Finite Difference and Finite Element Methods

Lars Ole Schwen
Fraunhofer Institute for Medical Image Computing MEVIS, Bremen

Based on a presentation by Tobias Preusser

EMBC 2015 Workshop on
Practical Computer Modeling for Medical Device Development

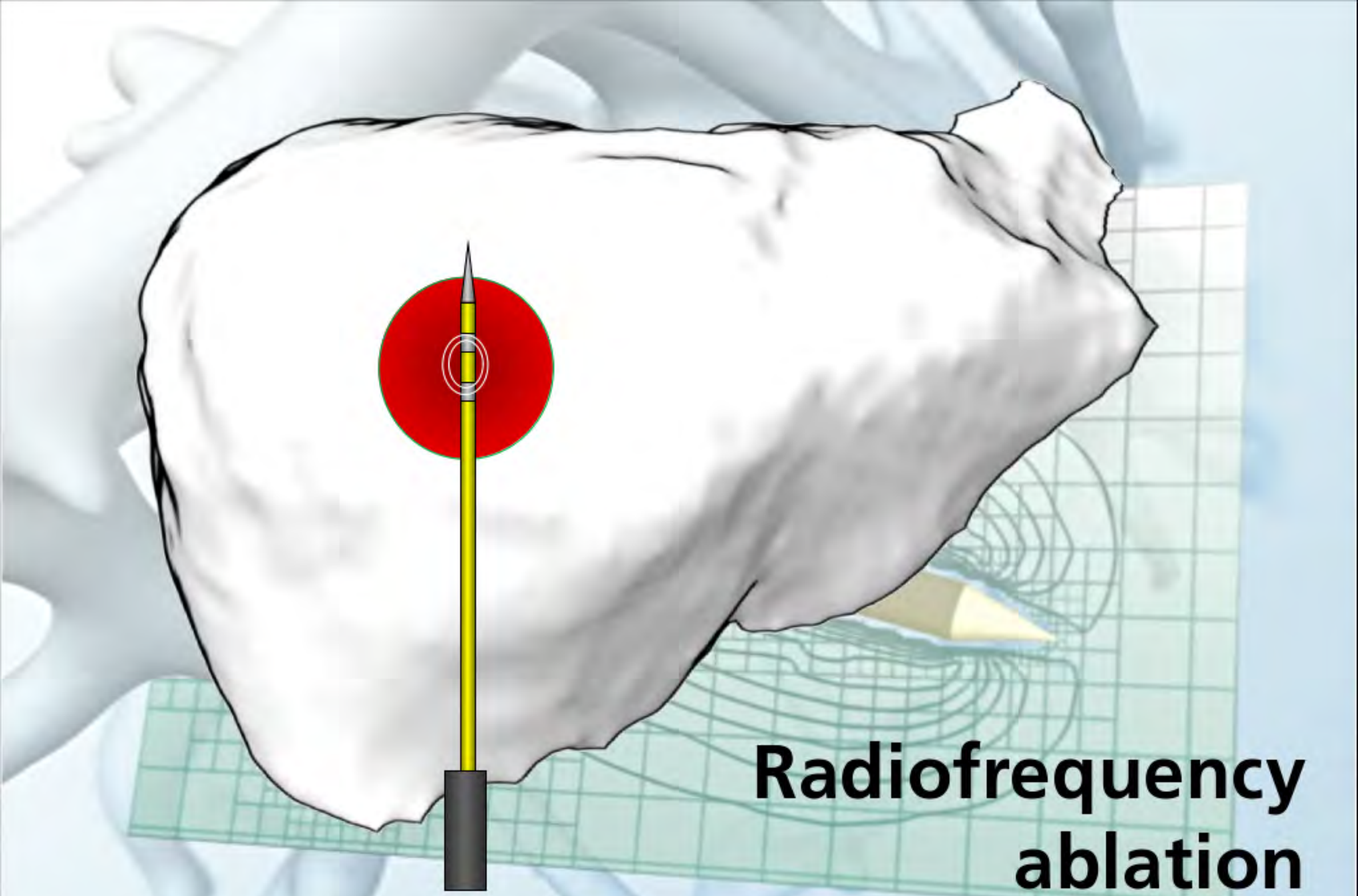
Milan, 2015-08-25

Contents

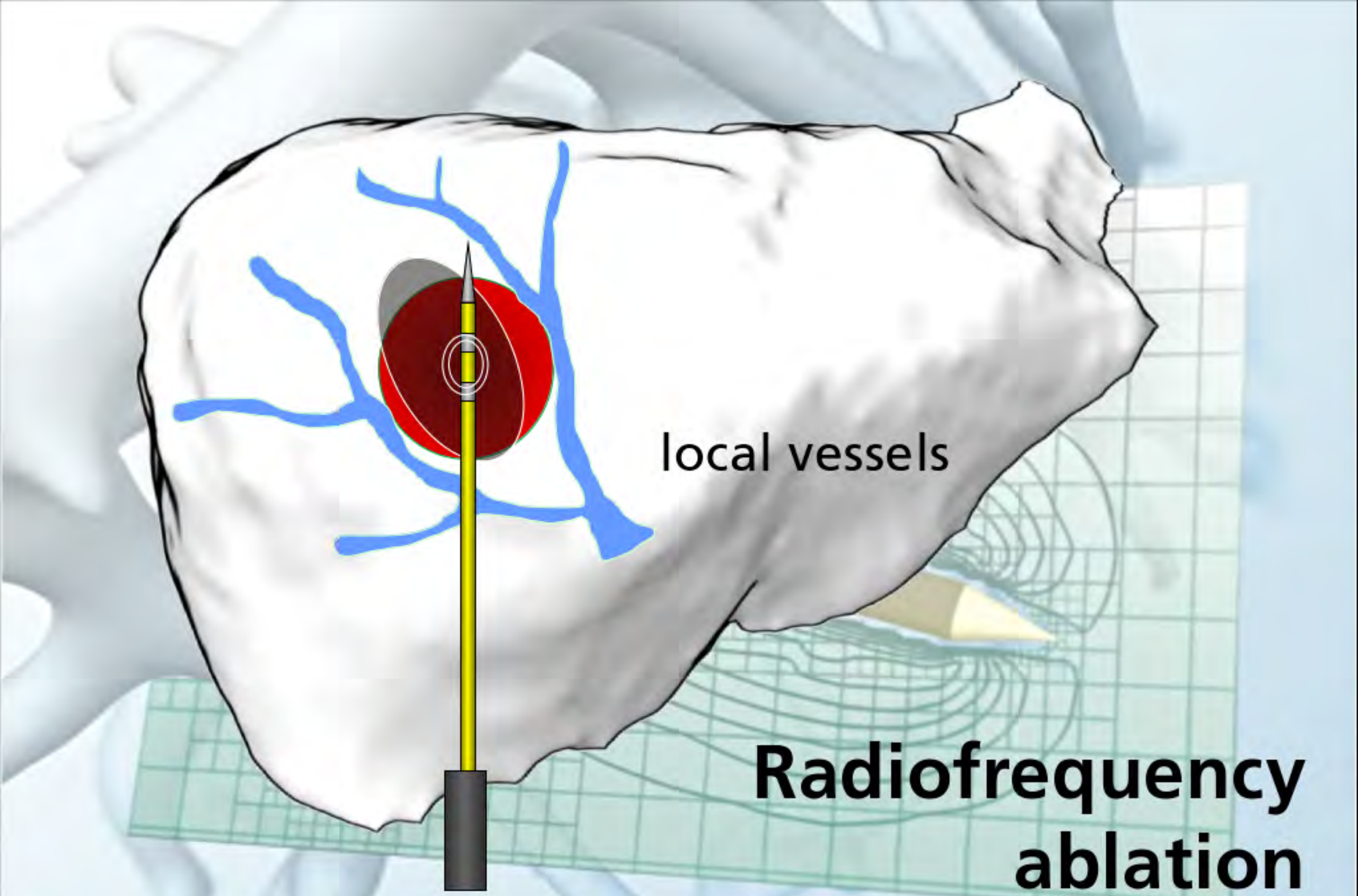
1

Modeling Thermal Ablation

Radiofrequency ablation

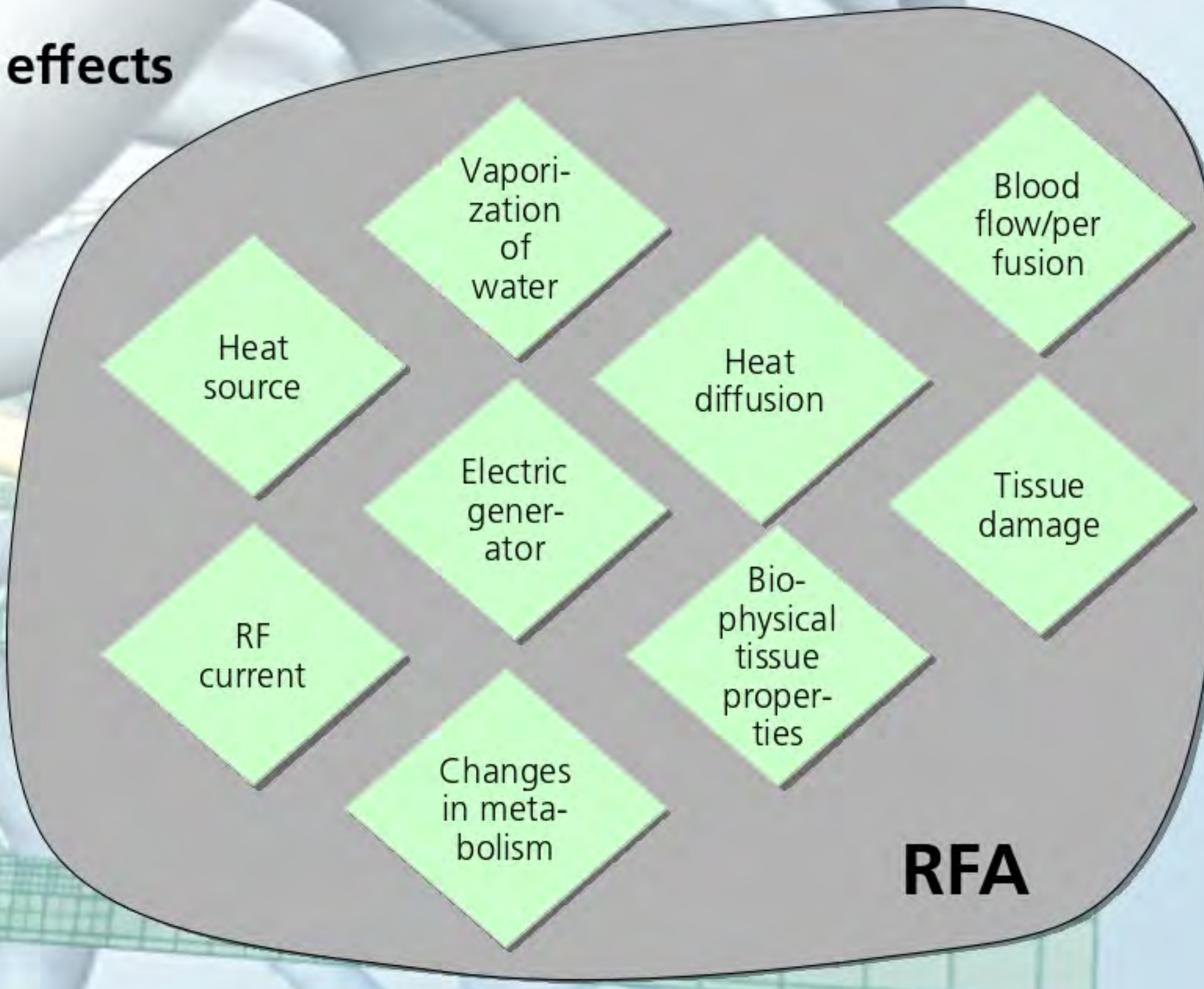


Radiofrequency ablation

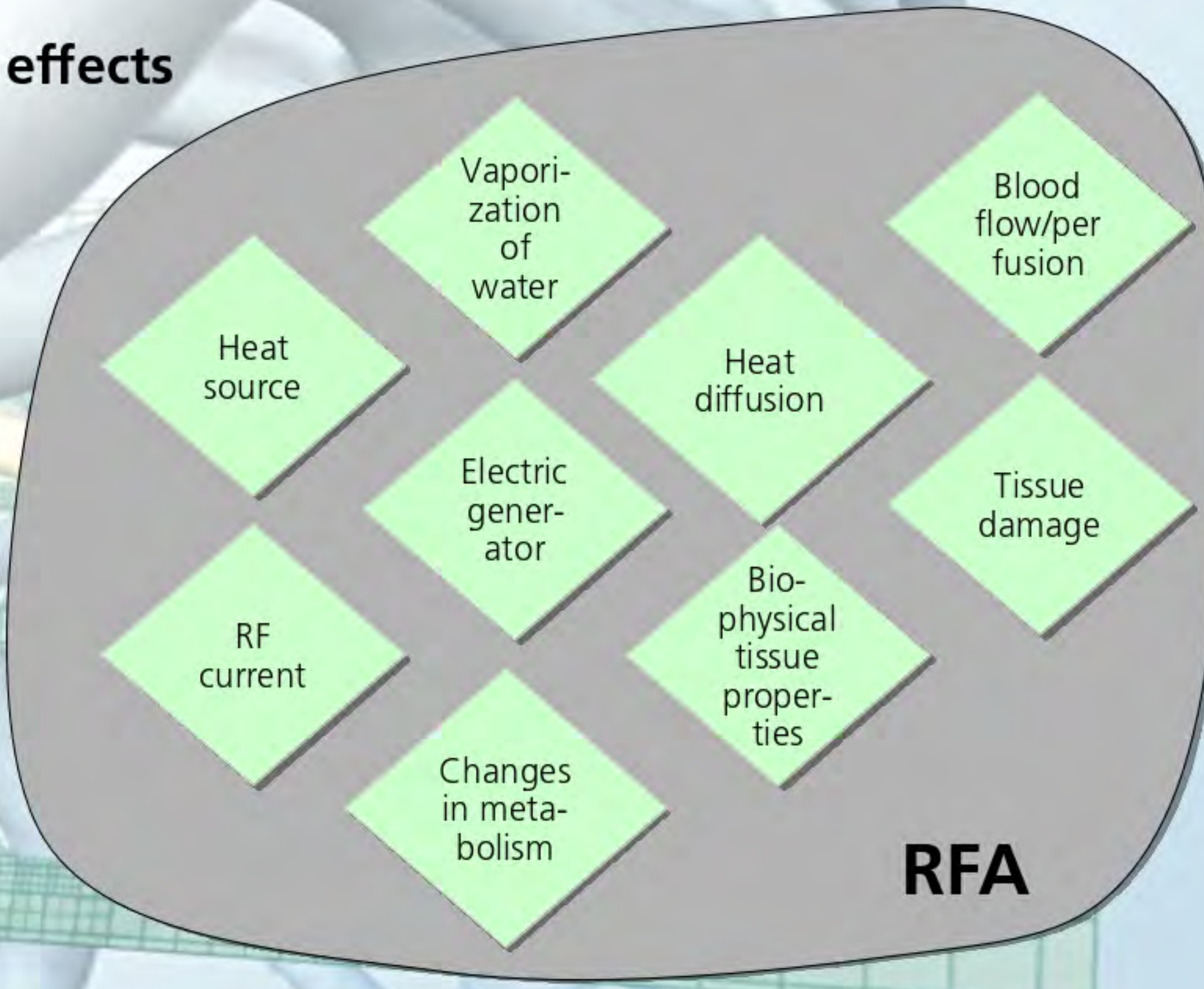


Radiofrequency ablation

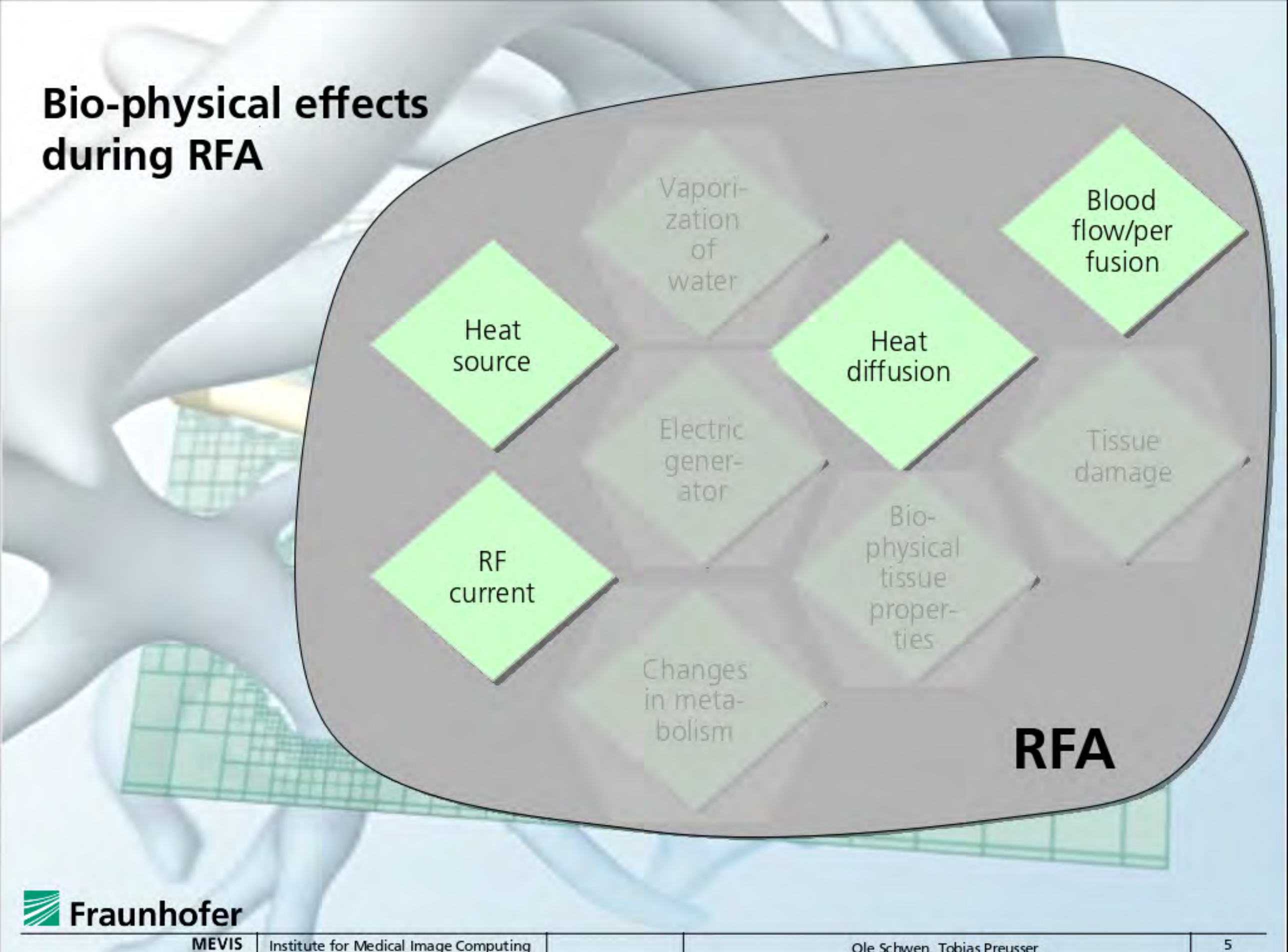
Bio-physical effects during RFA

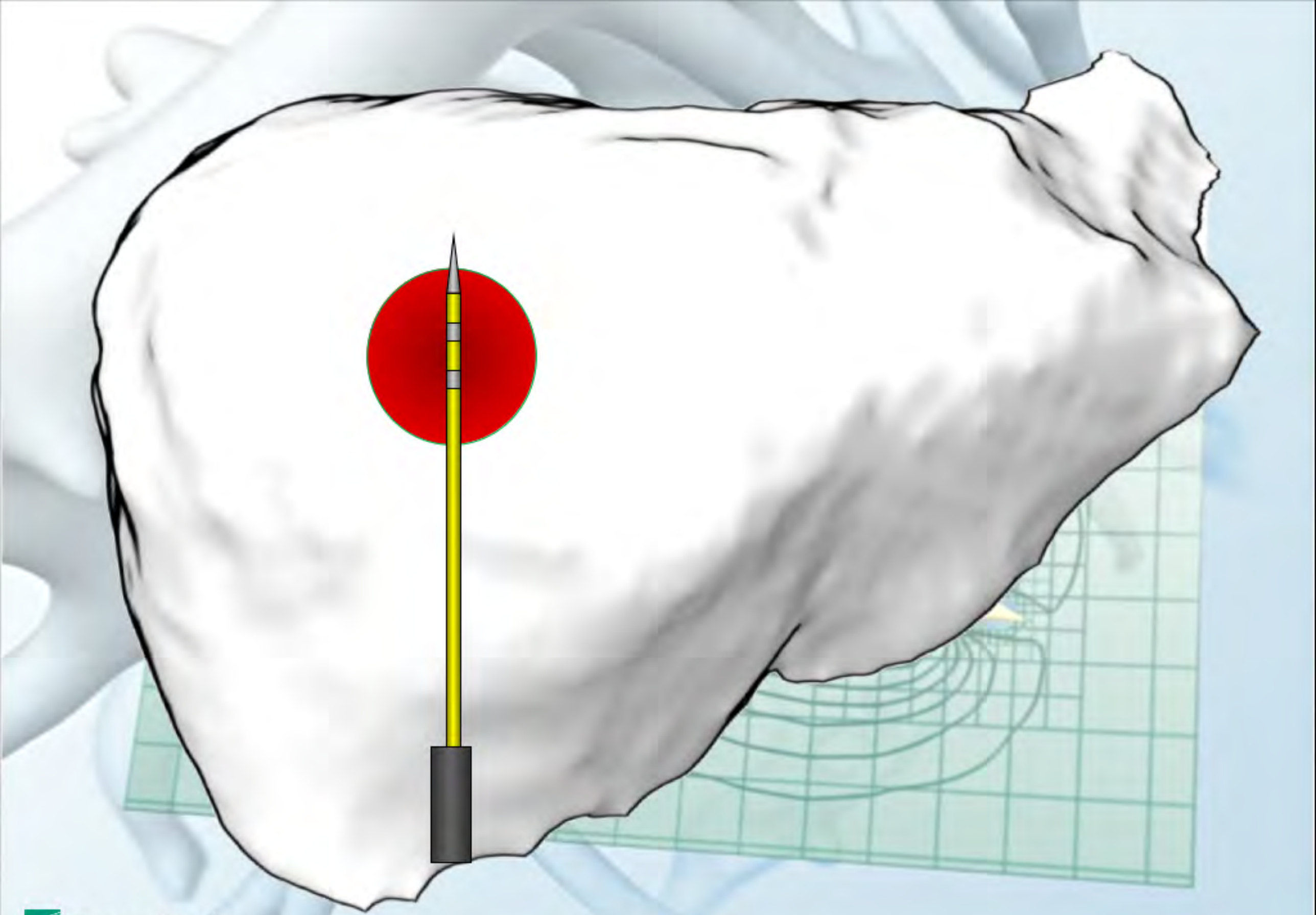


Bio-physical effects during RFA



Bio-physical effects during RFA

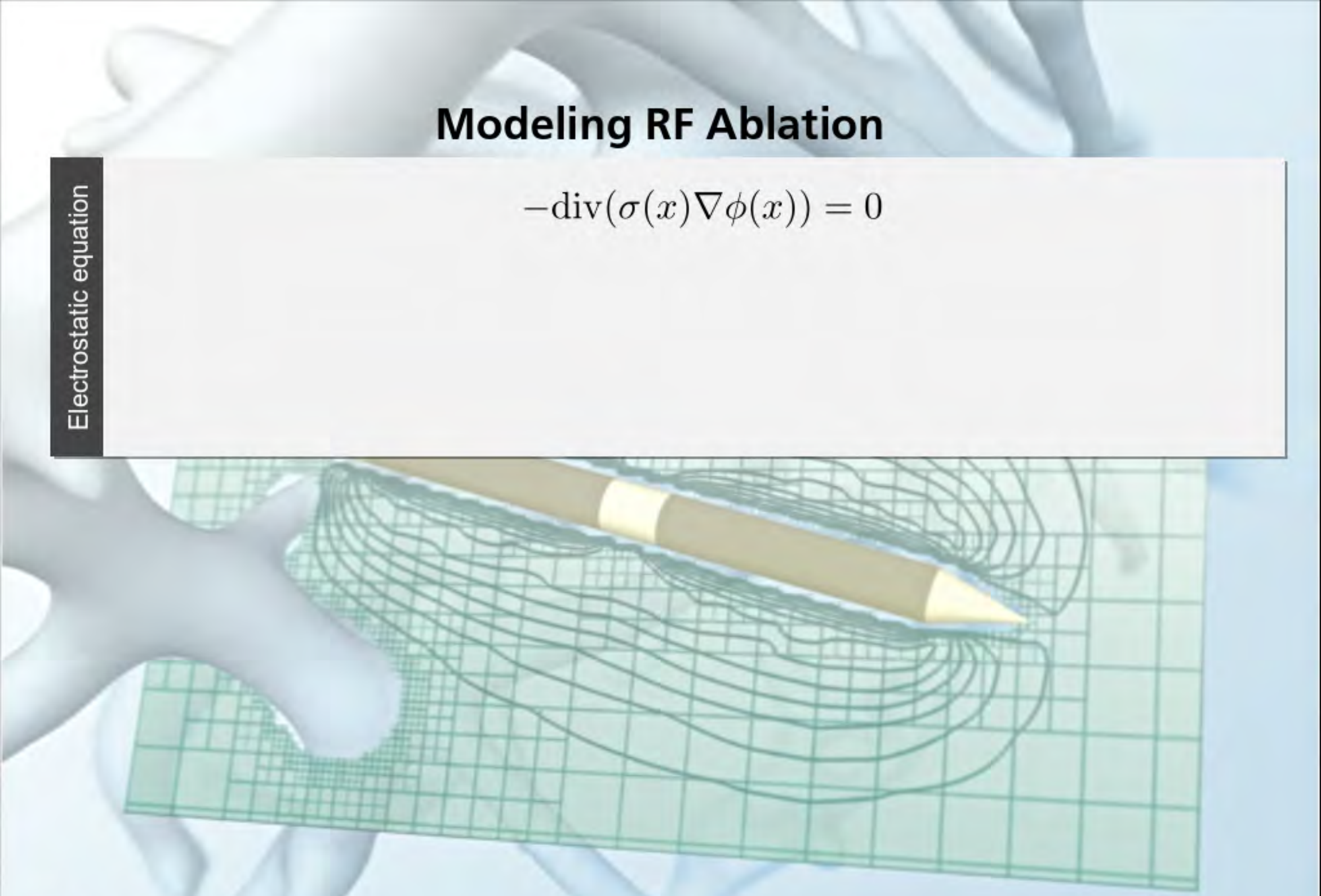




Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$



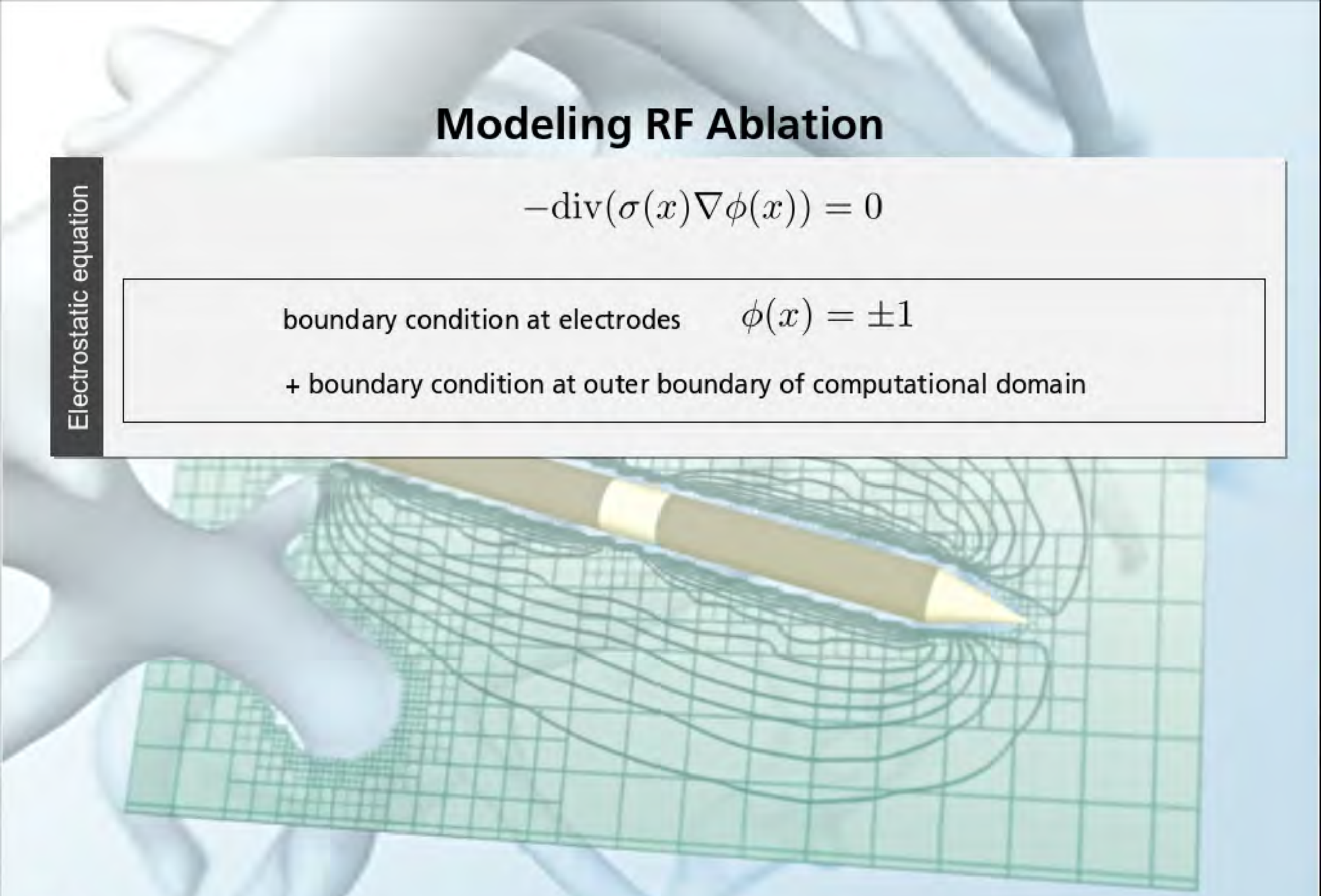
Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

boundary condition at electrodes $\phi(x) = \pm 1$

+ boundary condition at outer boundary of computational domain



Modeling RF Ablation

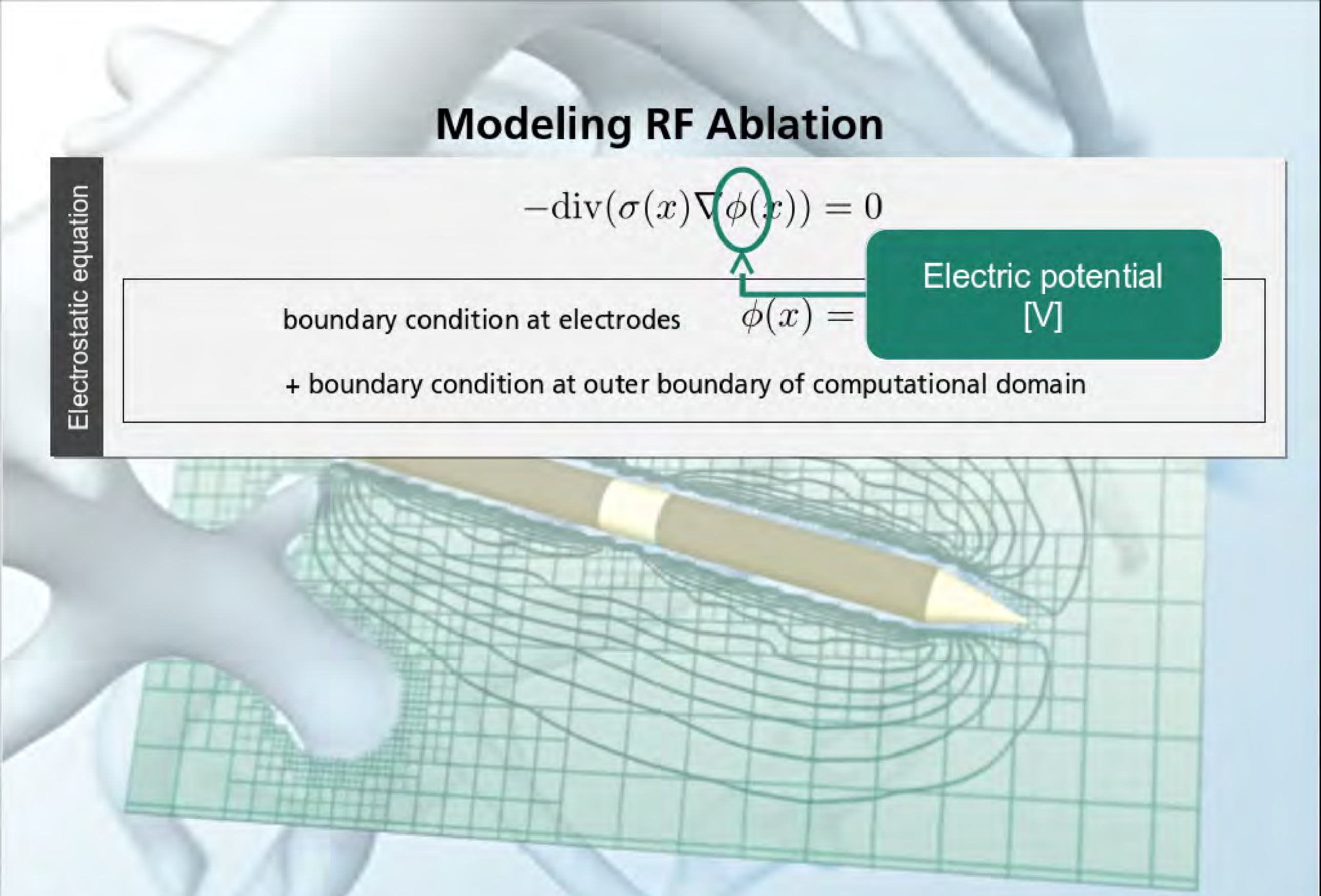
Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

boundary condition at electrodes $\phi(x) =$

Electric potential [V]

+ boundary condition at outer boundary of computational domain



Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

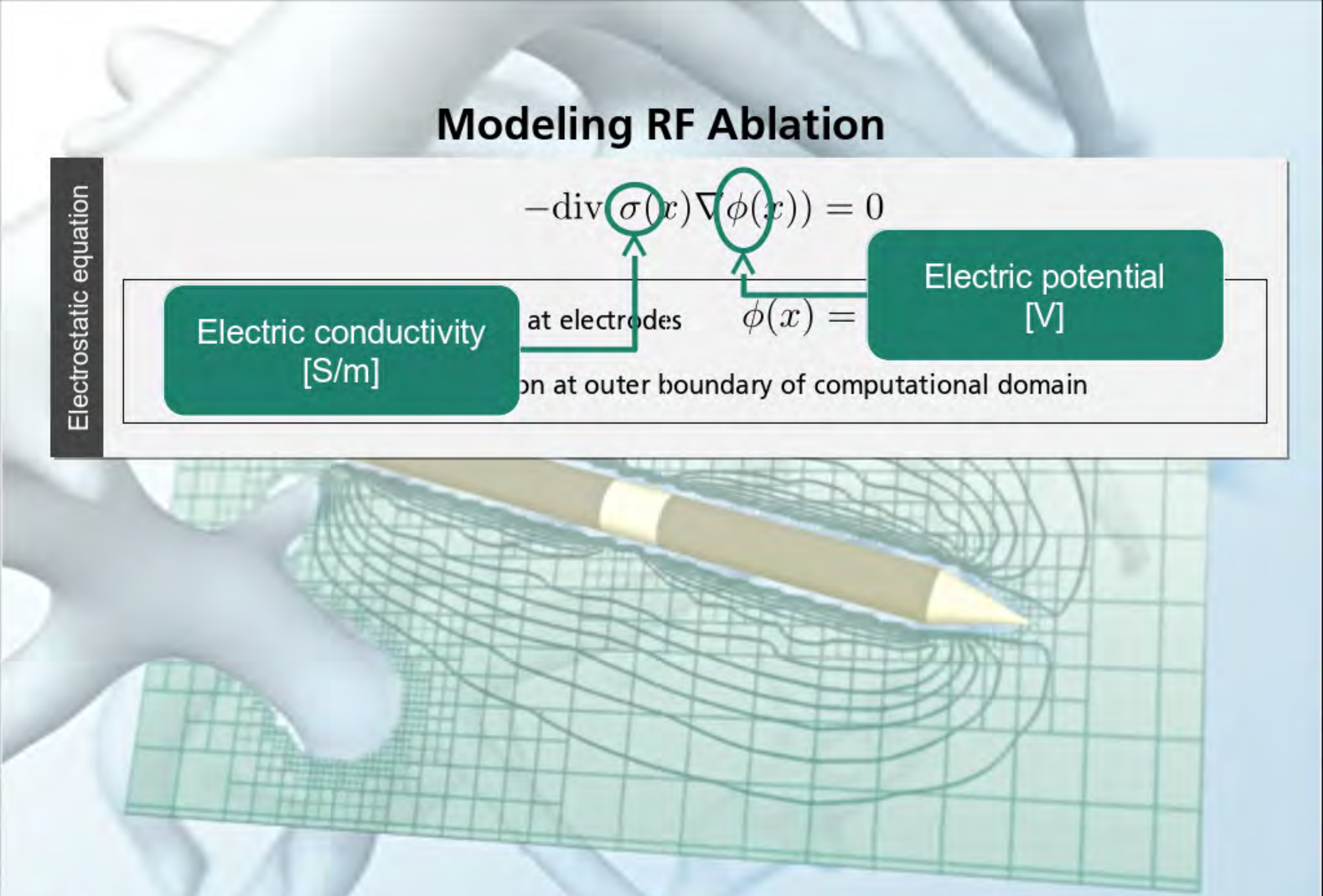
Electric conductivity [S/m]

at electrodes

$\phi(x) =$

Electric potential [V]

on at outer boundary of computational domain



Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

Electric conductivity [S/m]

at electrodes

$\phi(x) =$

Electric potential [V]

on at outer boundary of computational domain

Bioheat transfer equation

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

Electric conductivity [S/m]

at electrodes

$\phi(x) =$

Electric potential [V]

on at outer boundary of computational domain

Bioheat transfer equation

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

boundary condition at applicator $T(t,x) = T_\mathrm{b}$

+ boundary condition at outer boundary of computational domain

+ initial condition

Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

Electric conductivity [S/m]

at electrodes $\phi(x) =$

Electric potential [V]

on at outer boundary of computational domain

Bioheat transfer equation

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

boundary condition at applicator $T(t,x) = T_\mathrm{b}$

+ boundary condition at outer boundary of computational domain

+ initial condition

Temperature [K]

Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

at electrodes $\phi(x) =$

on at outer boundary of computational domain

Electric conductivity [S/m]

Electric potential [V]

Bioheat transfer equation

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

boundary condition at applicator $T(t,x) = T_\mathrm{b}$

+ boundary condition at outer boundary of computati

+ initial condition

Temperature [K]

Relative perfusion rate [J/s m³ K]

Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

at electrodes $\phi(x) =$

on at outer boundary of computational domain

Electric conductivity [S/m]

Electric potential [V]

Body temperature [K]

Bioheat transfer equation

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

boundary condition at applicator $T(t,x) = T_\mathrm{b}$

+ boundary condition at outer boundary of computati

+ initial condition

Temperature [K]

