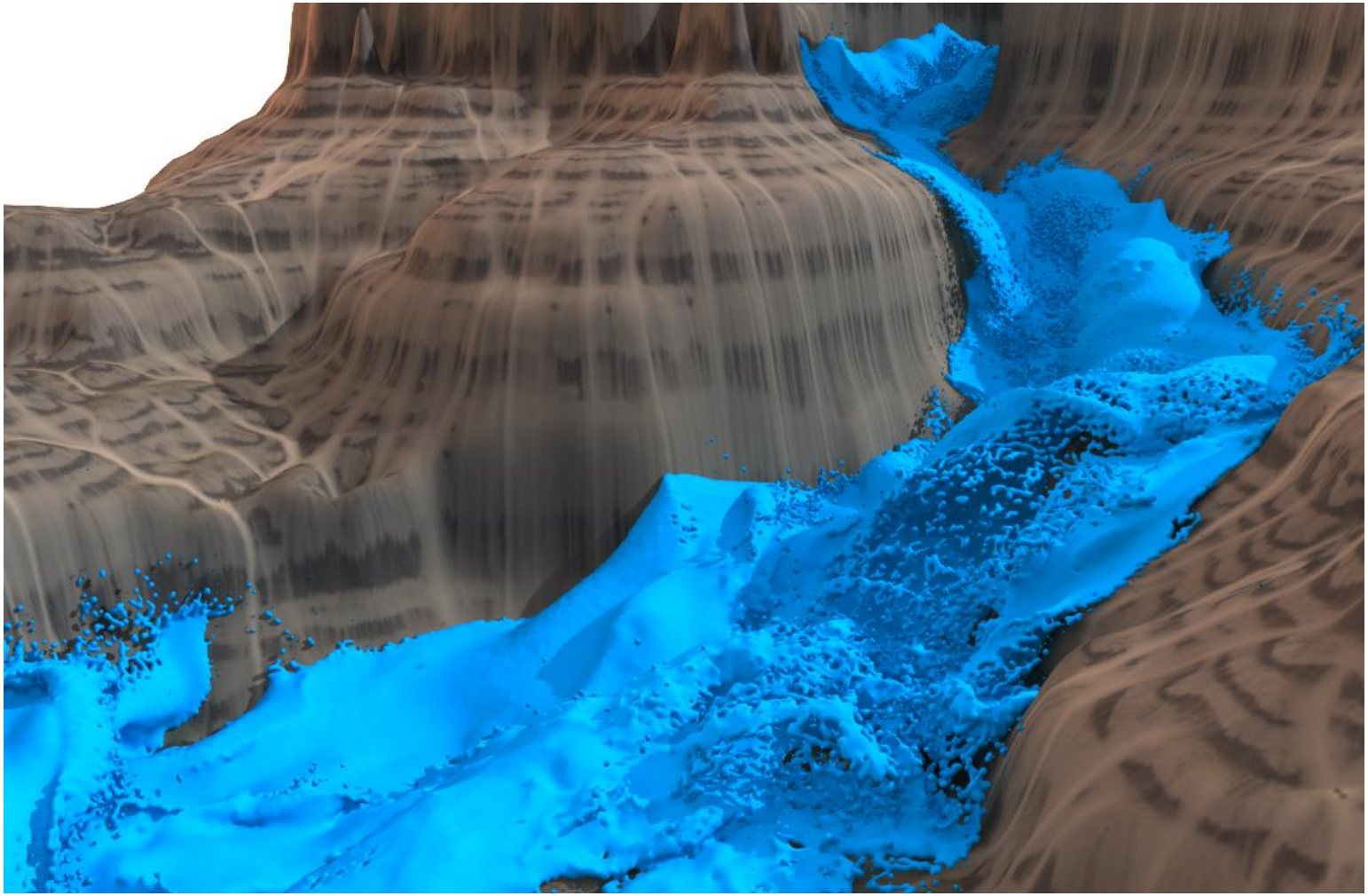
Surface detection

- We can find the surface by monitoring the density
- If the density at a particle deviates too much compared to expected density we tag it as a surface particle
- To model a fluid surface correctly we need to add a surface tension term in Navier-Stokes and take a SPH-average of it



Boundary conditions

- A simple way to implement boundary conditions is to add static SPH-particles on the rigid boundaries
- Such approach makes it very easy to simulate complicated free surface scenarios



Comparison of FV and SPH methods

Finite Volume Method (FVM)

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{u}$$

1. Divide the domain into finite control volumes (CV)
2. Integrate relevant PDEs over CVs and apply divergence theorem
3. Make approximations on surface and volume integrations
4. Make approximations on field variable interpolation
5. Collect terms and state a matrix problem

Smoothed Particle Hydrodynamics (SPH)

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{u}$$

1. Divide the domain into finite number of movable particles
2. Multiply relevant PDEs by a Weight-function (Kernel) and integrate
3. Make approximations on integrals, different terms may use different approximations on the Kernel
4. Collect terms and state a matrix problem