Relative perfusion rate [J/s m³ K]

Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

at electrodes $\phi(x) =$

on at outer boundary of computational domain

Electric conductivity [S/m]

Electric potential [V]

Body temperature [K]

Electrostatic equation

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

boundary condition at applicator $T(t,x) = T_\mathrm{b}$

n at outer boundary of computati

Heat capacity [J/kg K]

Temperature [K]

Relative perfusion rate [J/s m³ K]

Modeling RF Ablation

Electrostatic equation

$$-\operatorname{div}(\sigma(x)\nabla\phi(x)) = 0$$

at electrodes $\phi(x) =$

on at outer boundary of computational domain

Electric conductivity [S/m]

Electric potential [V]

Density [kg/m³]

Body temperature [K]

sfer equation

$$\rho(x)c(x)\partial_t T(t,x) - \operatorname{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_b)$$

boundary condition at applicator $T(t,x) = T_b$

n at outer boundary of computati

Heat capacity [J/kg K]

Temperature [K]

Relative perfusion rate [J/s m³ K]

Modeling RF Ablation

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

at electrodes $\phi(x) =$

on at outer boundary of computational domain

Electric conductivity [S/m]

Electric potential [V]

Density [kg/m³]

Heat conductivity [W/K m]

Body temperature [K]

sfer equation

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

boundary condition at applicator $T(t,x) = T_\mathrm{b}$

n at outer boundary of computati

Heat capacity [J/kg K]

Temperature [K]

Relative perfusion rate [J/s m³ K]

Differential Operators

Electrostatic equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0$$

Differential Operators

Electrostatic equation

$$-\operatorname{div}(\sigma(x)\nabla\phi(x)) = 0$$

Differential Operators

$$-\operatorname{div}(\sigma(x)\nabla\phi(x)) = 0$$

Partial derivative = directional derivative in a coordinate direction

$$u(x_1, x_2, x_3) \qquad \frac{\partial u}{\partial x_1} = \partial_{x_1} u = \text{ slope of } u \text{ in direction } x_1$$

Differential Operators

$$-\operatorname{div}(\sigma(x)\nabla\phi(x)) = 0$$

Partial derivative = directional derivative in a coordinate direction

$$u(x_1, x_2, x_3) \qquad \frac{\partial u}{\partial x_1} = \partial_{x_1} u = \text{ slope of } u \text{ in direction } x_1$$

Gradient = vector of all partial derivatives

$$\nabla u = (\partial_{x_1} u, \partial_{x_2} u, \partial_{x_3} u)^t$$

- Points in direction of steepest ascent
- Perpendicular to level lines

Differential Operators

$$-\operatorname{div}(\sigma(x)\nabla\phi(x)) = 0$$

Partial derivative = directional derivative in a coordinate direction

$$u(x_1, x_2, x_3) \qquad \frac{\partial u}{\partial x_1} = \partial_{x_1} u = \text{ slope of } u \text{ in direction } x_1$$

Gradient = vector of all partial derivatives

$$\nabla u = (\partial_{x_1} u, \partial_{x_2} u, \partial_{x_3} u)^t$$

- Points in direction of steepest ascent
- Perpendicular to level lines

Divergence = sum of partial derivatives of a vector field

$$\operatorname{div}(\nabla u) = \sum_{i=1}^{3} \partial_{x_i} (\nabla u)_i = \sum_{i=1}^{3} \partial_{x_i} (\partial_{x_i} u) = \sum_{i=1}^{3} \partial_{x_i}^2 u = \Delta u$$

Differential Operators

$$-\operatorname{div}(\sigma(x)\nabla\phi(x)) = 0$$

Partial derivative = directional derivative in a coordinate direction

$$u(x_1, x_2, x_3) \qquad \frac{\partial u}{\partial x_1} = \partial_{x_1} u = \text{ slope of } u \text{ in direction } x_1$$