Discrete Vortex Method (DVM)

Navier-Stokes

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2\mathbf{u}$$

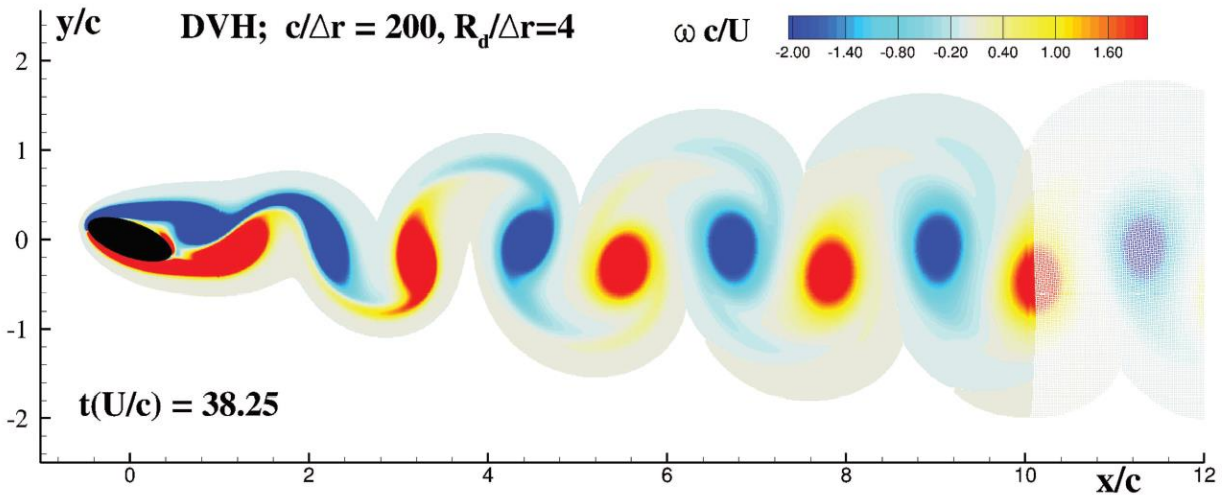
Vorticity equation

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \frac{\mu}{\rho}\nabla^2\boldsymbol{\omega} \qquad \left(\boldsymbol{\omega} \equiv \nabla\times\mathbf{u}\right)$$

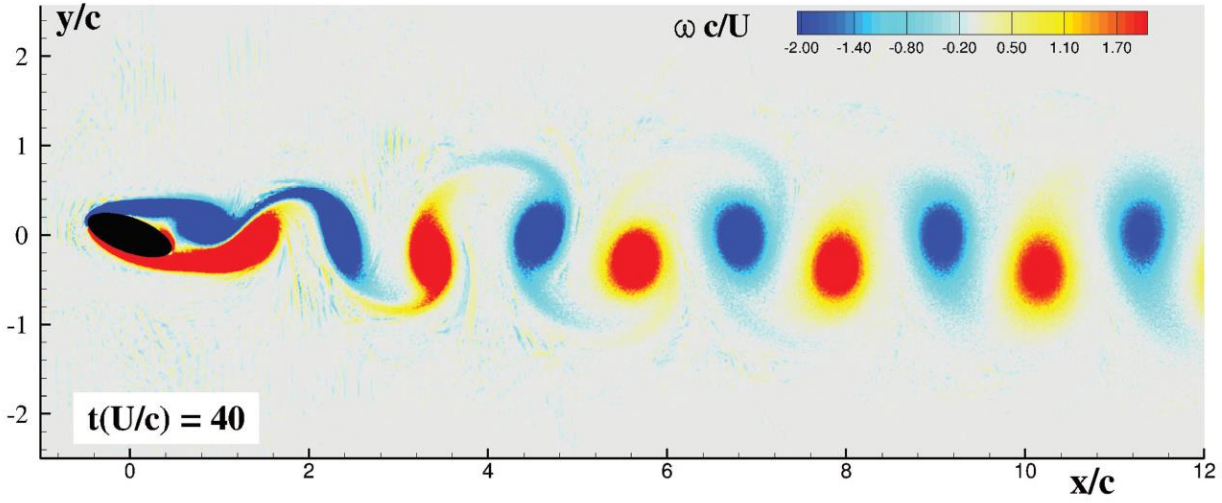
- In DVM the fluid is discretized using a finite number of moving vortex particles and the method is thus meshless.
- Remember that vorticity is a measure the local rotation of the fluid, and at boundaries with no-slip conditions vortex elements are created and convected with local flow. The vortices then interacts and exchange momentum.
- The fluid velocity is thus represented by three parts:
  - Free-stream contribution
  - Solid body contribution
  - Vortex-vortex contribution

- Historically the method was first used by Rosehead (1932) and Westwater (1936) on vorticity layers.
- They noted that few particles gave better results than many, contradicting that finer ”mesh” should give better solution.
- The problem was the center singularity in the potential vortex solution they used. Solution, use vortex with center core.