Gradient = vector of all partial derivatives

$$\nabla u = (\partial_{x_1} u, \partial_{x_2} u, \partial_{x_3} u)^t$$

- Points in direction of steepest ascent
- Perpendicular to level lines

Divergence = sum of partial derivatives of a vector field

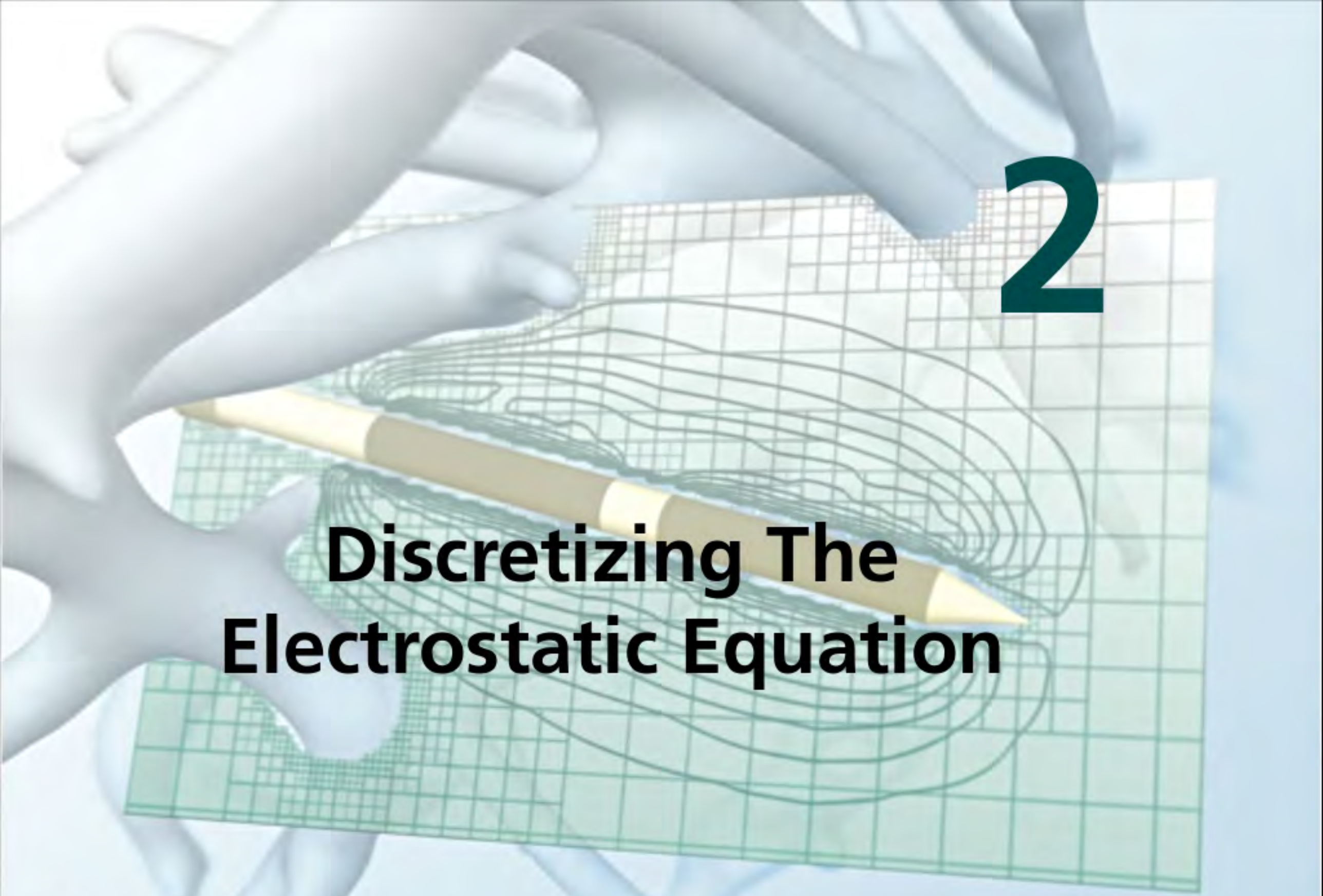
$$\operatorname{div}(\nabla u) = \sum_{i=1}^{3} \partial_{x_i} (\nabla u)_i = \sum_{i=1}^{3} \partial_{x_i} (\partial_{x_i} u) = \sum_{i=1}^{3} \partial_{x_i}^2 u = \Delta u$$

Electrostatic equation in 1D

$$-(\sigma\, \phi')' = 0$$

Electrostatic equation in 2D

$$-\partial_{x_1}(\sigma\, \partial_{x_1}\phi) - \partial_{x_2}(\sigma\, \partial_{x_2}\phi) = 0$$



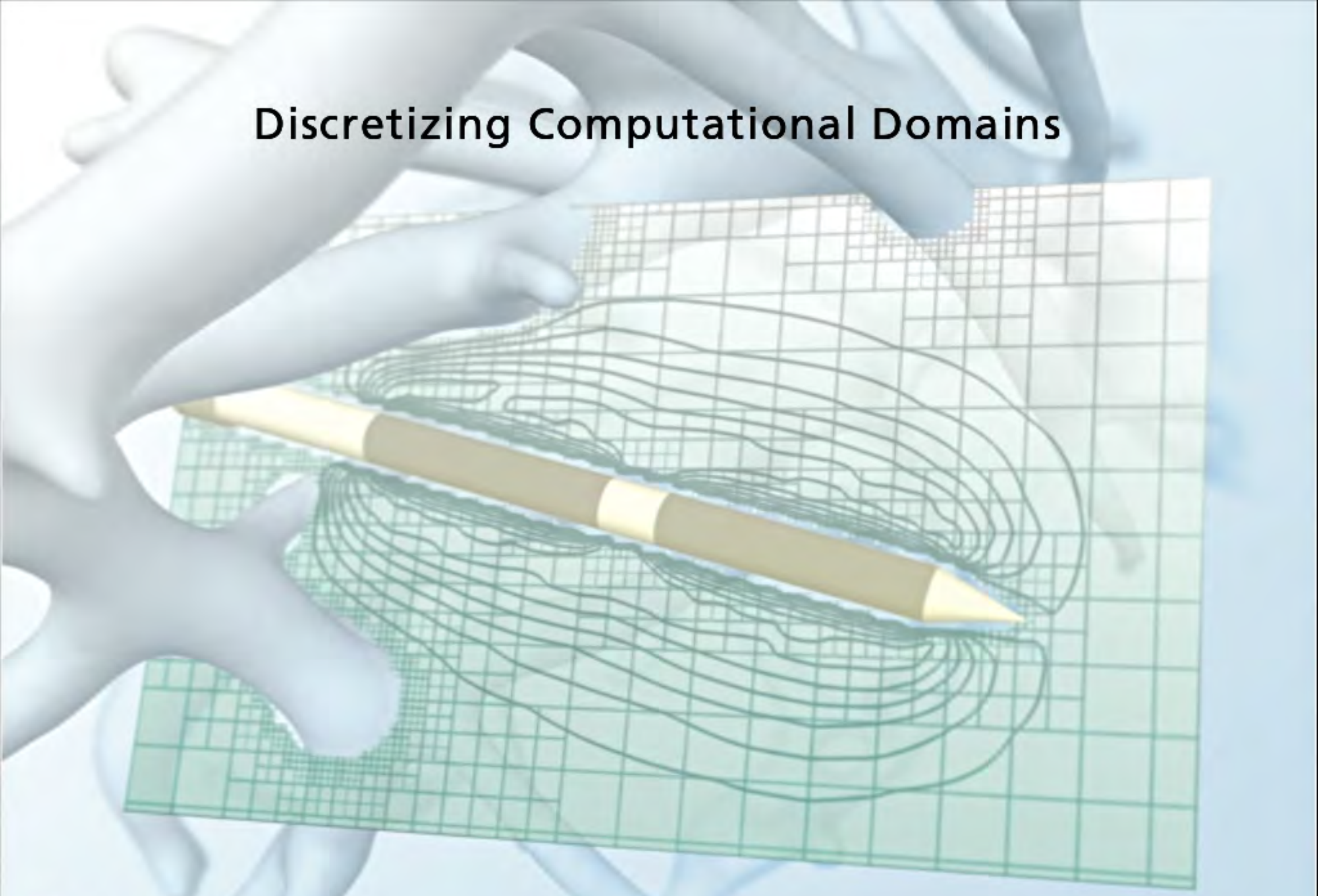
2

Discretizing The Electrostatic Equation

Solving Partial Differential Equations With a Computer

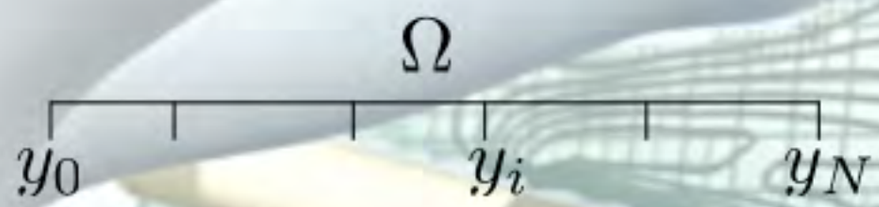
1. • **Discretize the computational domain**
2. • **Discretize the differential operators**
3. • **Solve a system of algebraic equations**

Discretizing Computational Domains



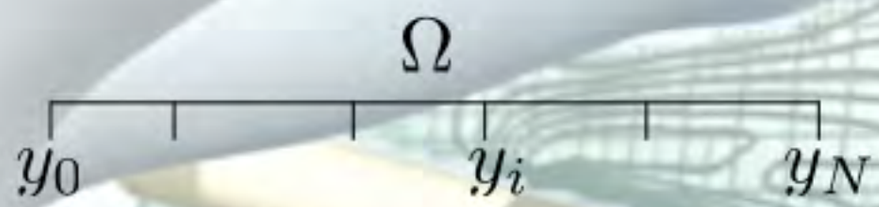
Discretizing Computational Domains

- One space dimension

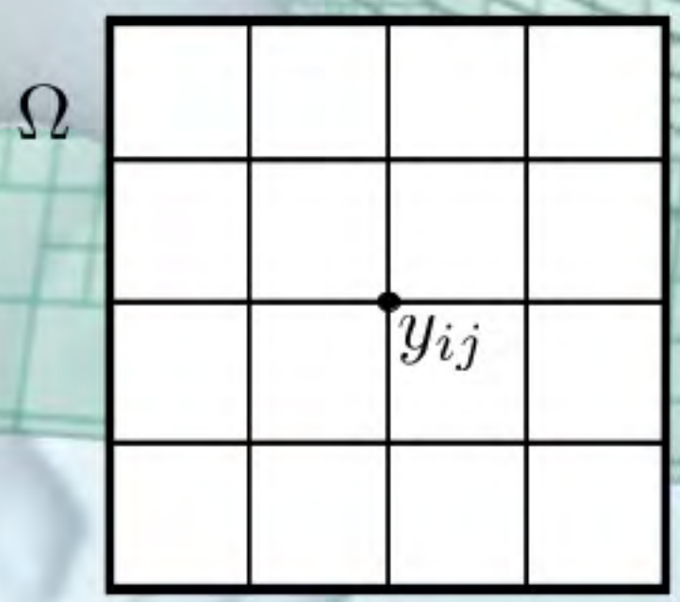


Discretizing Computational Domains

› One space dimension



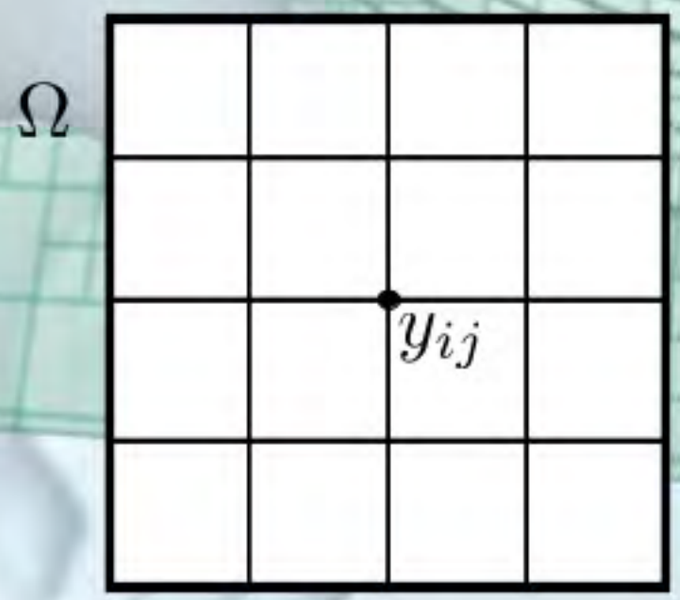
› Two space dimensions



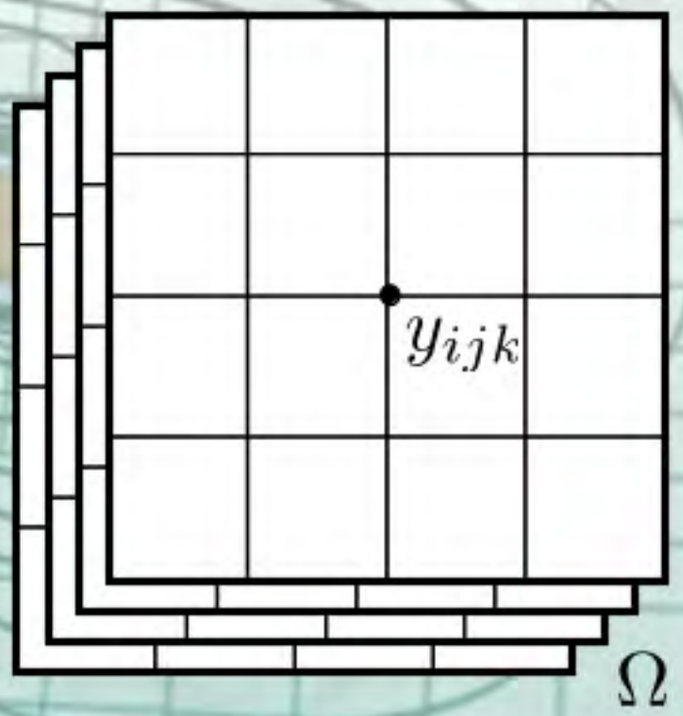
Discretizing Computational Domains

› One space dimension

› Two space dimensions



› Three space dimensions



Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)

Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Difference quotients	

Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Difference quotients › Easy to implement	

Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Difference quotients › Easy to implement › Not very flexible w.r.t geometry of domain	

Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Difference quotients › Easy to implement › Not very flexible w.r.t geometry of domain	› Shape functions of variable smoothness

Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Difference quotients › Easy to implement › Not very flexible w.r.t geometry of domain	› Shape functions of variable smoothness › Difficult to implement

Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Difference quotients › Easy to implement › Not very flexible w.r.t geometry of domain	› Shape functions of variable smoothness › Difficult to implement › Very flexible w.r.t geometry of domain

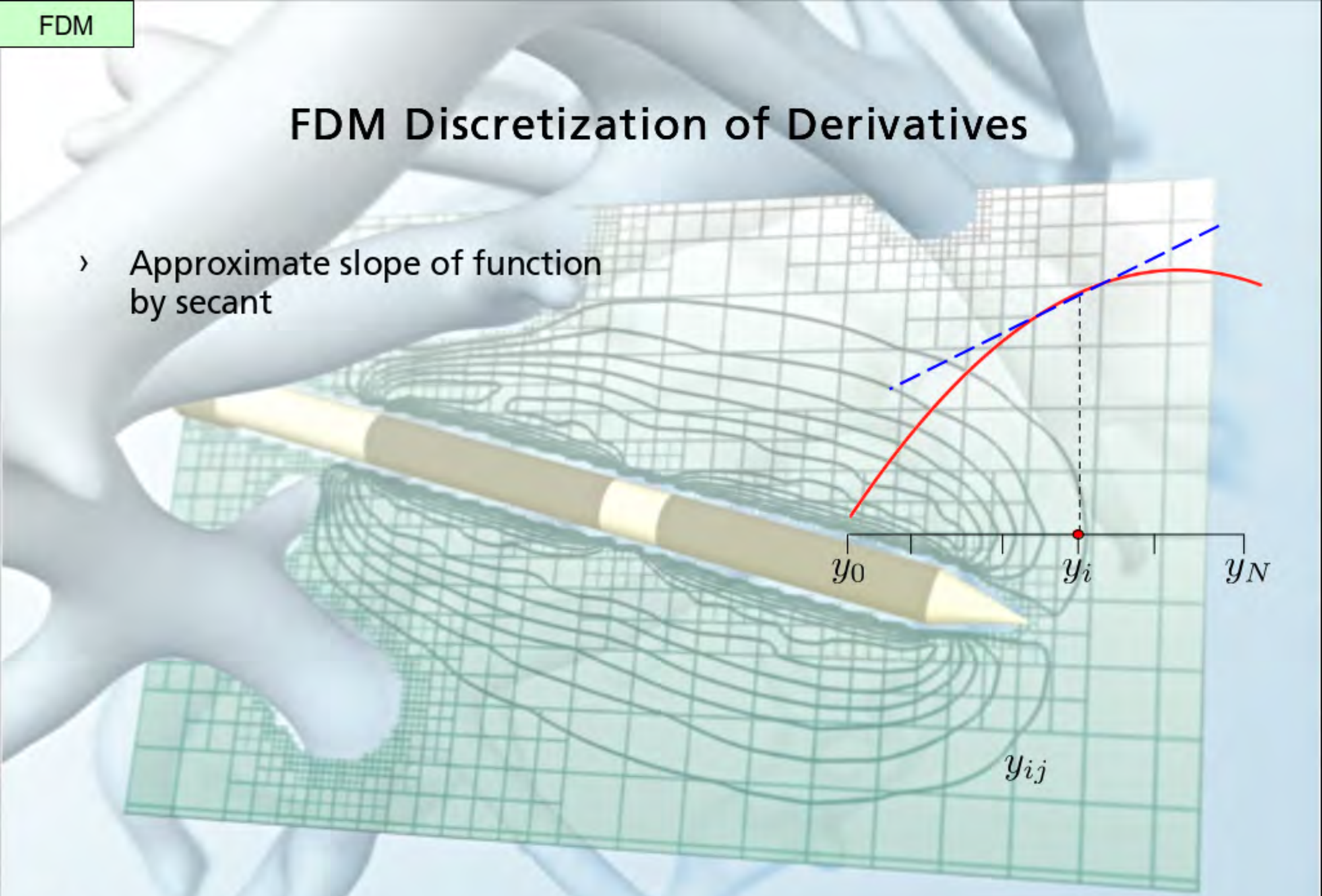
Discretizing Differential Operators

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Difference quotients › Easy to implement › Not very flexible w.r.t geometry of domain	› Shape functions of variable smoothness › Difficult to implement › Very flexible w.r.t geometry of domain › Rich mathematical framework

The Finite Difference Method

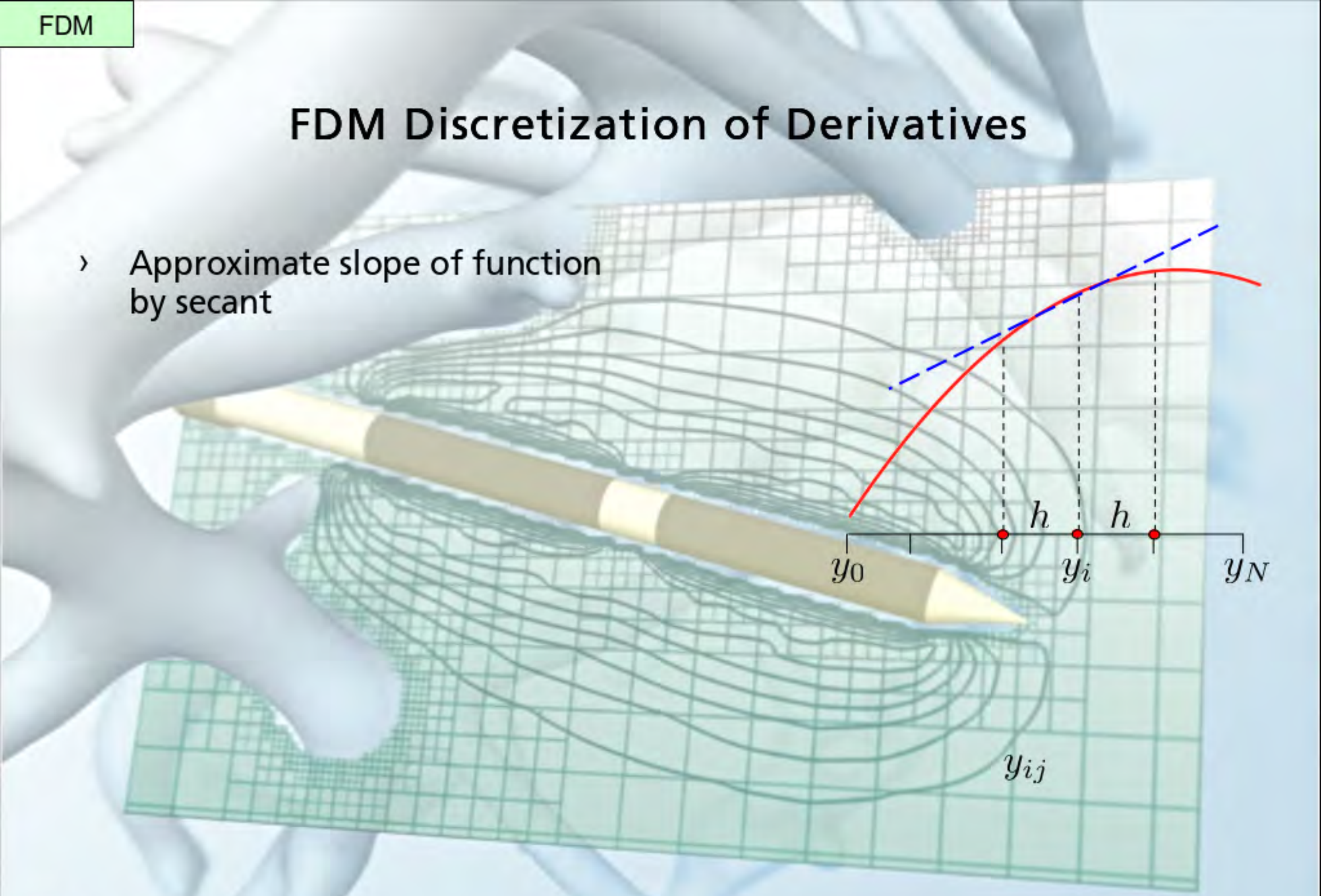
FDM Discretization of Derivatives

- Approximate slope of function by secant



FDM Discretization of Derivatives

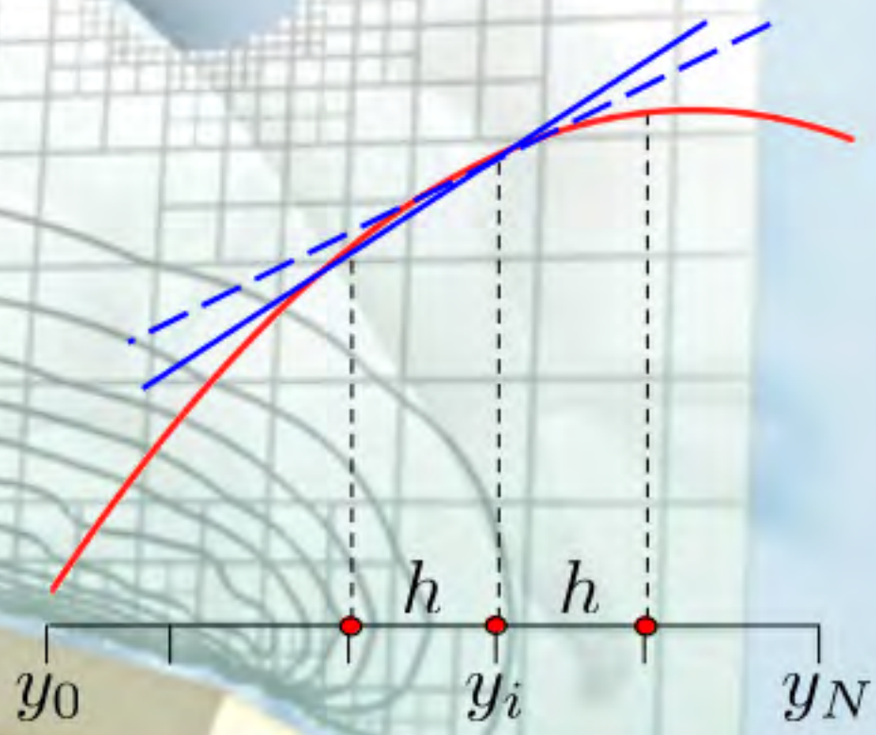
- Approximate slope of function by secant



FDM Discretization of Derivatives

- Approximate slope of function by secant

Backward difference $$u'(y_i) \approx \frac{u_i - u_{i-1}}{h}$$

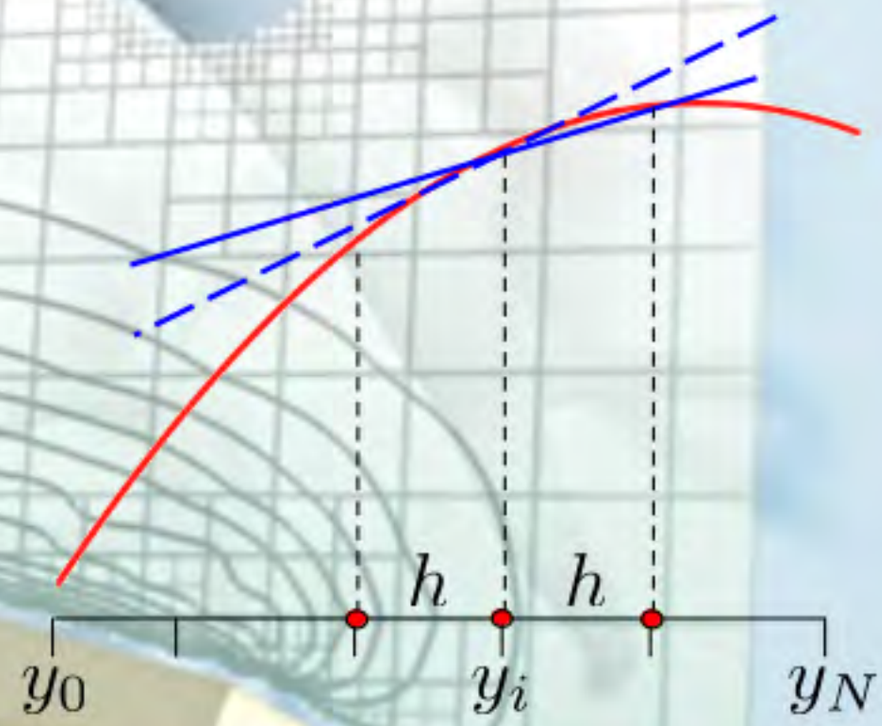


FDM Discretization of Derivatives

- Approximate slope of function by secant

Backward difference $u'(y_i) \approx \frac{u_i - u_{i-1}}{h}$

Forward difference $u'(y_i) \approx \frac{u_{i+1} - u_i}{h}$



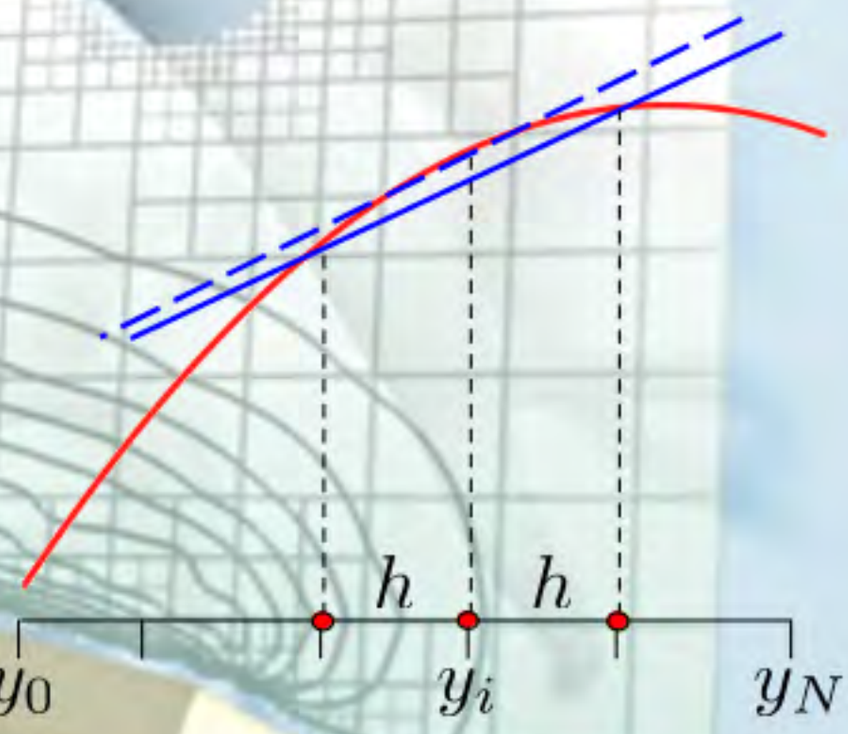
FDM Discretization of Derivatives

- Approximate slope of function by secant

Backward difference $u'(y_i) \approx \frac{u_i - u_{i-1}}{h}$

Forward difference $u'(y_i) \approx \frac{u_{i+1} - u_i}{h}$

Central difference $u'(y_i) \approx \frac{u_{i+1} - u_{i-1}}{2h}$



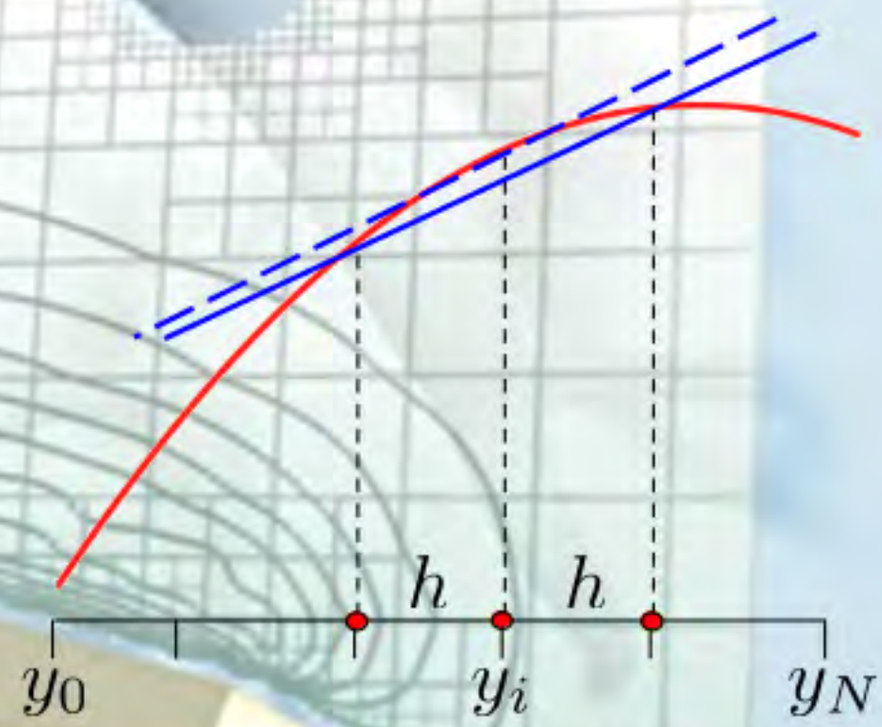
FDM Discretization of Derivatives

- Approximate slope of function by secant

Backward difference $$u'(y_i) \approx \frac{u_i - u_{i-1}}{h}$$

Forward difference $$u'(y_i) \approx \frac{u_{i+1} - u_i}{h}$$

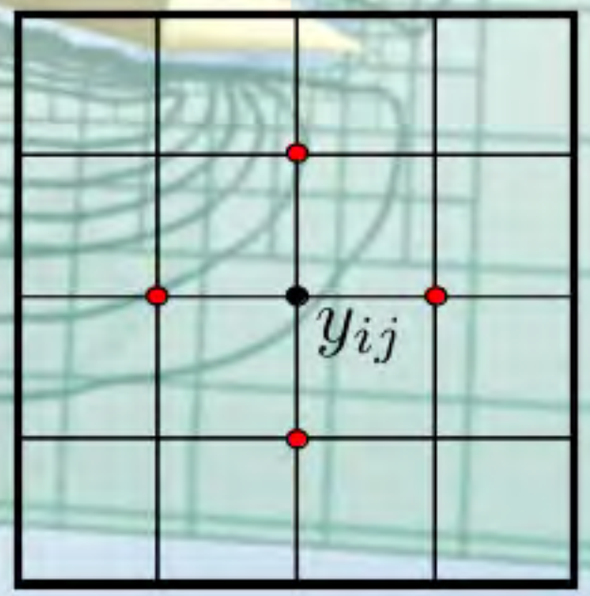
Central difference $$u'(y_i) \approx \frac{u_{i+1} - u_{i-1}}{2h}$$



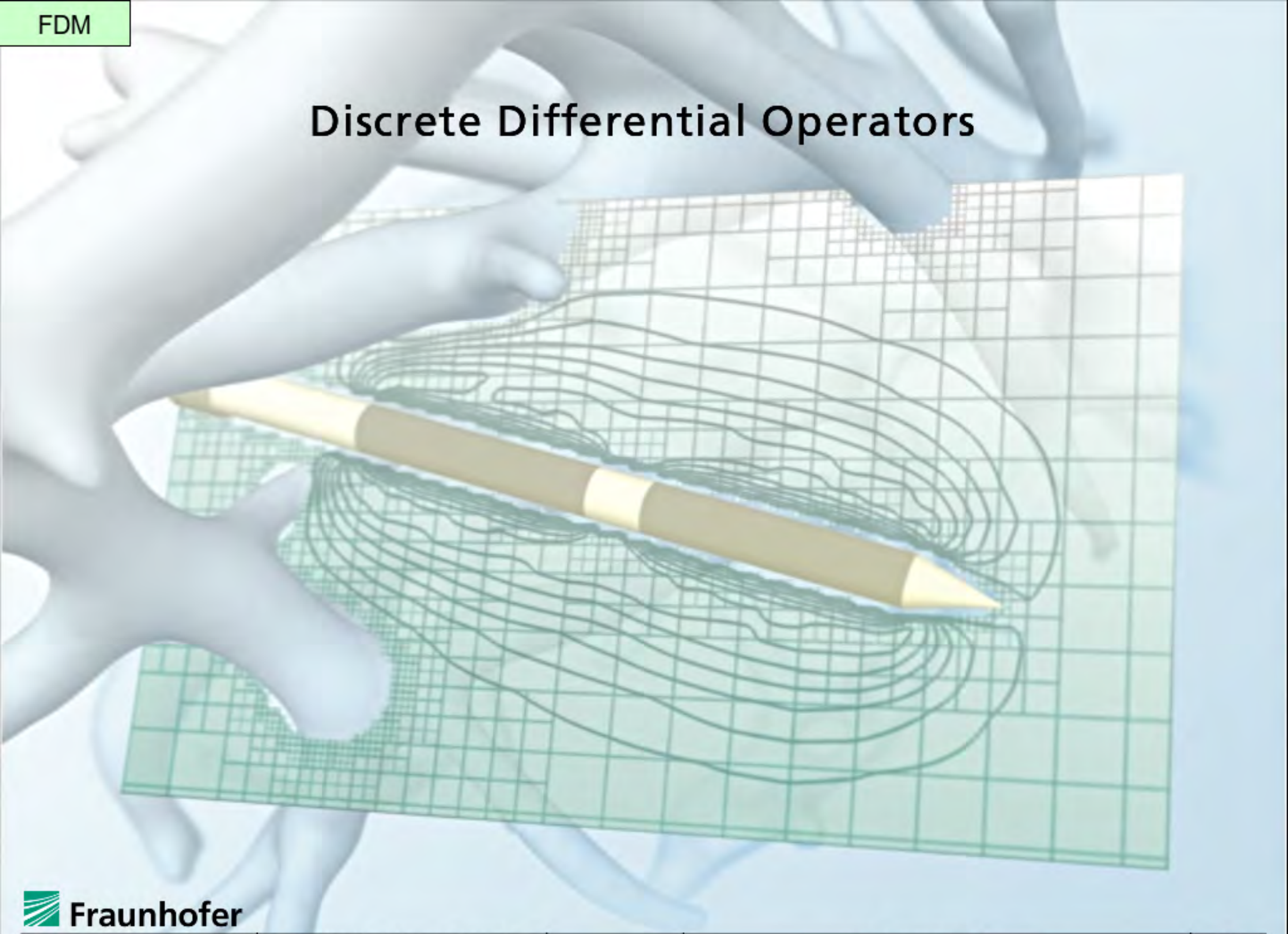
- Higher dimensions: partial derivatives

$$\partial_{x_1} u(y_{ij}) \approx \frac{u_{i,j} - u_{i-1,j}}{h}$$

$$\partial_{x_2} u(y_{ij}) \approx \frac{u_{i,j+1} - u_{i,j}}{h}$$



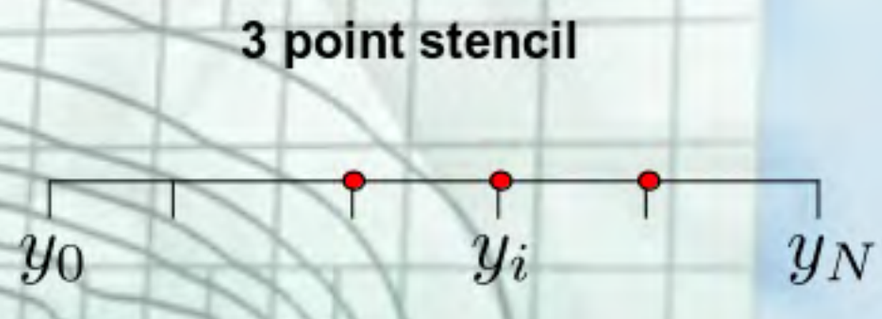
Discrete Differential Operators



Discrete Differential Operators

- Discrete Laplacian in 1D

$$\Delta u(y_i) = u''(y_i) \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$$

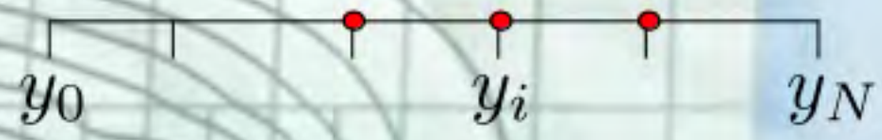


Discrete Differential Operators

- Discrete Laplacian in 1D

$$\Delta u(y_i) = u''(y_i) \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$$

3 point stencil



- Discrete Laplacian in 2D

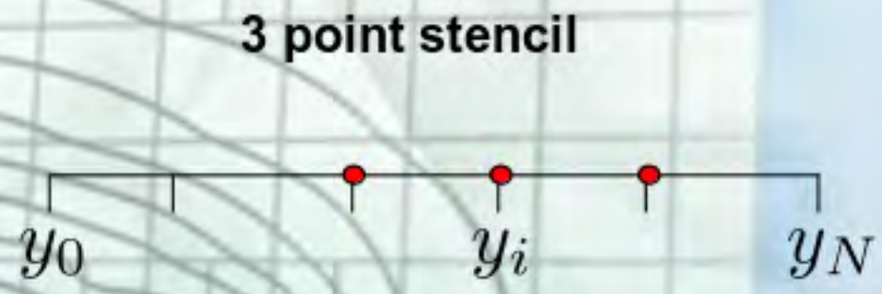
$$\Delta u(y_{ij}) = \partial_{x_1 x_1} u(y_{ij}) + \partial_{x_2 x_2} u(y_{ij})$$
$$\approx \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}}{h^2}$$

5 point stencil

Discrete Differential Operators

- Discrete Laplacian in 1D

$$\Delta u(y_i) = u''(y_i) \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$$



- Discrete Laplacian in 2D

$$\Delta u(y_{ij}) = \partial_{x_1 x_1} u(y_{ij}) + \partial_{x_2 x_2} u(y_{ij})$$
$$\approx \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}}{h^2}$$

5 point stencil

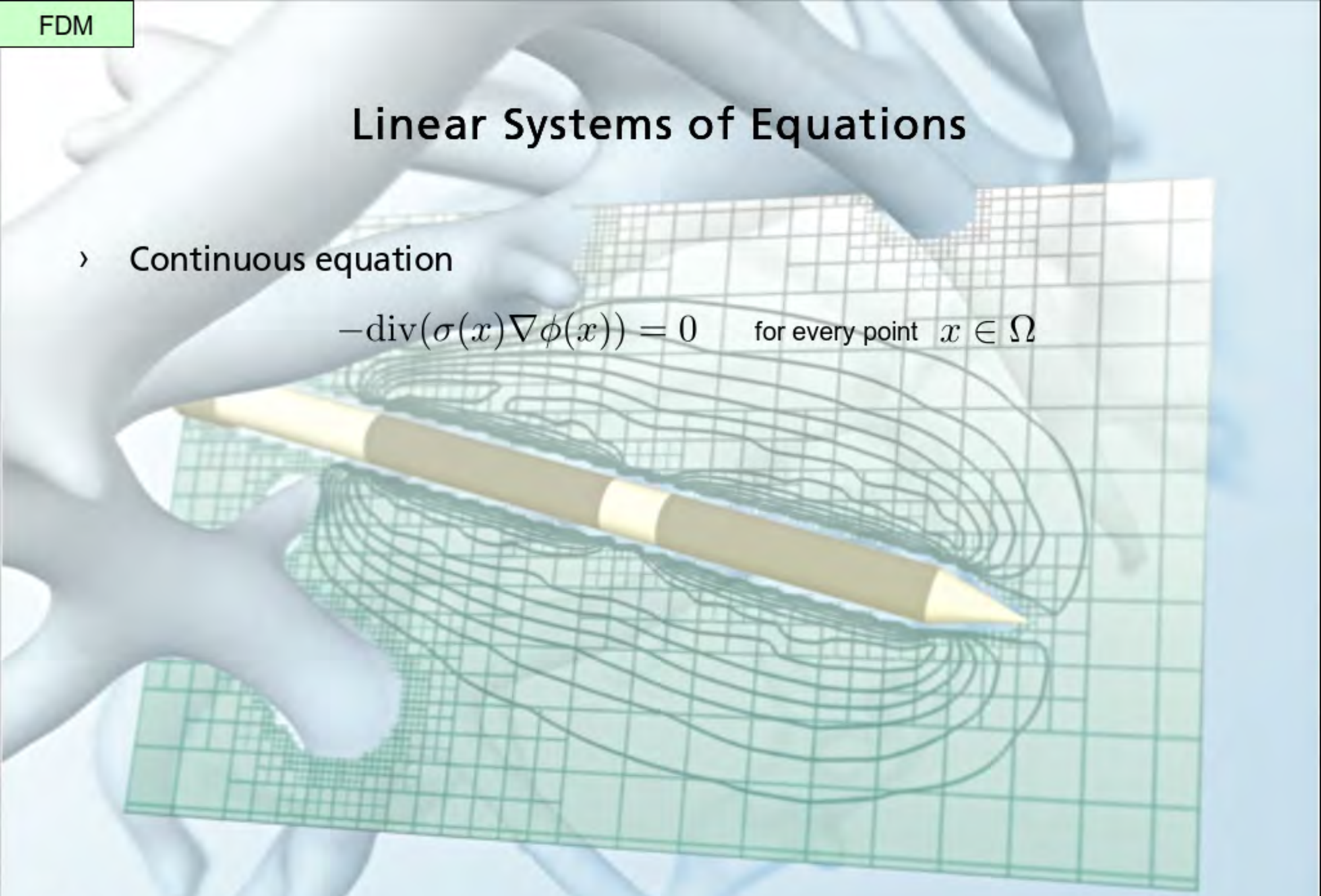
- Discrete Laplacian in 3D

7 point stencil

Linear Systems of Equations

- Continuous equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$



Linear Systems of Equations

- Continuous equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$

- Discrete equation

$$\sum_{kl} \alpha_{ij,kl}\phi_{kl} = 0 \qquad \text{for every grid node } y_{ij}$$

Linear Systems of Equations

- Continuous equation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$

- Discrete equation

$$\sum_{kl} \alpha_{ij,kl}\phi_{kl} = 0 \qquad \text{for every grid node } y_{ij}$$