Flow around inclined elliptical cylinder: DVM vs SPH



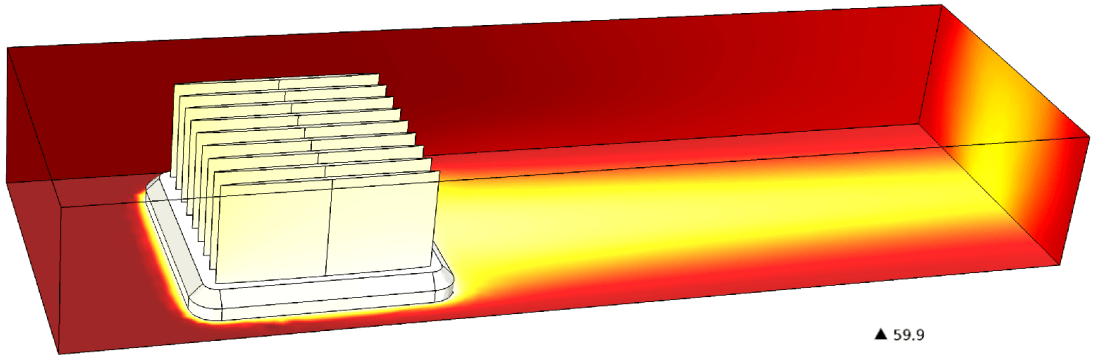
Discrete Vortex Method



Smoothed Particle Hydrodynamic

Finite Element Method

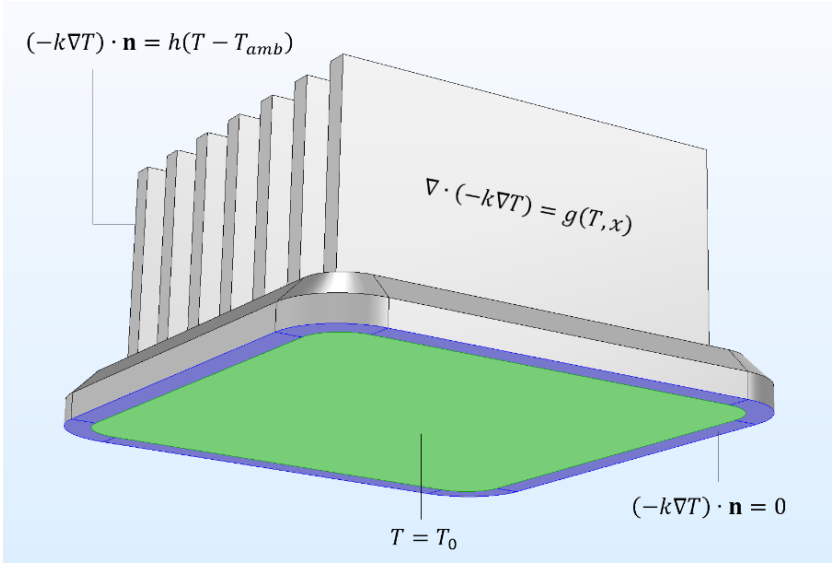
Material below is borrowed from and inspired by www.comsol.com/multiphysics/fea-software

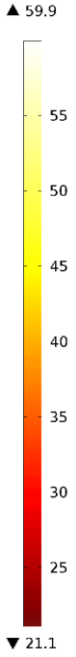


Cool air flowing from left to right over a heat sink.

$$\nabla \cdot \left(-k\nabla T\right) = g(T, \mathbf{x})$$

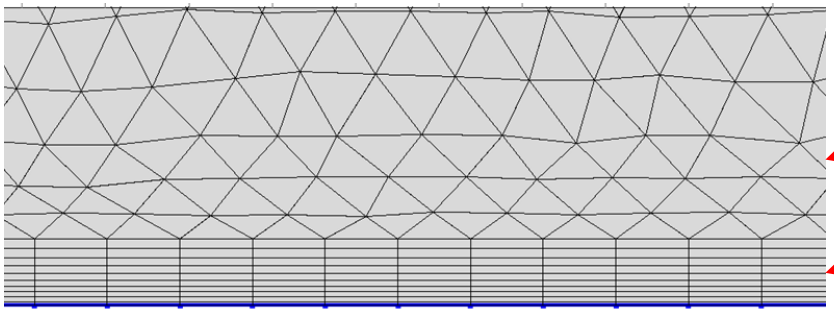
+ suitable boundary conditions



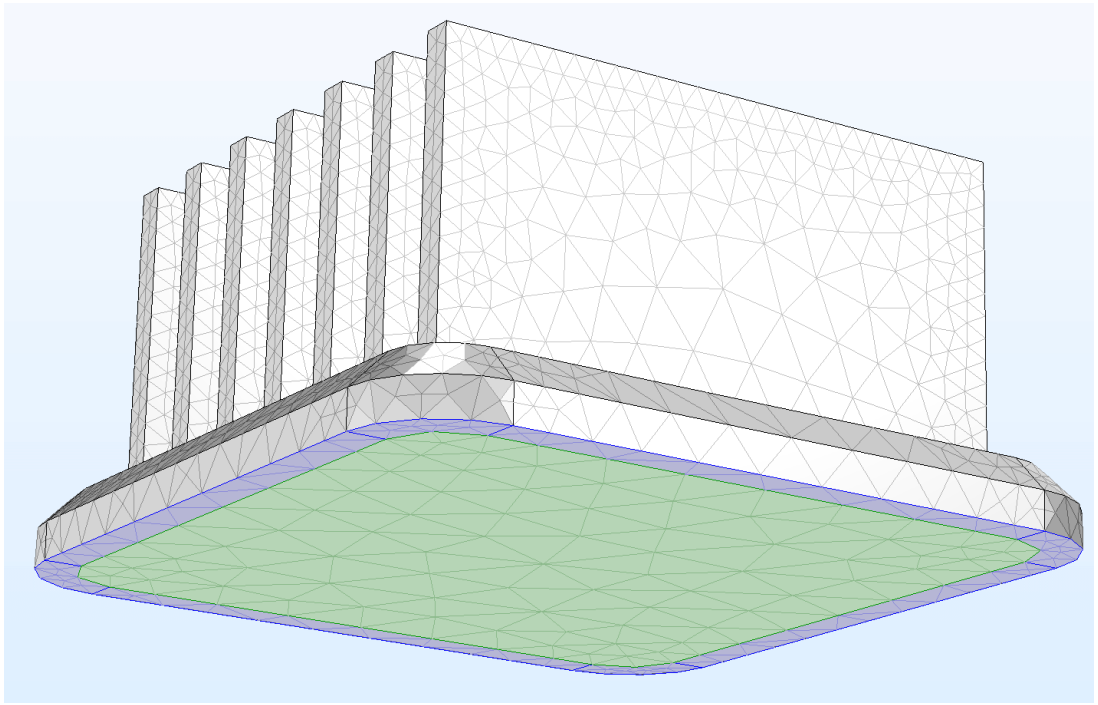


Note:
For those of you that are new to FEM or want to refresh your knowledge I have put together reading instructions for some nice and simple tutorial links.

Discretization of the domain



Exampel: Channel flow, lower boundary



If T is the variable we want to find then we denote its approximation by

$$T_h = \sum_i T_i \psi_i$$

T_i = coefficients (unknown)

ψ_i = basis functions (known)

How do we find the coefficients?

Finite Element Method

FEM recipe $\nabla \cdot \left(-k\nabla T\right) = g(T, \mathbf{x})$

1) Multiply the governing equation by a test-function and integrate over the domain:

$$\int_\Omega \varphi \nabla \cdot \left(-k\nabla T\right) dV = \int_\Omega \varphi g(T, \mathbf{x}) dV$$

2) Integrate by parts (Green´s first identity):

$$\int_\Omega k\nabla T \cdot \nabla \varphi dV + \int_{\partial\Omega} \left(-k\nabla T\right) \cdot \mathbf{n} \varphi dS = \int_\Omega \varphi g(T, \mathbf{x}) dV$$

The derivative is moved to the known test-function

3) Use the basis functions as test functions (Galerkin-method).

$$T \approx T_h = \sum_i T_i \psi_i$$

$$\sum_i T_i \int_\Omega k\nabla \psi_i \cdot \nabla \psi_j dV + \sum_i T_i \int_{\partial\Omega} \left(-k\nabla \psi_i\right) \cdot \mathbf{n} \psi_j dS = \int_\Omega g\left(\sum_i T_i \psi_i\right) \psi_j dV$$

Known Known Known

T_i = coefficients (unknown), ψ_i = basis functions (known)

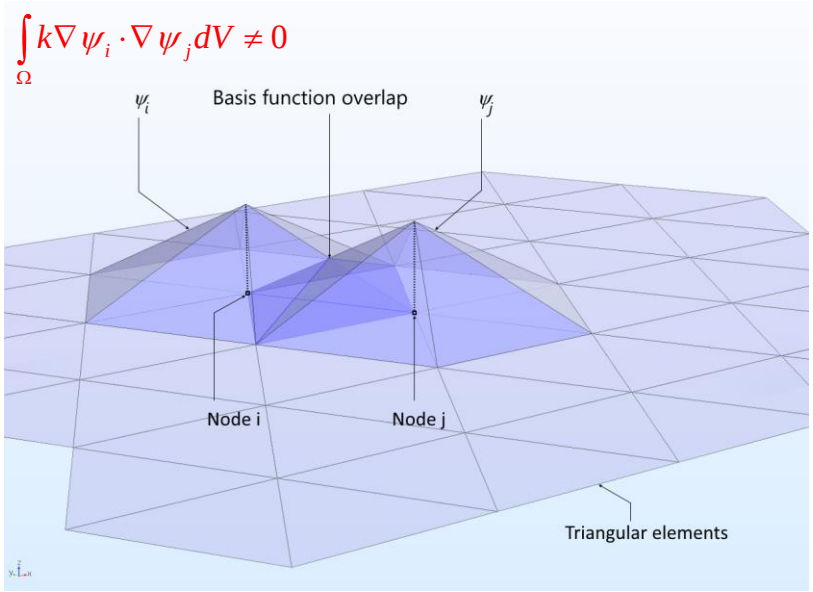
$$\sum_i T_i \int_\Omega k\nabla\psi_i \cdot \nabla\psi_j dV + \sum_i T_i \int_{\partial\Omega} \left(-k\nabla\psi_i\right)\cdot \mathbf{n}\psi_j dS = \int_\Omega g\left(\sum_i T_i\psi_i\right)\psi_j dV$$