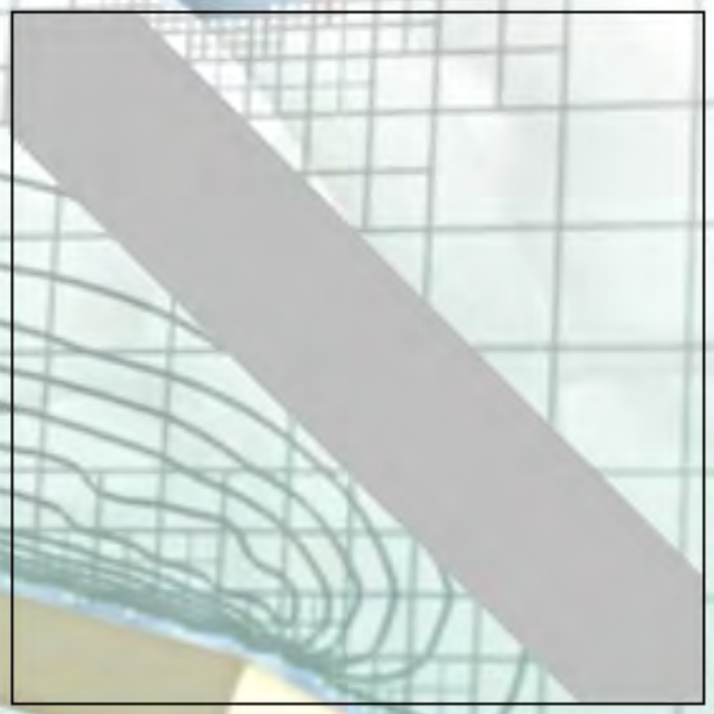
- Linear system of equations

$$A\Phi = F$$

$$A \in I\!R^{(N+1)\times(N+1)}$$
$$\Phi, F \in I\!R^{(N+1)}$$

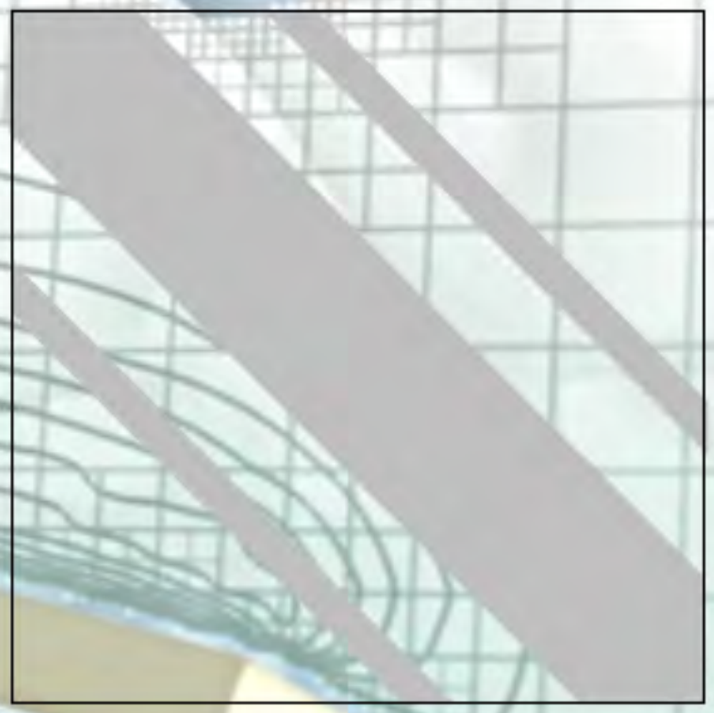
Solving Linear System of Equations

- 3-Band matrix in 1D



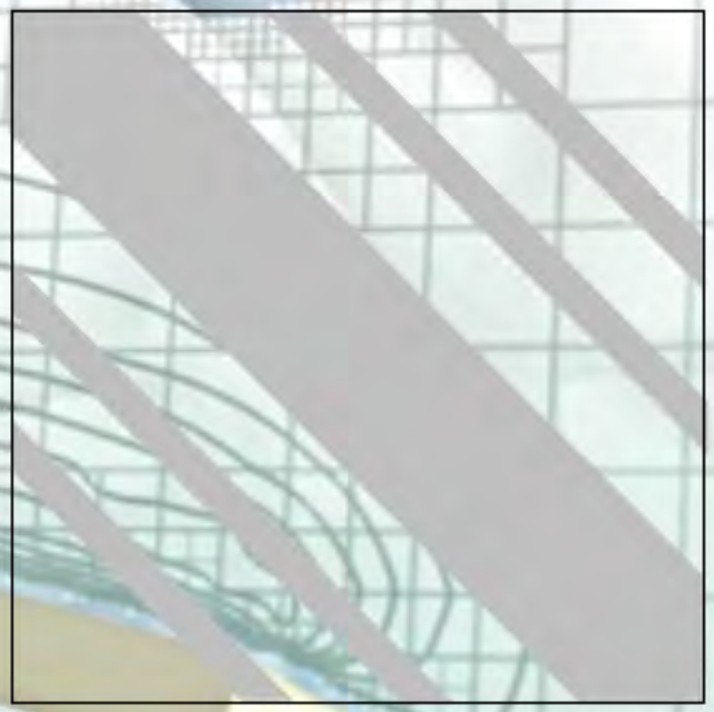
Solving Linear System of Equations

- 3-Band matrix in 1D
- 5-Band matrix in 2D



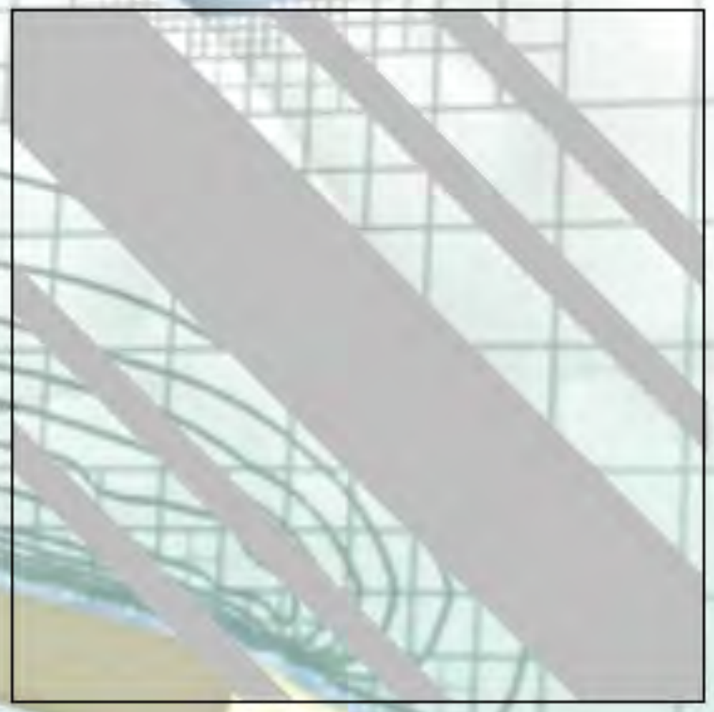
Solving Linear System of Equations

- 3-Band matrix in 1D
- 5-Band matrix in 2D
- 7-Band matrix in 3D



Solving Linear System of Equations

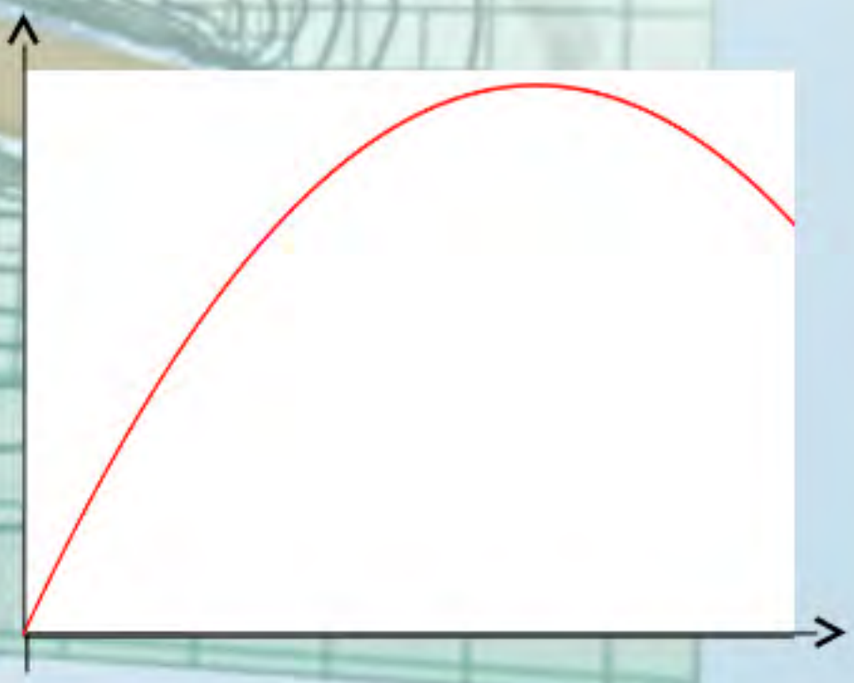
- 3-Band matrix in 1D
- 5-Band matrix in 2D
- 7-Band matrix in 3D



- Matrix inversion:
  - Direct method, e.g. Thomas algorithm
  - Iterative method: e.g. Gauss-Seidel method, conjugate gradients

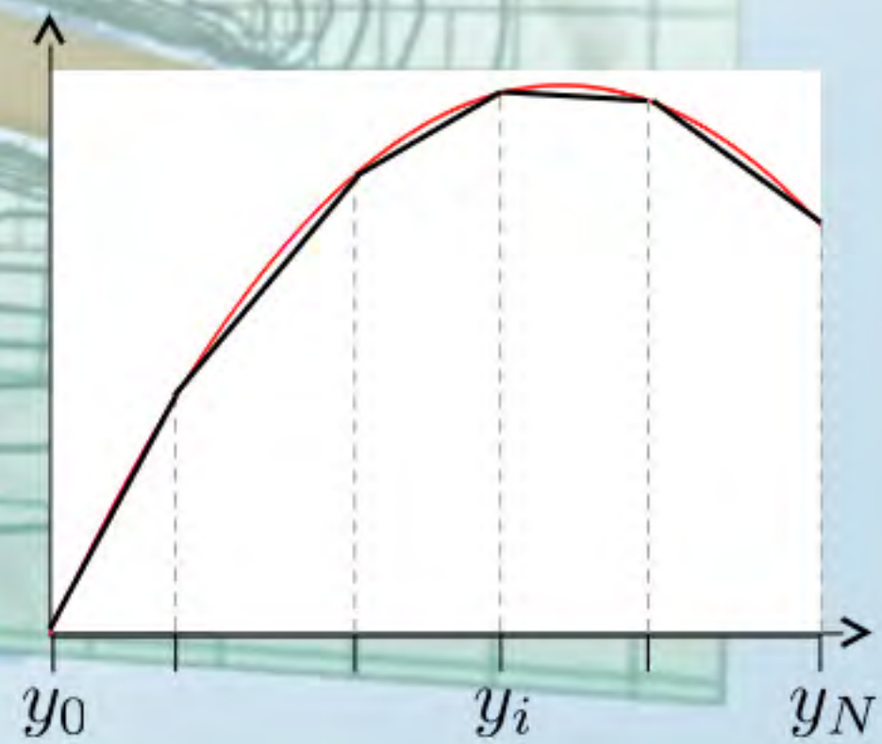
Summary FDM

- Discretize the domain by a regular grid
- Approximate differential operators by difference quotients
- System of linear equations
- Solve by iterative or direct method
- Obtain a numerical solution ϕ_h of the PDE



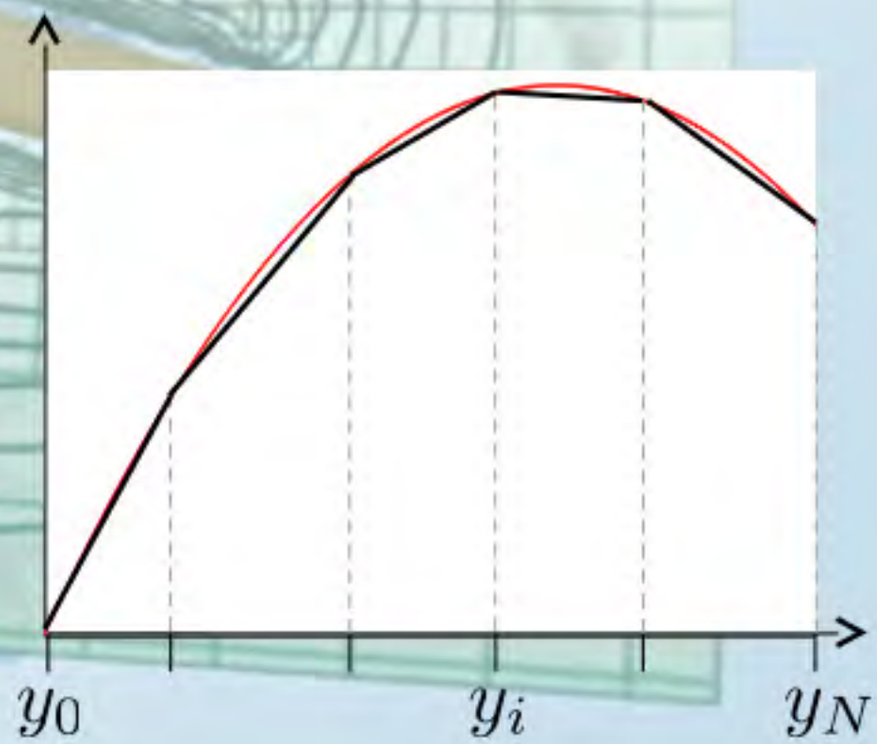
Summary FDM

- Discretize the domain by a regular grid
- Approximate differential operators by difference quotients
- System of linear equations
- Solve by iterative or direct method
- Obtain a numerical solution ϕ_h of the PDE



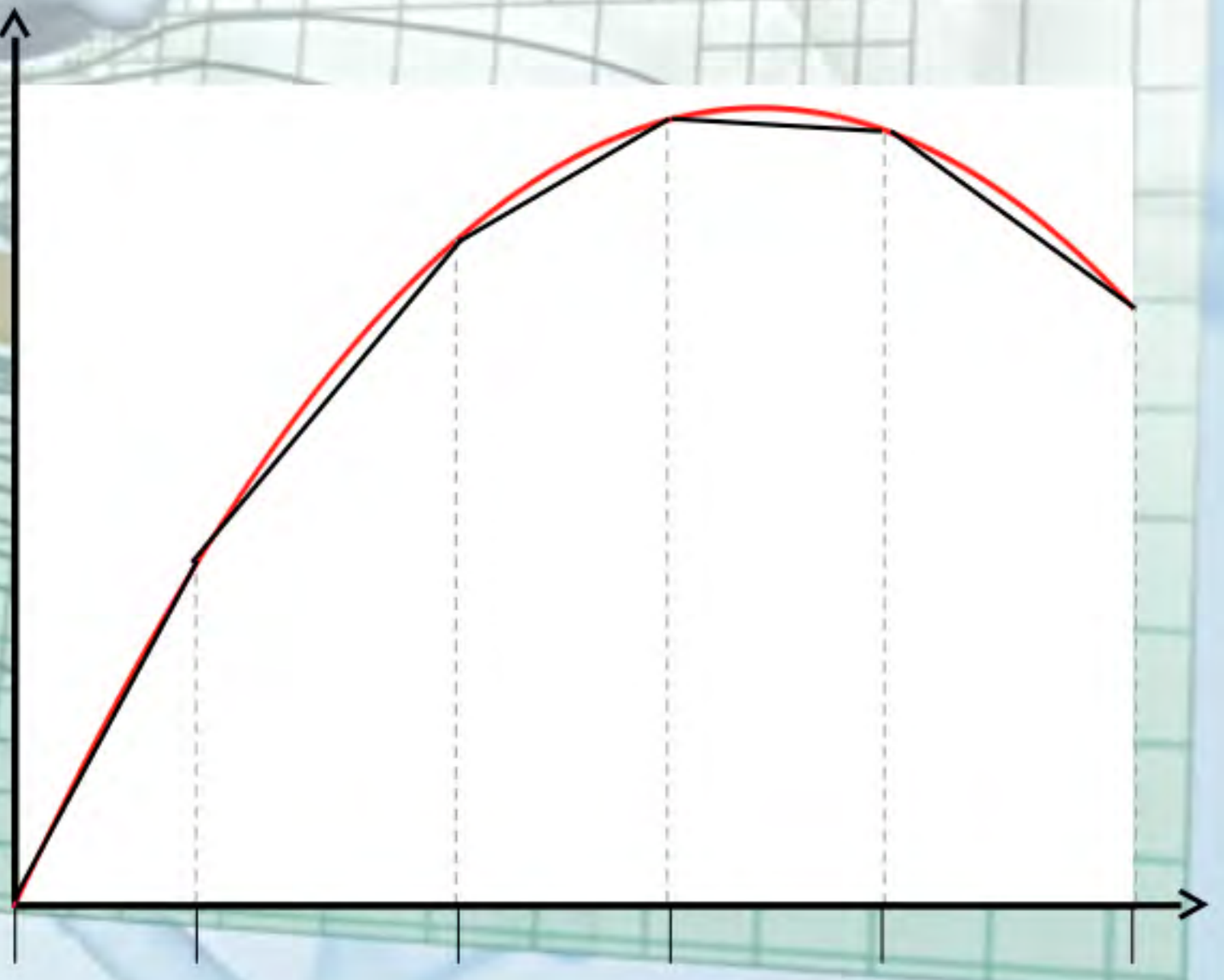
Summary FDM

- Discretize the domain by a regular grid
- Approximate differential operators by difference quotients
- System of linear equations
- Solve by iterative or direct method
- Obtain a numerical solution ϕ_h of the PDE
- Accuracy $\|\phi - \phi_h\|_{L^2} = Ch^2$