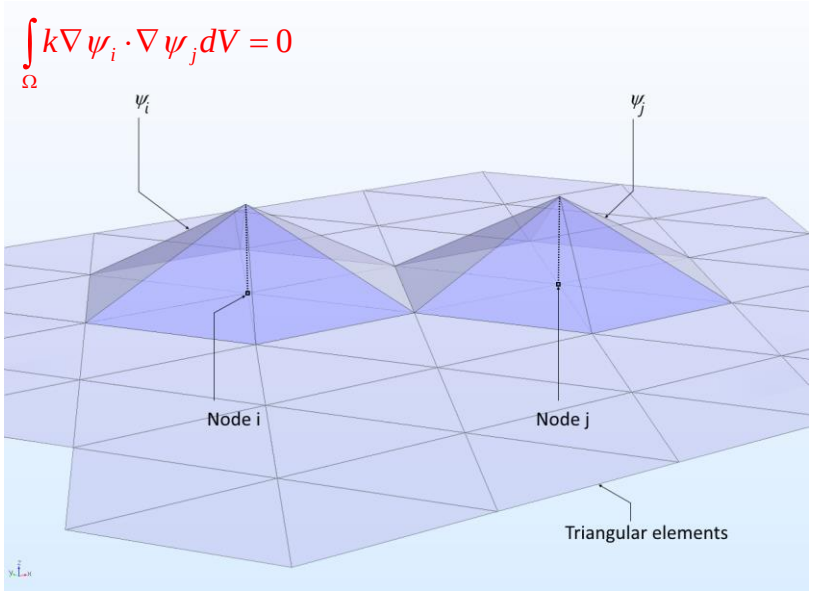
Known Known Known

The integral equation can be stated in a matrix form where the elements in the matrix consists of combinations of the known basis functions:

$$A_{ij}T_i = b_j$$

5) Select basis functions that are zero everywhere except at in small regions. This will make the integrals non-zero only at regions where there are overlap between the basis functions ψ_i and ψ_j





Element order

www.comsol.com/blogs/keeping-track-of-element-order-in-multiphysics-models/

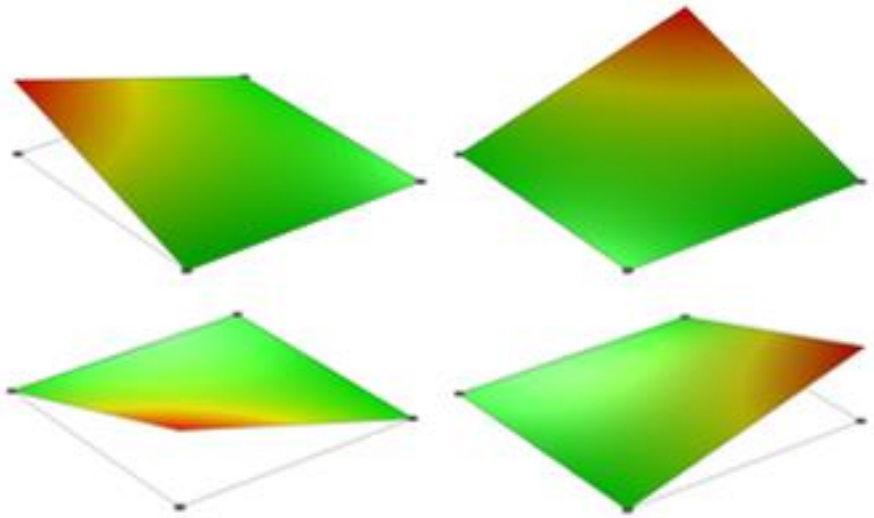
Shape functions = Basis functions

$$T \approx T_h = \sum_i T_i \psi_i$$

T_i = coefficients (unknown)