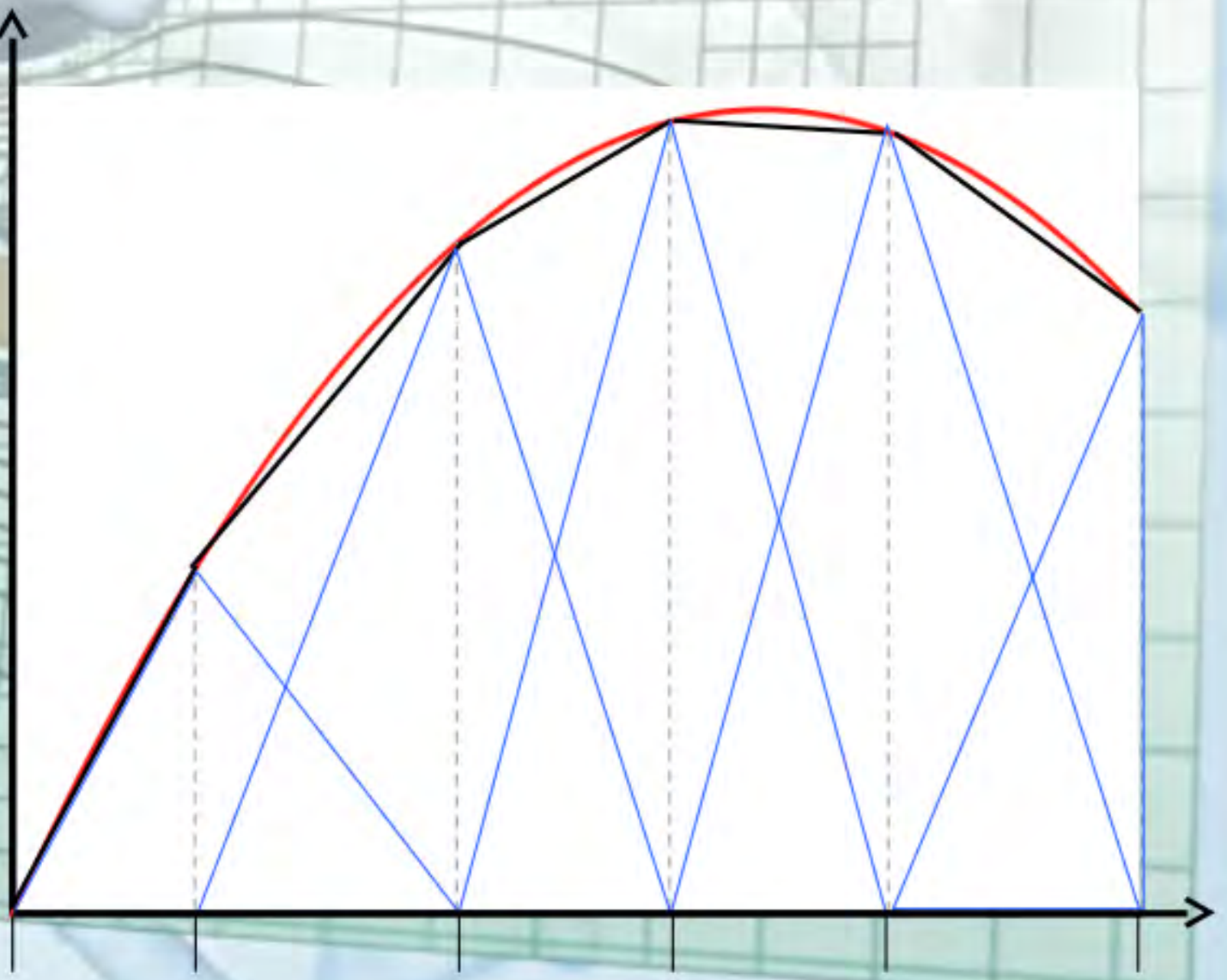


The Finite Element Method

Finite Element Shape Functions

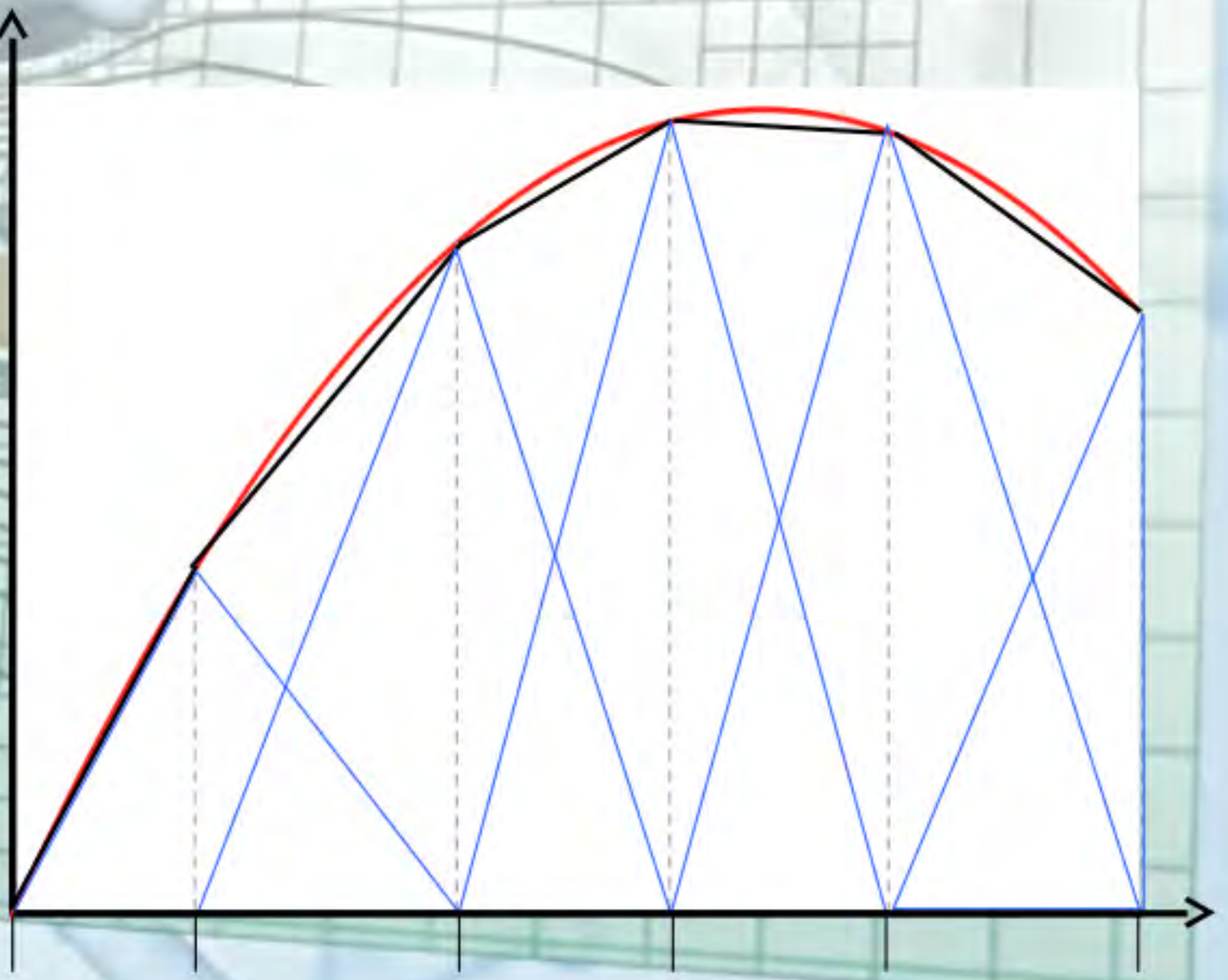


Finite Element Shape Functions

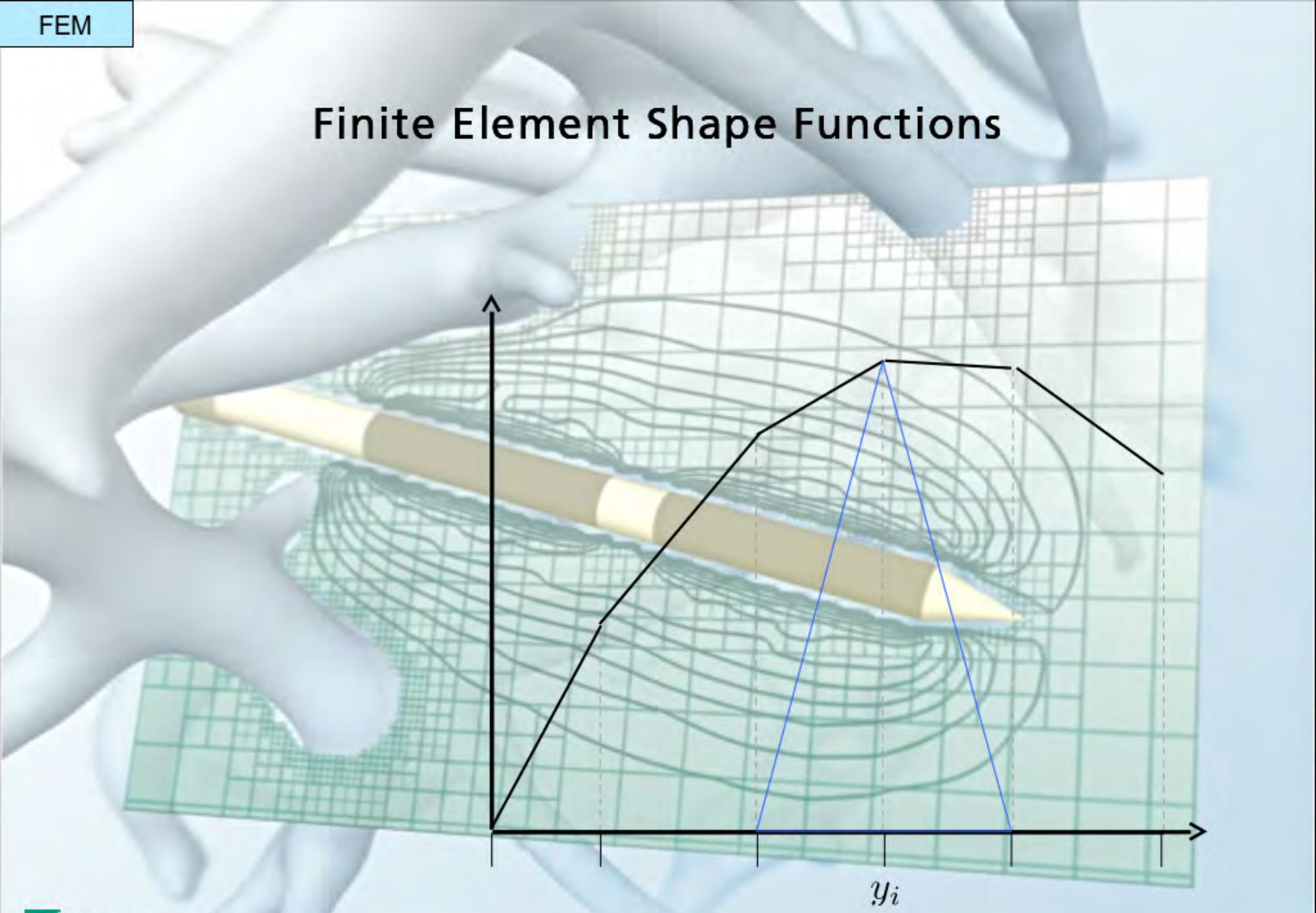


Finite Element Shape Functions

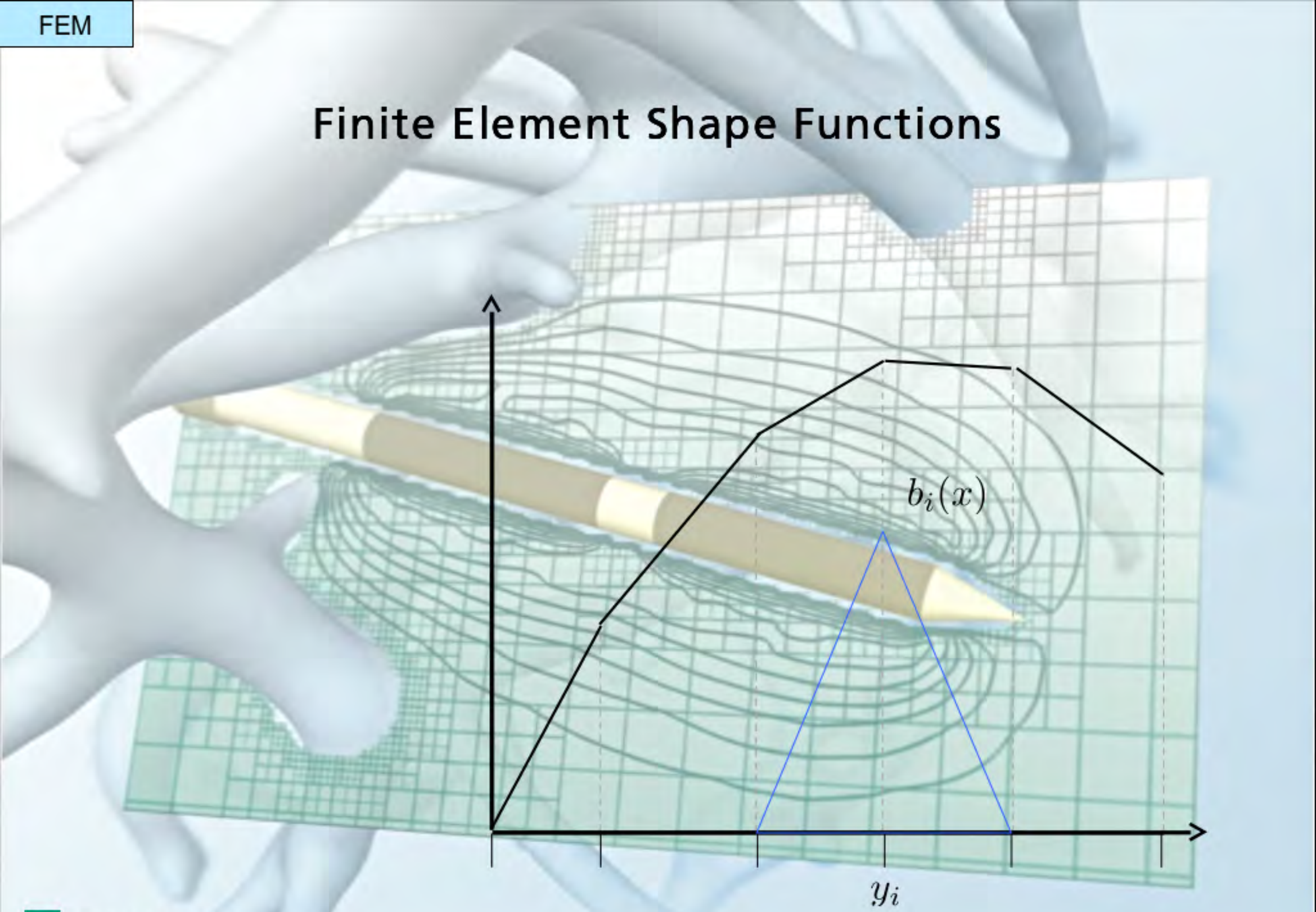
› Compose the solution from predefined shape functions



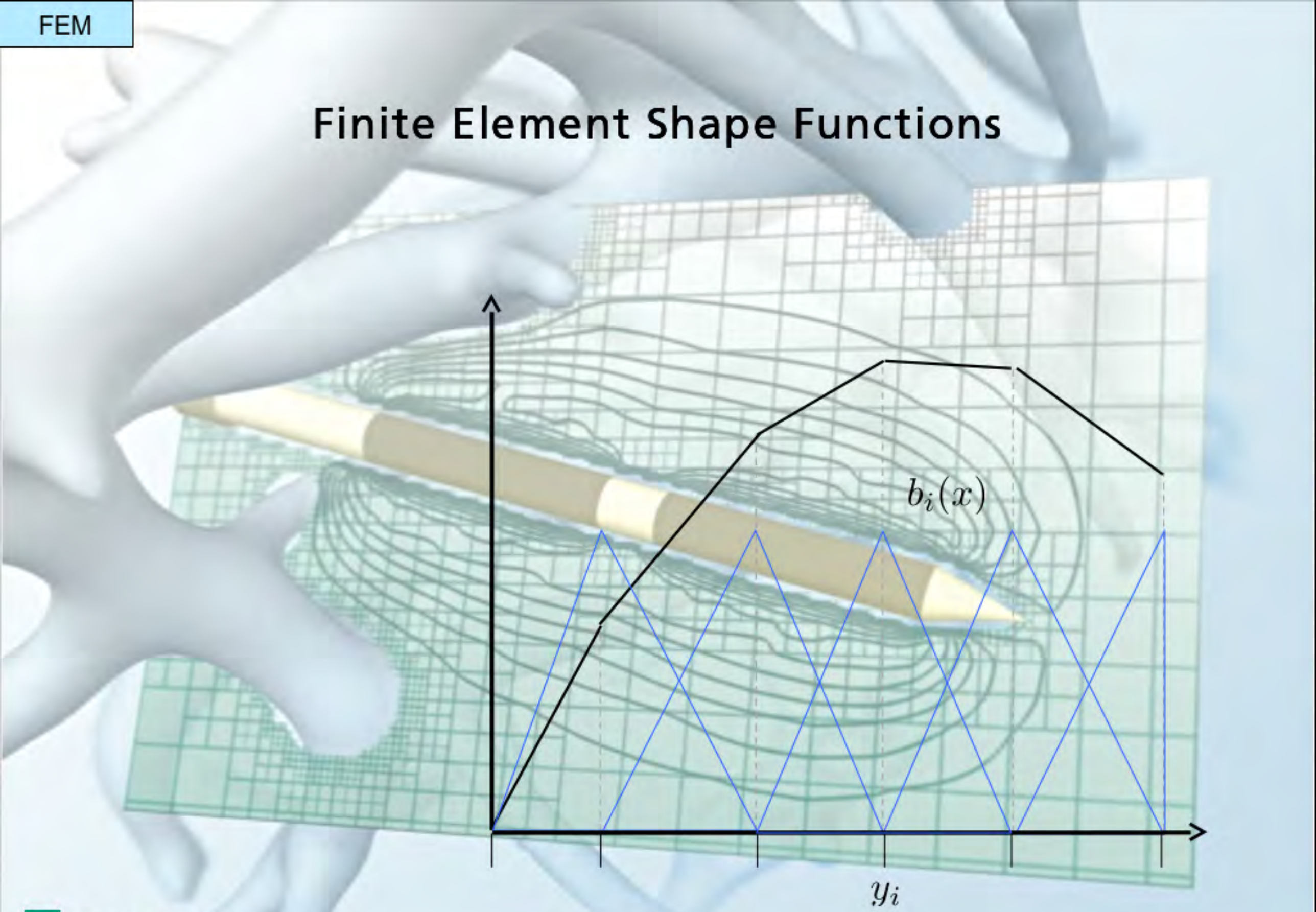
Finite Element Shape Functions



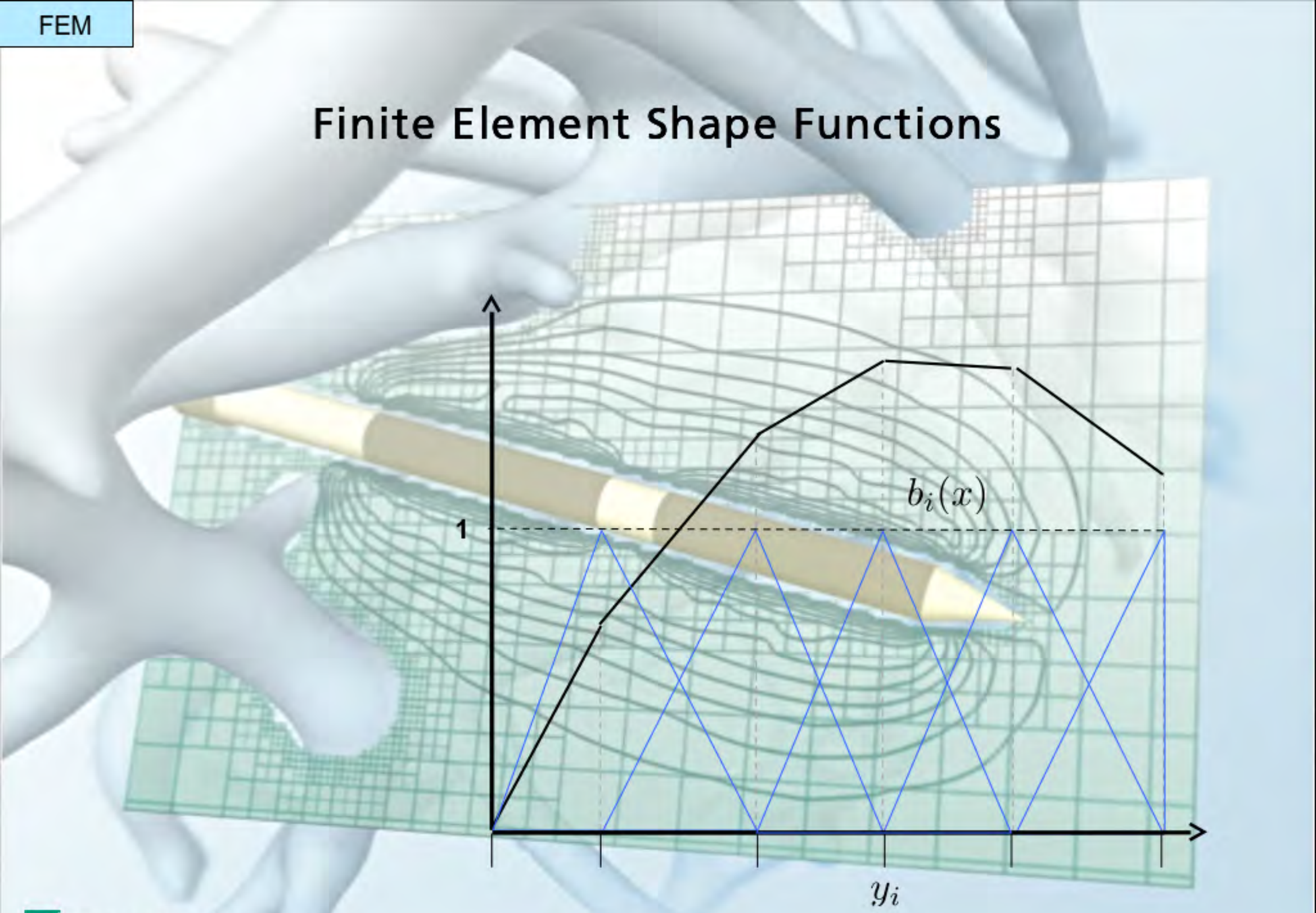
Finite Element Shape Functions



Finite Element Shape Functions



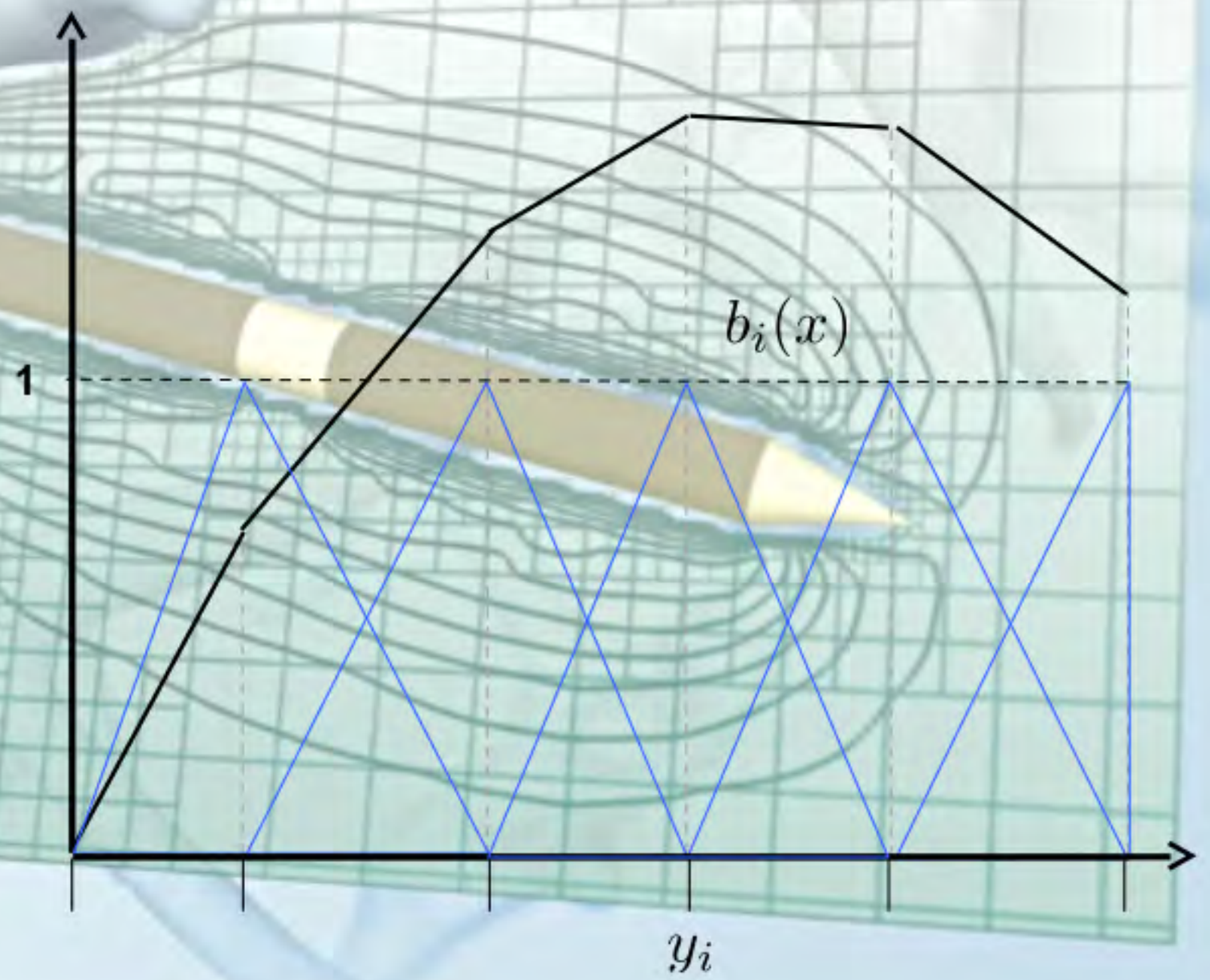
Finite Element Shape Functions



Finite Element Shape Functions

$$\phi(x) = \phi_0 b_0(x) + \phi_1 b_1(x) + \cdots + \phi_N b_N(x)$$

$$= \sum_{i=0}^{N} \phi_i b_i(x)$$

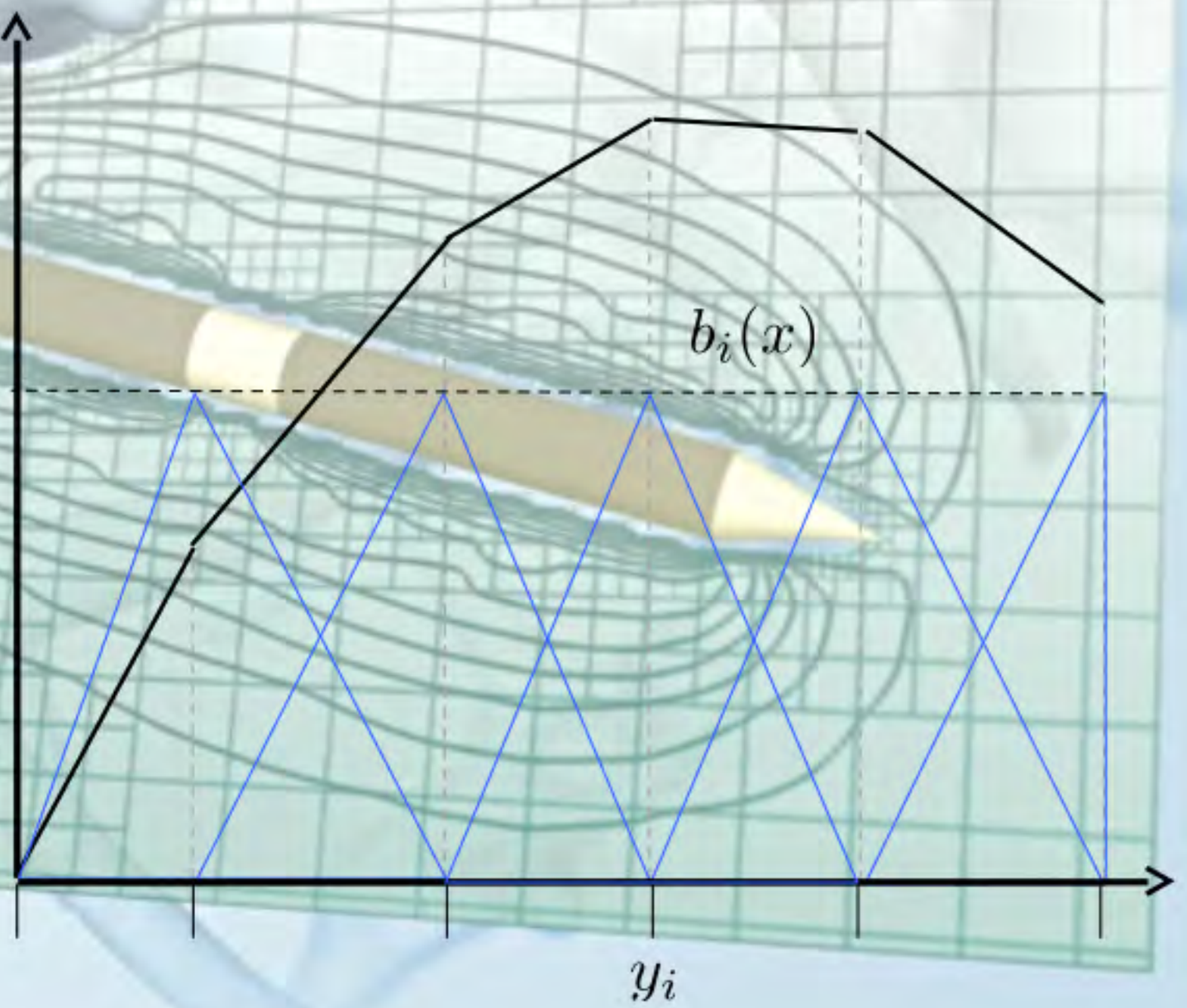


Finite Element Shape Functions

$$\phi(x) = \phi_0 b_0(x) + \phi_1 b_1(x) + \cdots + \phi_N b_N(x)$$

$$= \sum_{i=0}^{N} \phi_i b_i(x)$$

Piecewise linear function

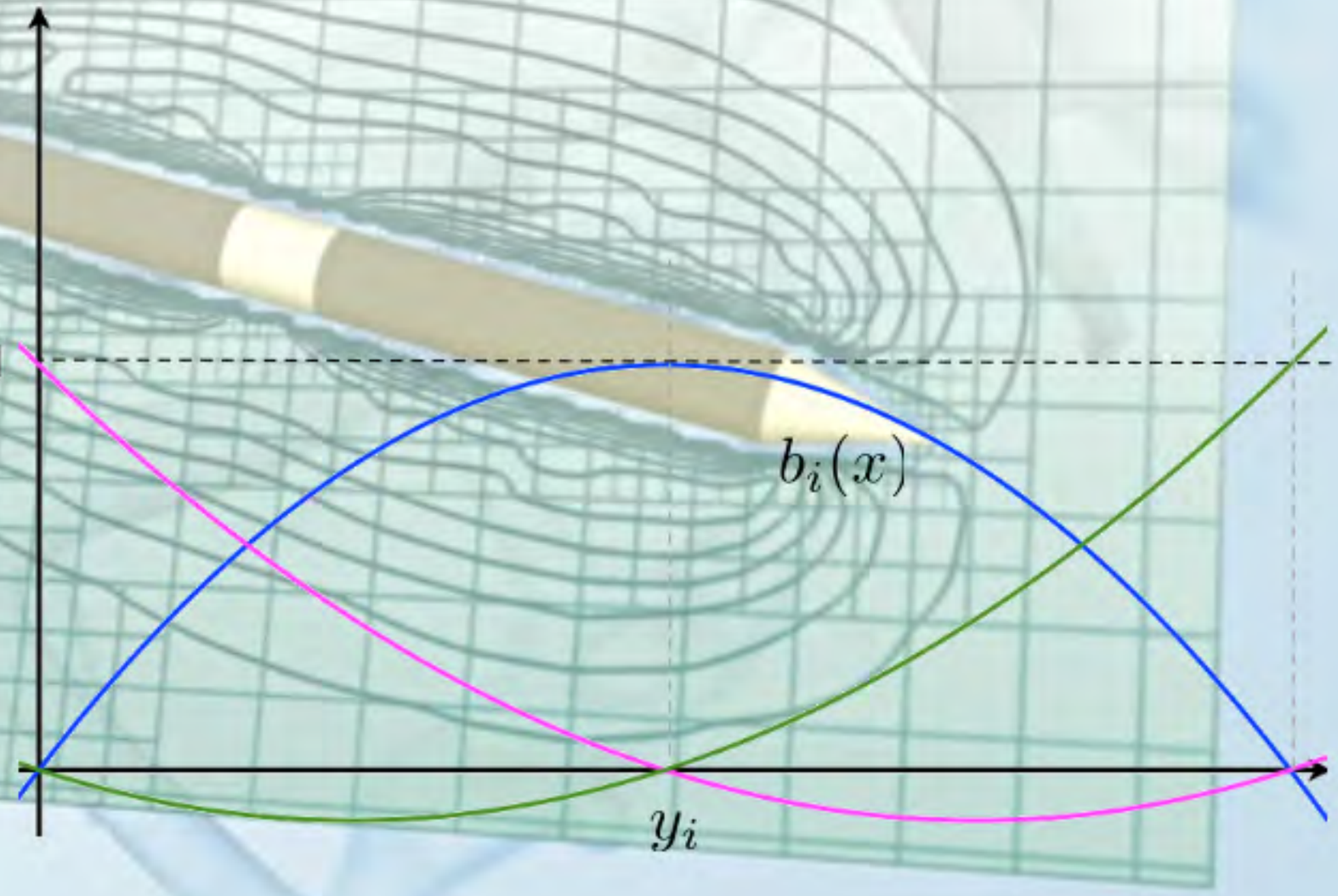


Flexibility of Shape Functions

$$\phi(x) = \phi_0 b_0(x) + \phi_1 b_1(x) + \cdots + \phi_N b_N(x)$$

$$= \sum_{i=0}^{N} \phi_i b_i(x)$$

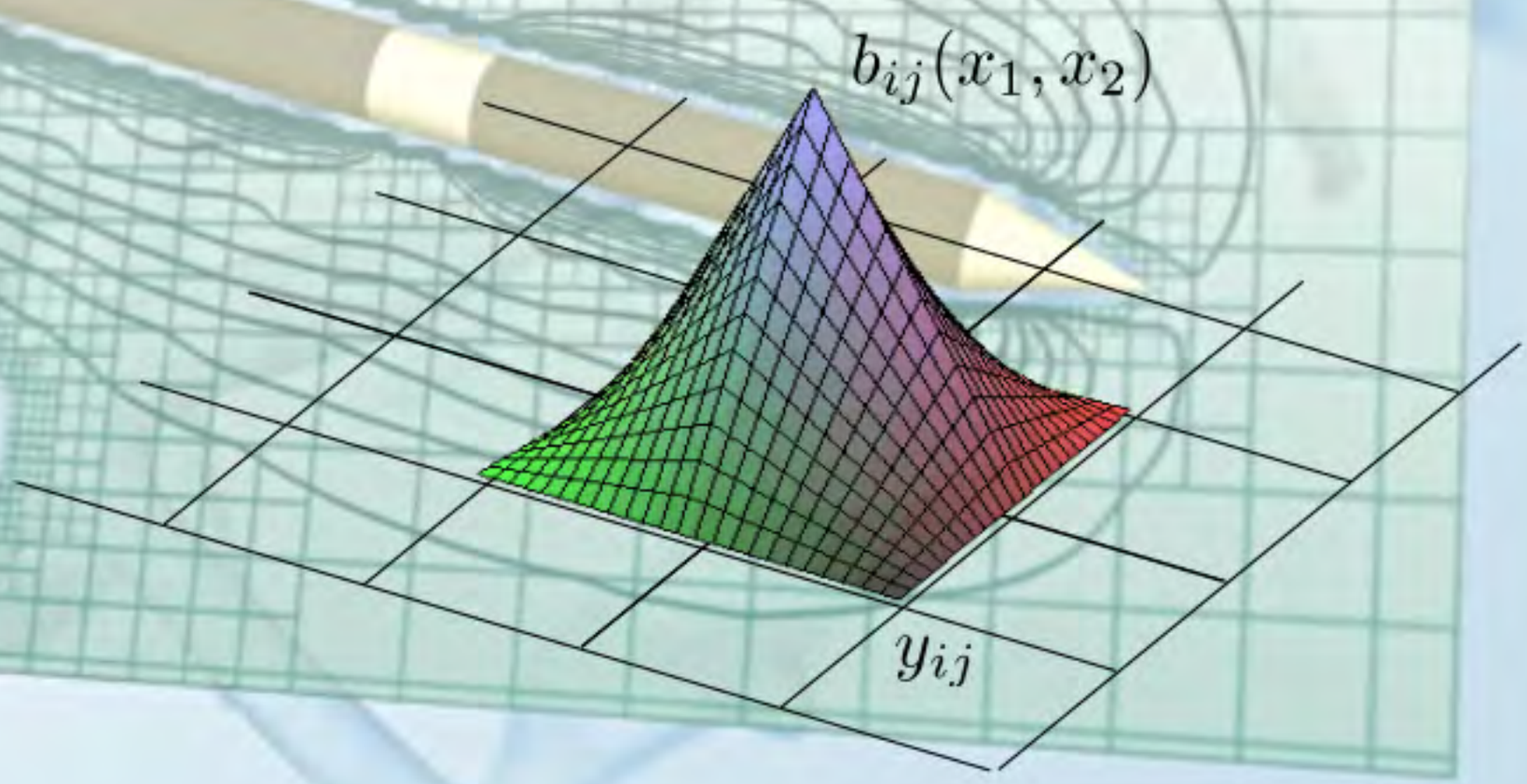
Piecewise polynomial function



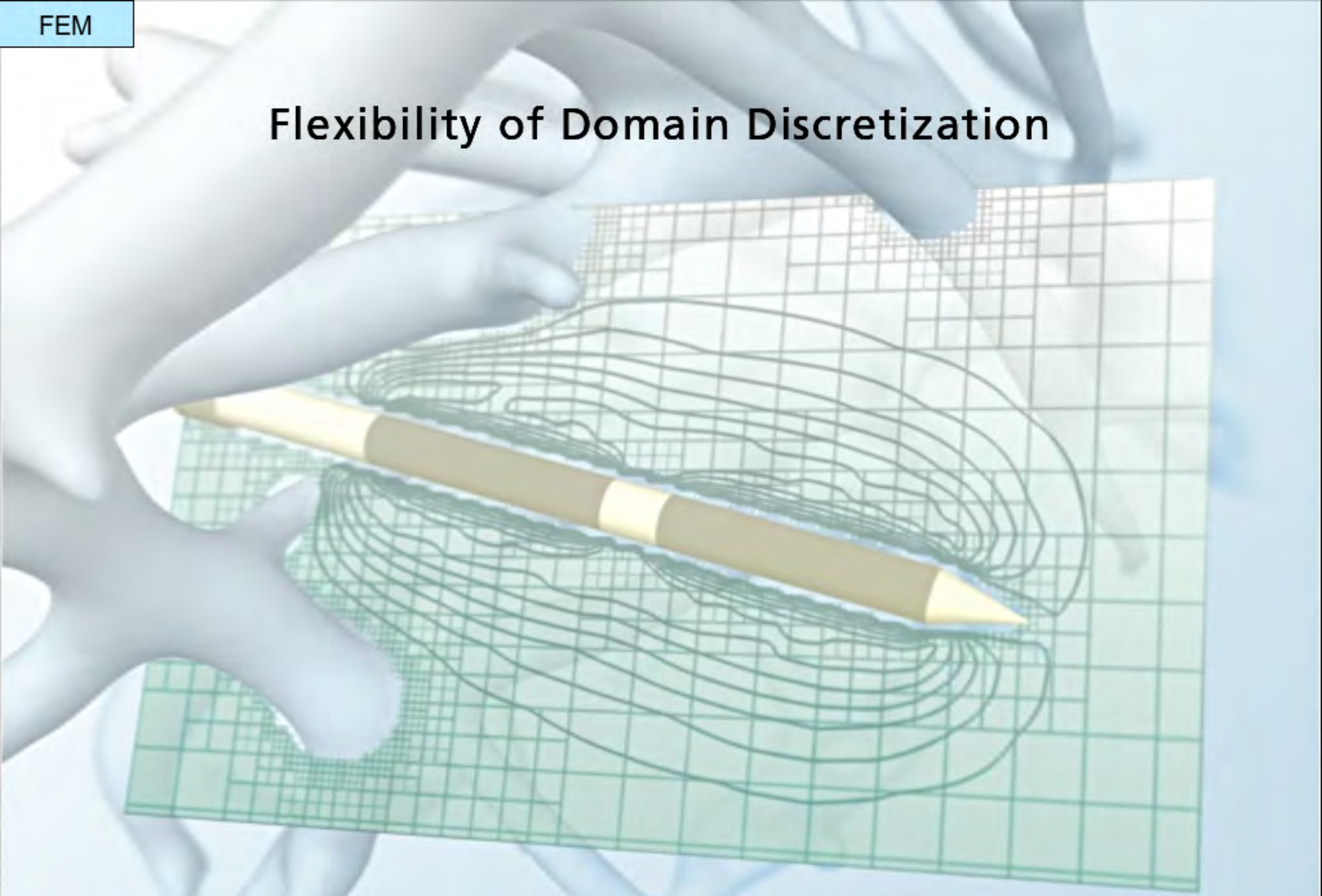
Shape Functions in 2D

$$\phi(x) = \phi(x_1, x_2) = \sum_{i,j} \phi_{ij} b_{ij}(x_1, x_2)$$

Piecewise bilinear function

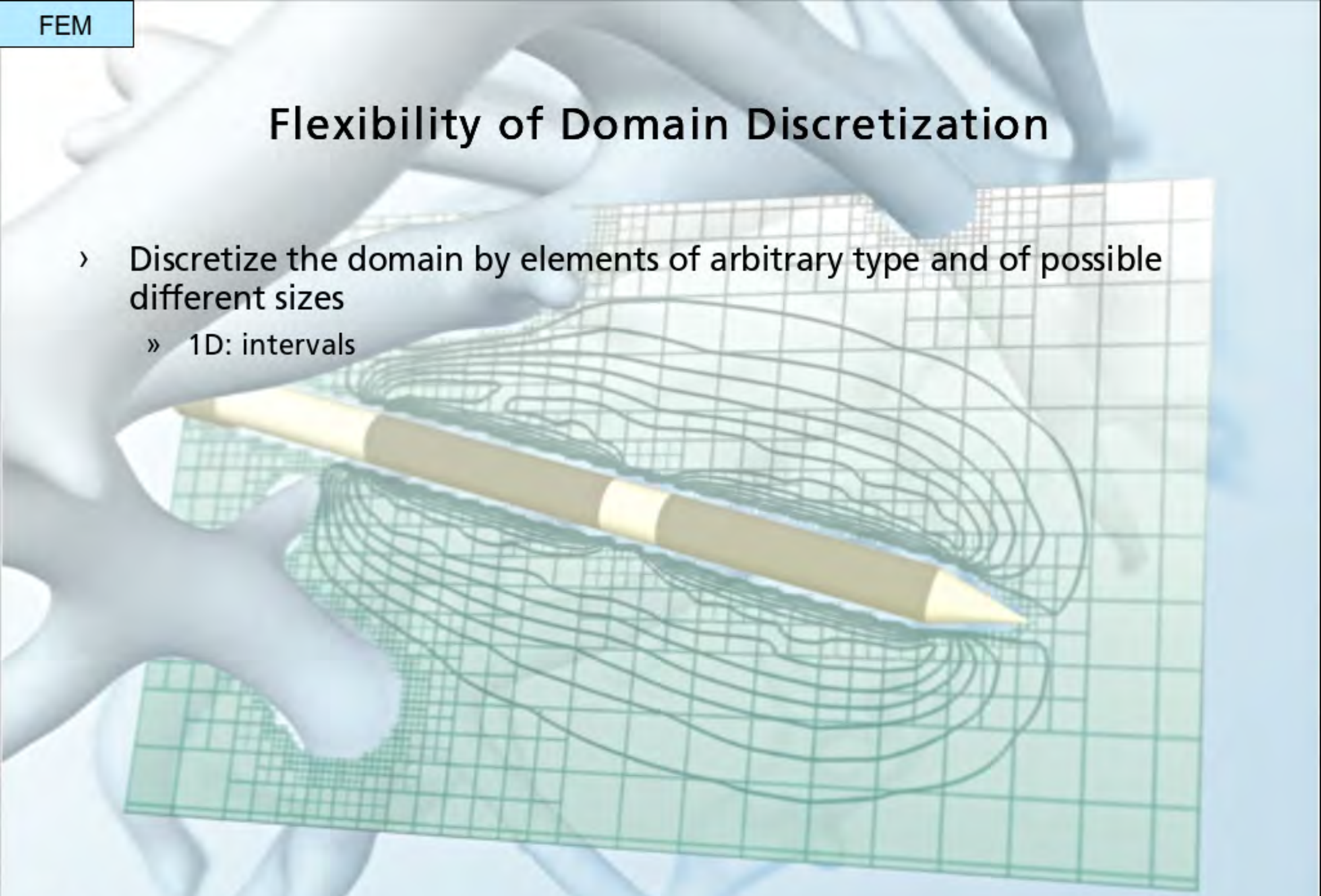


Flexibility of Domain Discretization



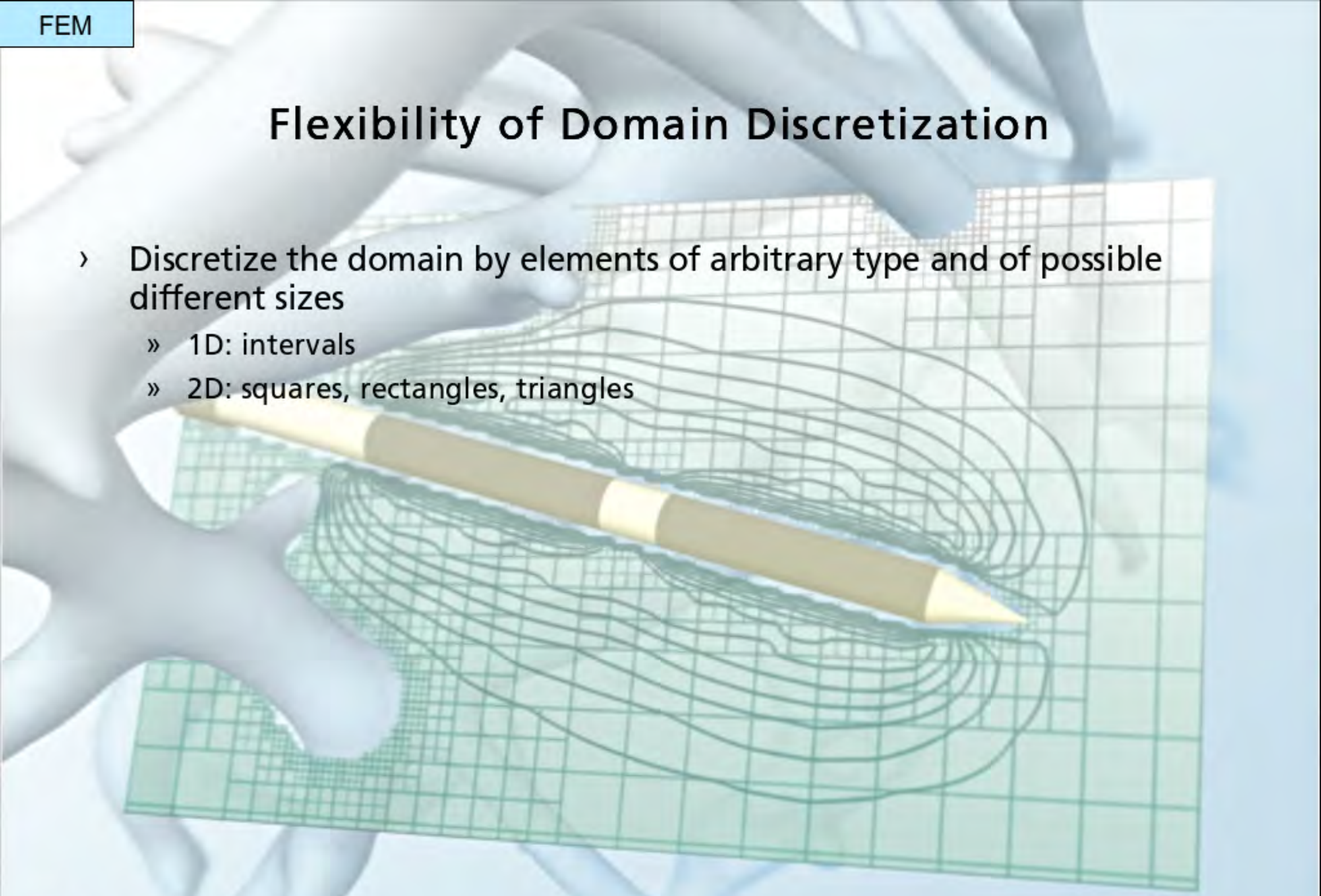
Flexibility of Domain Discretization

- Discretize the domain by elements of arbitrary type and of possible different sizes
  - 1D: intervals