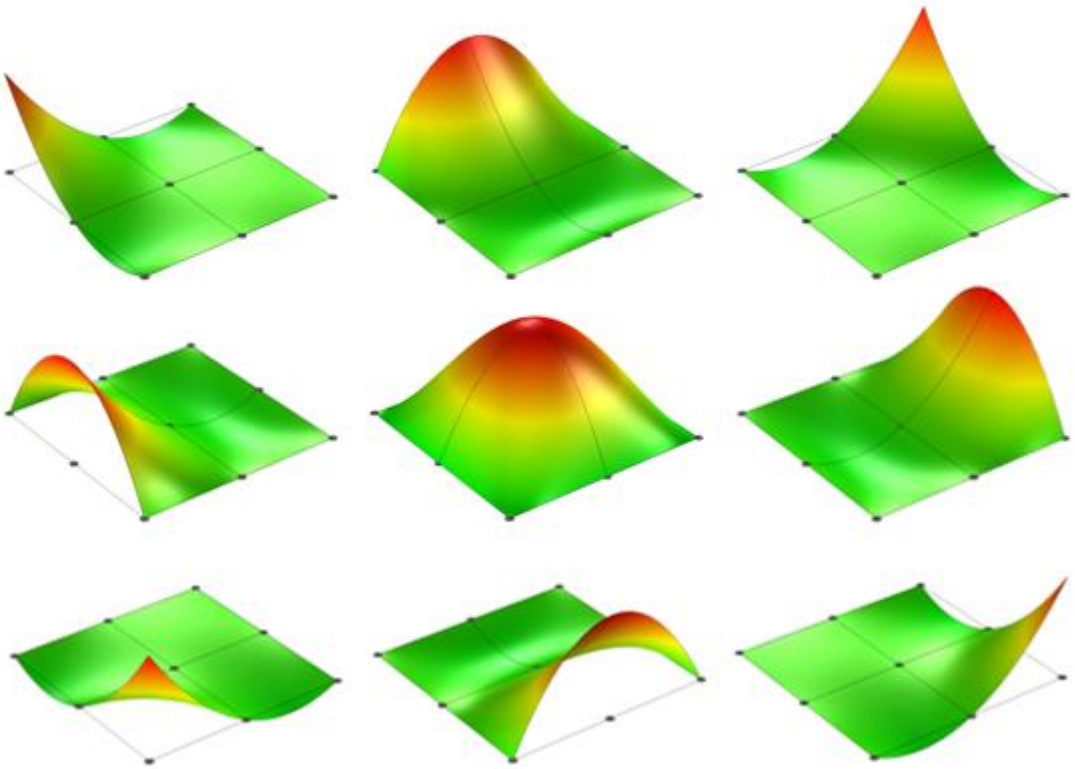
ψ_i = basis functions (known)

Shape functions for
first-order square quadrilateral Lagrange element



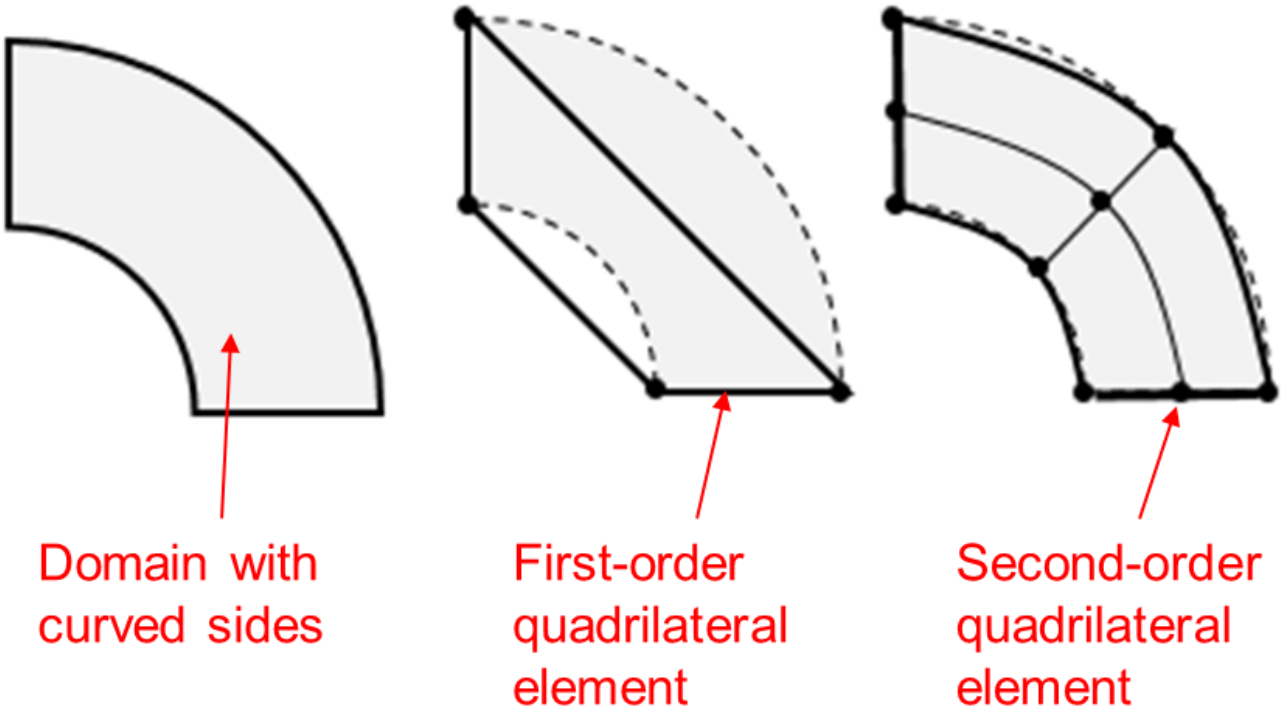
All shape functions are unity at one node and zero at all of the others

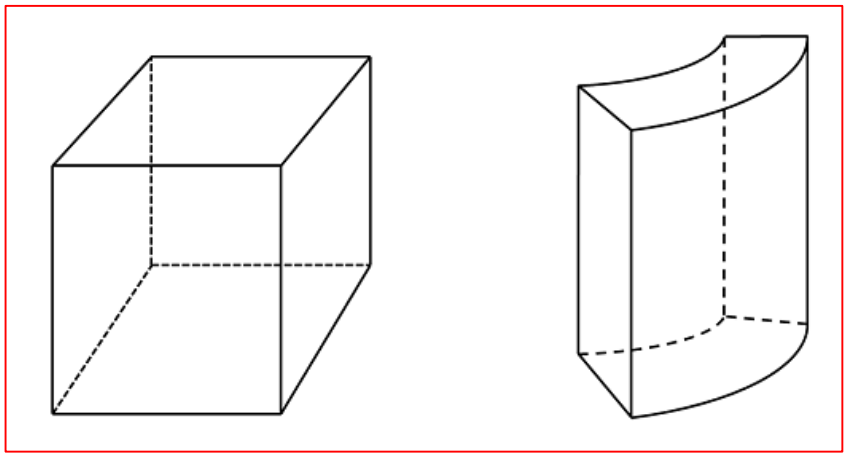
Shape functions for
second-order square quadrilateral Lagrange element

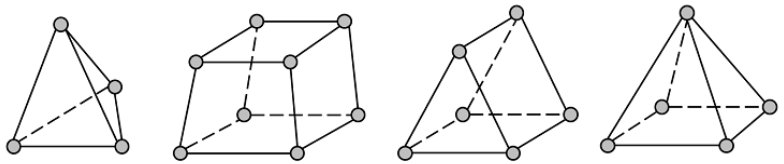


Unity at one node and
zero at all of the others

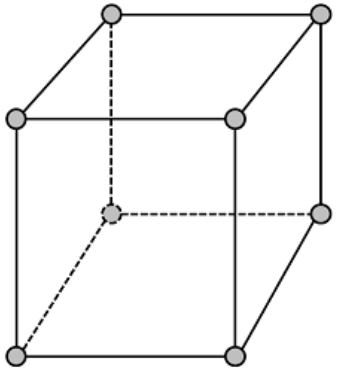
Geometric discretization error

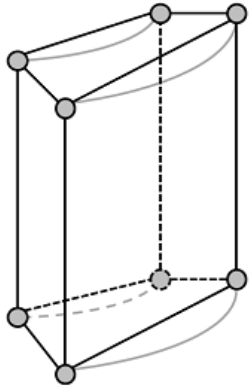




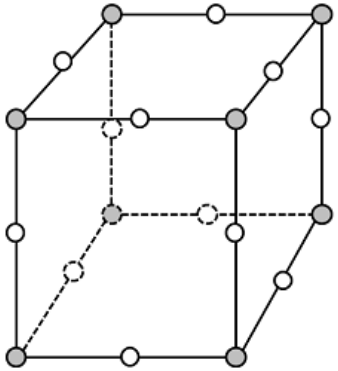


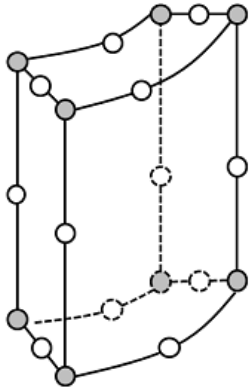
First order elements



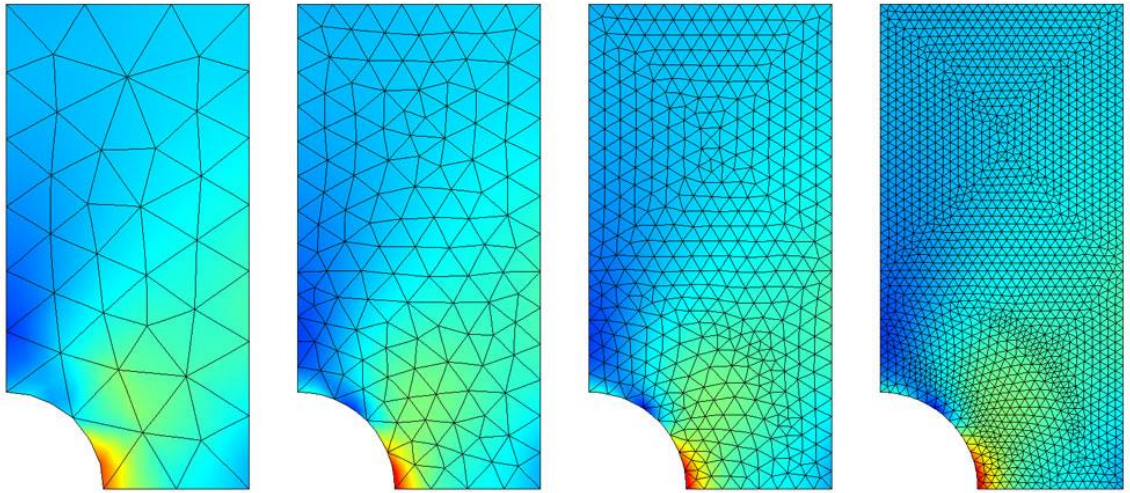


Second order elements

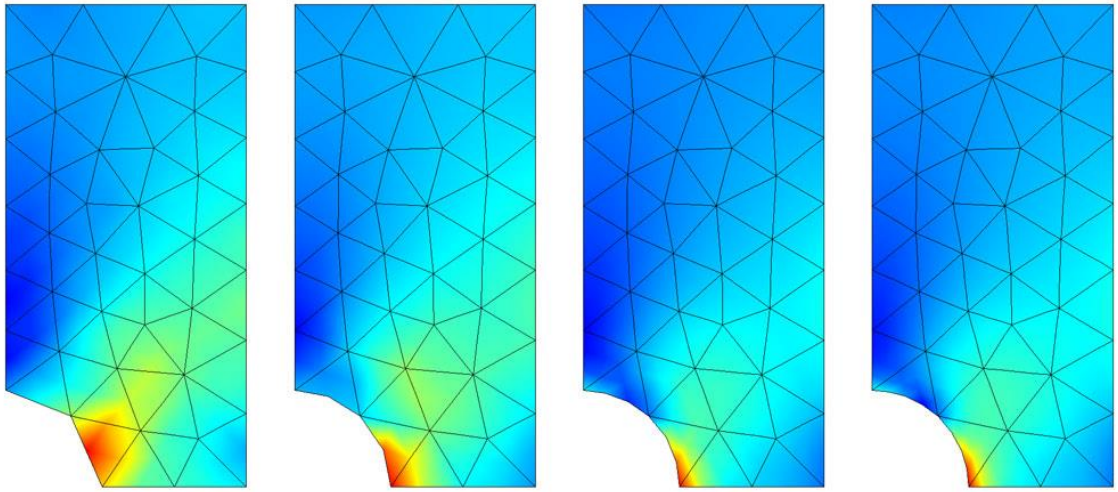




Example: Stress on a plate



Mesh refinements using fixed element order.



Increasing element order with fixed mesh size.

Degrees of Freedom

DoF = NrNodes x NrVariables ← u, v, k, ε

NrNodes = ElementFactor x NrElements

From Comsol Knowledgebase-875

"The relation between the number of nodes and the number of elements depends on the order of the elements and differs between 2D and 3D. The relation is only approximate since it depends on the ratio of the elements that lie on the boundary of the geometry. For thin geometries, where a large proportion of the elements lie on the boundary, the number of nodes per element is a bit higher."

From Comsol Knowledgebase-875

2D

Linear triangular elements: (#nodes) = 0.5 * (#elements)
Linear quad elements: (#nodes) = 1 * (#elements)
Quadratic triangular elements: (#nodes) = 2 * (#elements)
Quadratic quad elements: (#nodes) = 4 * (#elements)
Cubic triangular elements: (#nodes) = 4.5 * (#elements)
Cubic quad elements: (#nodes) = 9 * (#elements)

3D

Linear tetrahedral elements: (#nodes) = 0.2 * (#elements)
Linear brick elements: (#nodes) = 1.2 * (#elements)
Quadratic tetrahedral elements: (#nodes) = 1.4 * (#elements)
Quadratic brick elements: (#nodes) = 8.5 * (#elements)
Cubic tetrahedral elements: (#nodes) = 4.6 * (#elements)
Cubic brick elements: (#nodes) = 28 * (#elements)

Example: 2D vs 3D

From Comsol Knowledgebase-875

2D

Linear triangular elements: (#nodes) = 0.5 * (#elements)

Linear quad elements: (#nodes) = 1 * (#elements)

Quadratic triangular elements: (#nodes) = 2 * (#elements)

Quadratic quad elements: (#nodes) = 4 * (#elements)

Cubic triangular elements: (#nodes) = 4.5 * (#elements)

Cubic quad elements: (#nodes) = 9 * (#elements)

3D

Linear tetrahedral elements: (#nodes) = 0.2 * (#elements)

Linear brick elements: (#nodes) = 1.2 * (#elements)

Quadratic tetrahedral elements: (#nodes) = 1.4 * (#elements)

Quadratic brick elements: (#nodes) = 8.5 * (#elements)

Cubic tetrahedral elements: (#nodes) = 4.6 * (#elements)

Cubic brick elements: (#nodes) = 28 * (#elements)

2/0.5=4 times more number of nodes
=> 4 times more DoF

1.4/0.2=7 times more number of nodes
=> 7 times more DoF

End of lecture