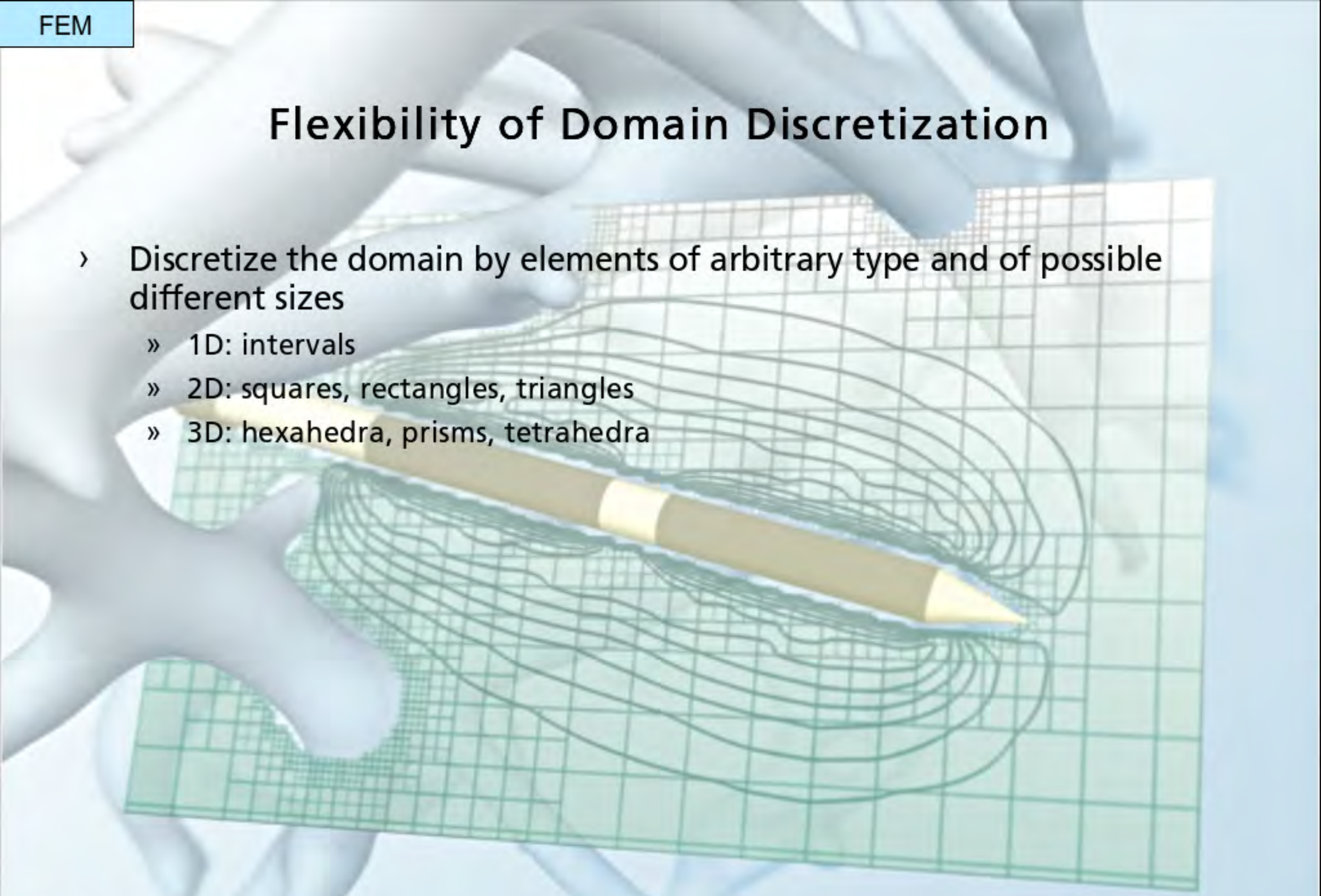
Flexibility of Domain Discretization

- Discretize the domain by elements of arbitrary type and of possible different sizes
  - 1D: intervals
  - 2D: squares, rectangles, triangles



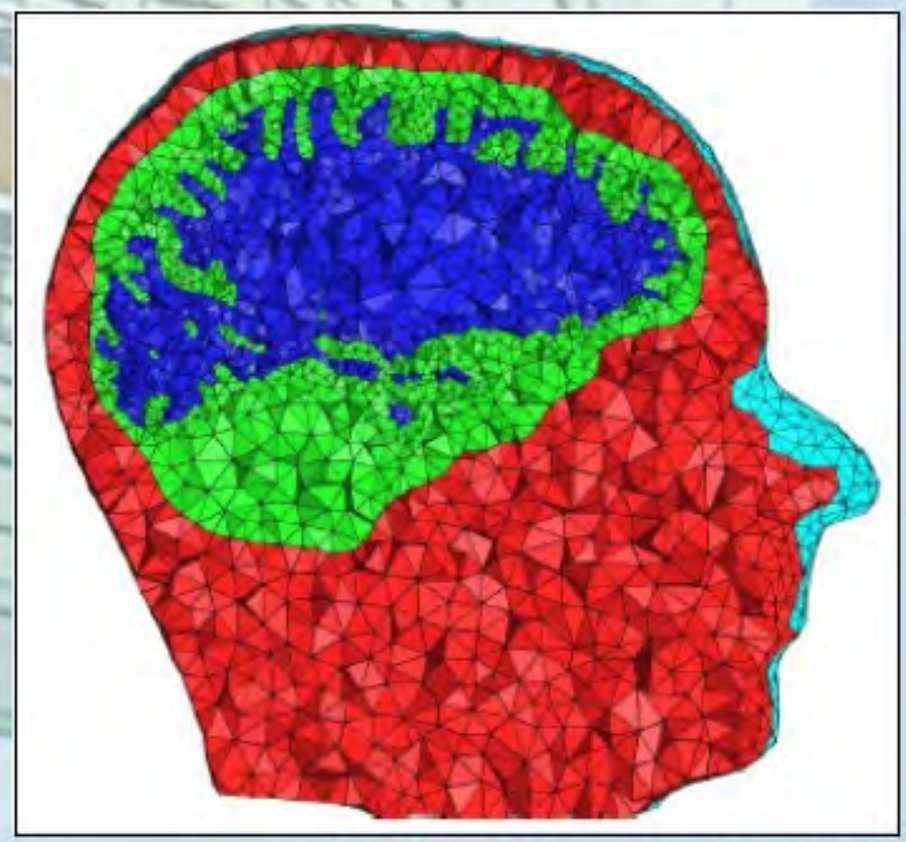
Flexibility of Domain Discretization

- Discretize the domain by elements of arbitrary type and of possible different sizes
  - 1D: intervals
  - 2D: squares, rectangles, triangles
  - 3D: hexahedra, prisms, tetrahedra



Flexibility of Domain Discretization

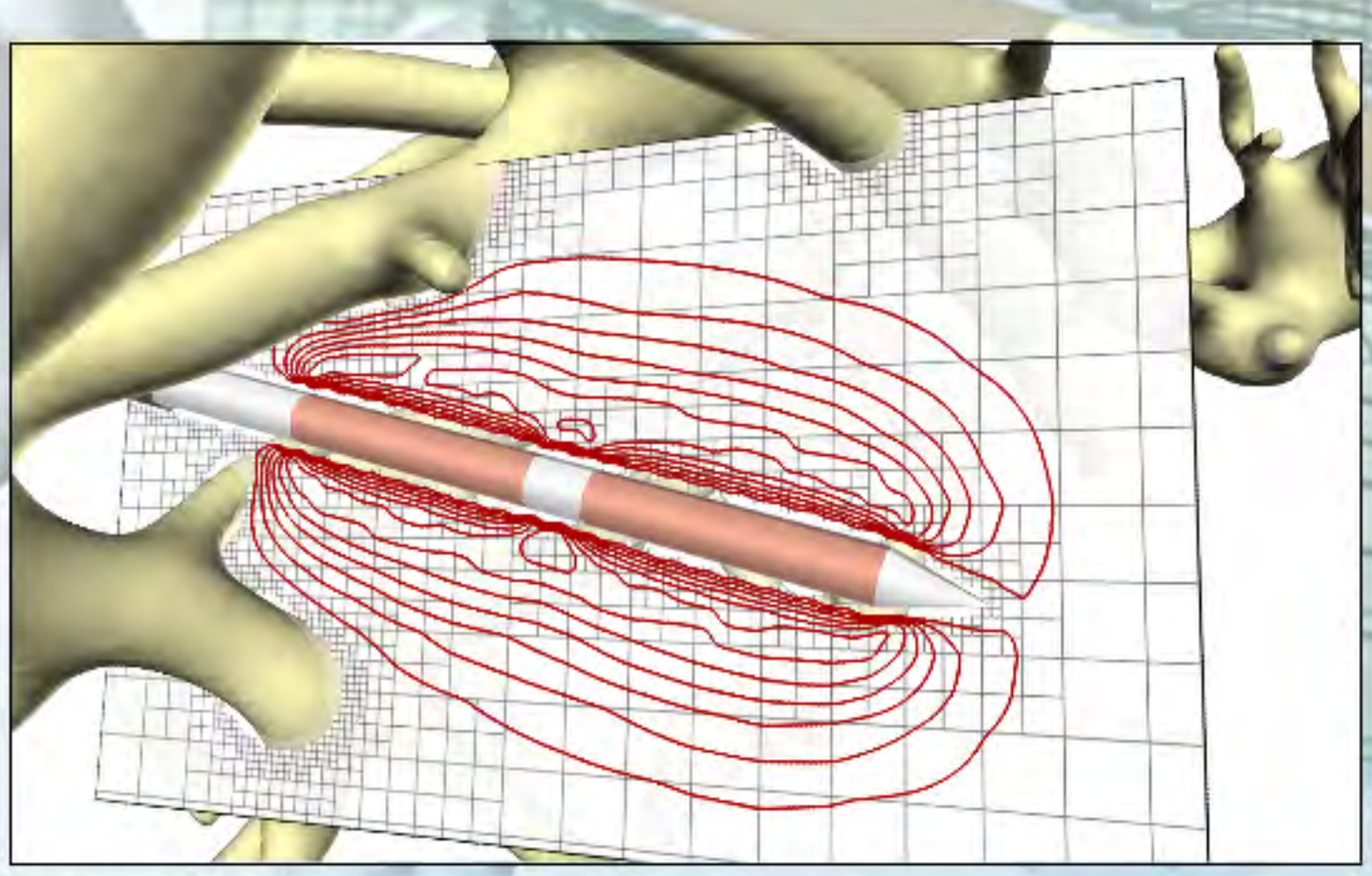
- Discretize the domain by elements of arbitrary type and of possible different sizes
  - 1D: intervals
  - 2D: squares, rectangles, triangles
  - 3D: hexahedra, prisms, tetrahedra

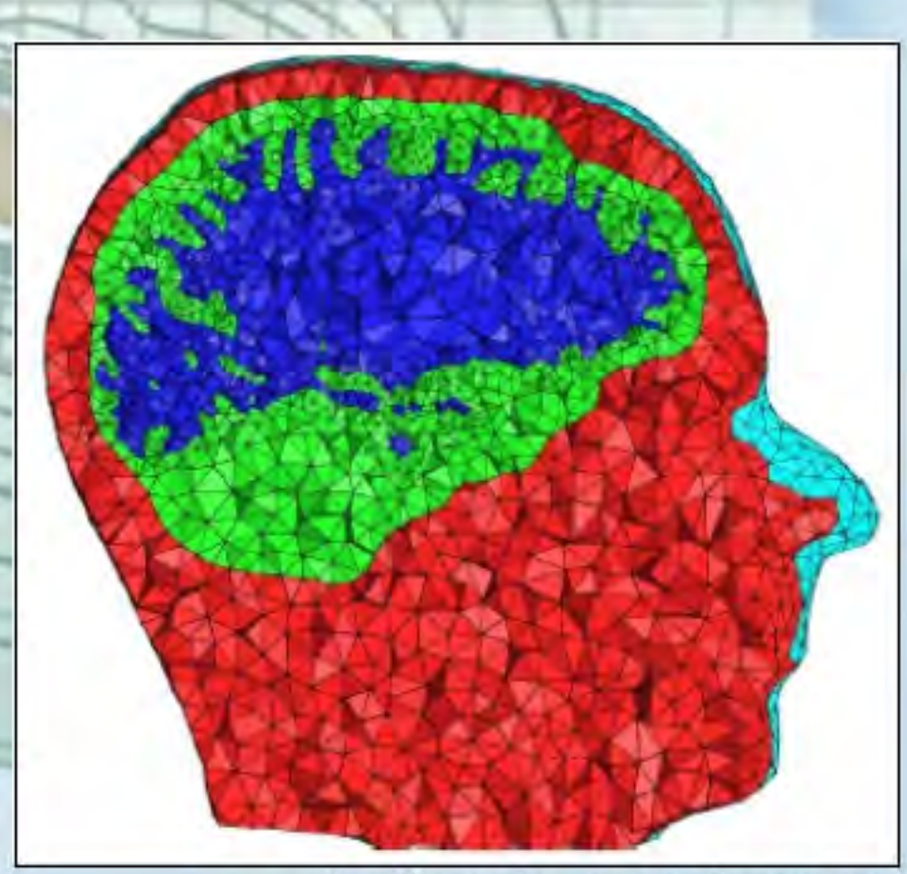


iso2mesh.sourceforge.net

Flexibility of Domain Discretization

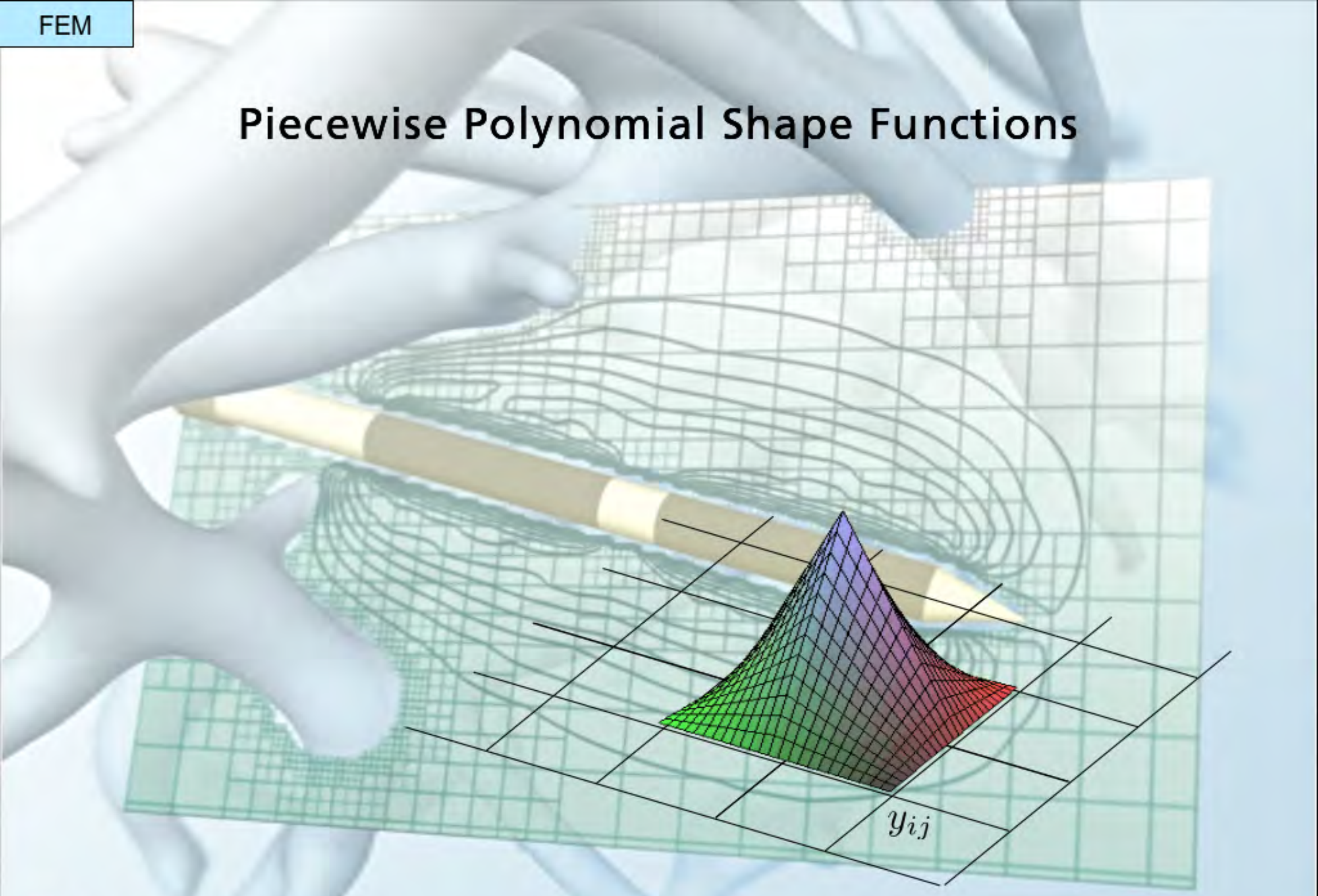
- Discretize the domain by elements of arbitrary type and of possible different sizes
  - 1D: intervals
  - 2D: squares, rectangles, triangles
  - 3D: hexahedra, prisms, tetrahedra





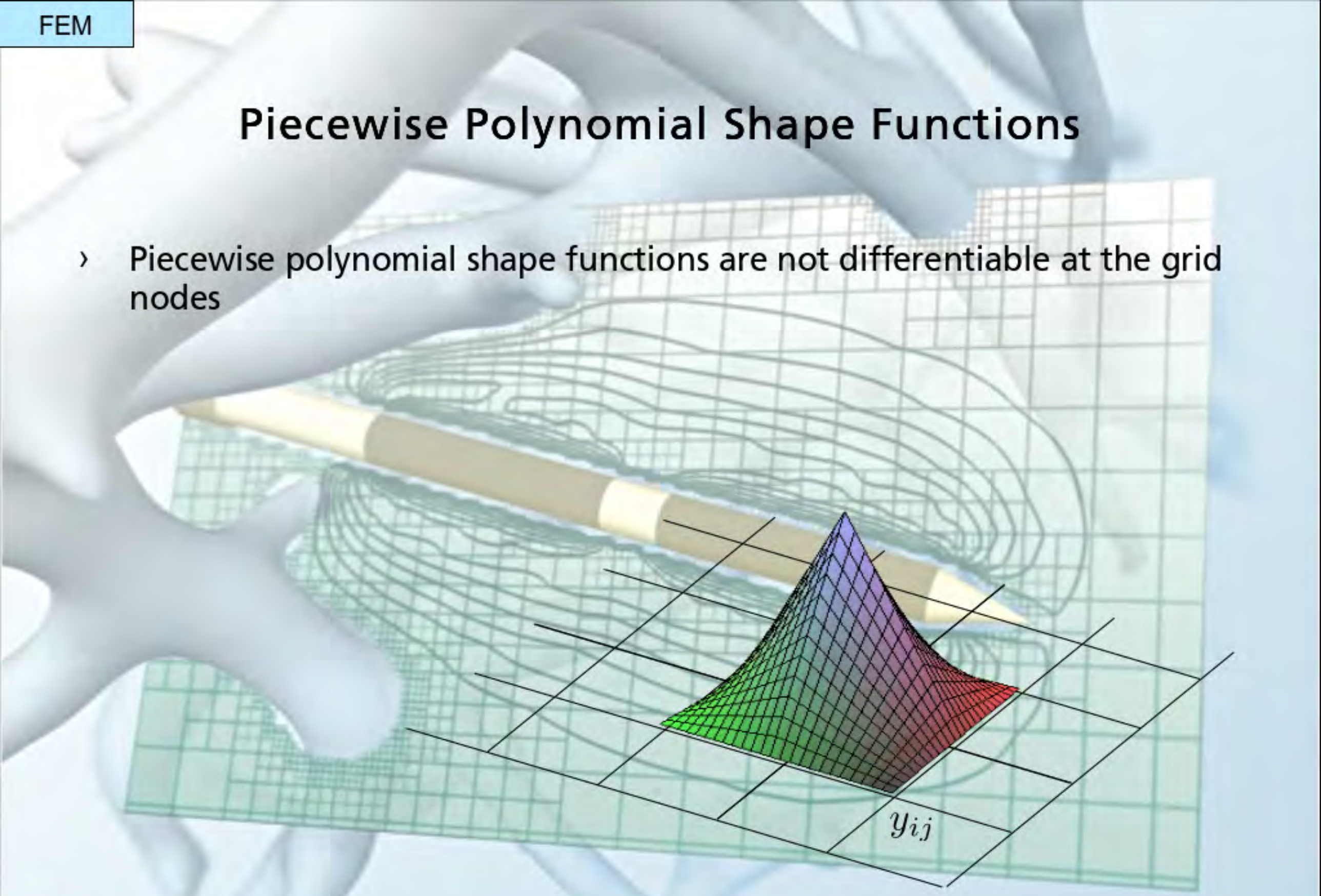
iso2mesh.sourceforge.net

Piecewise Polynomial Shape Functions



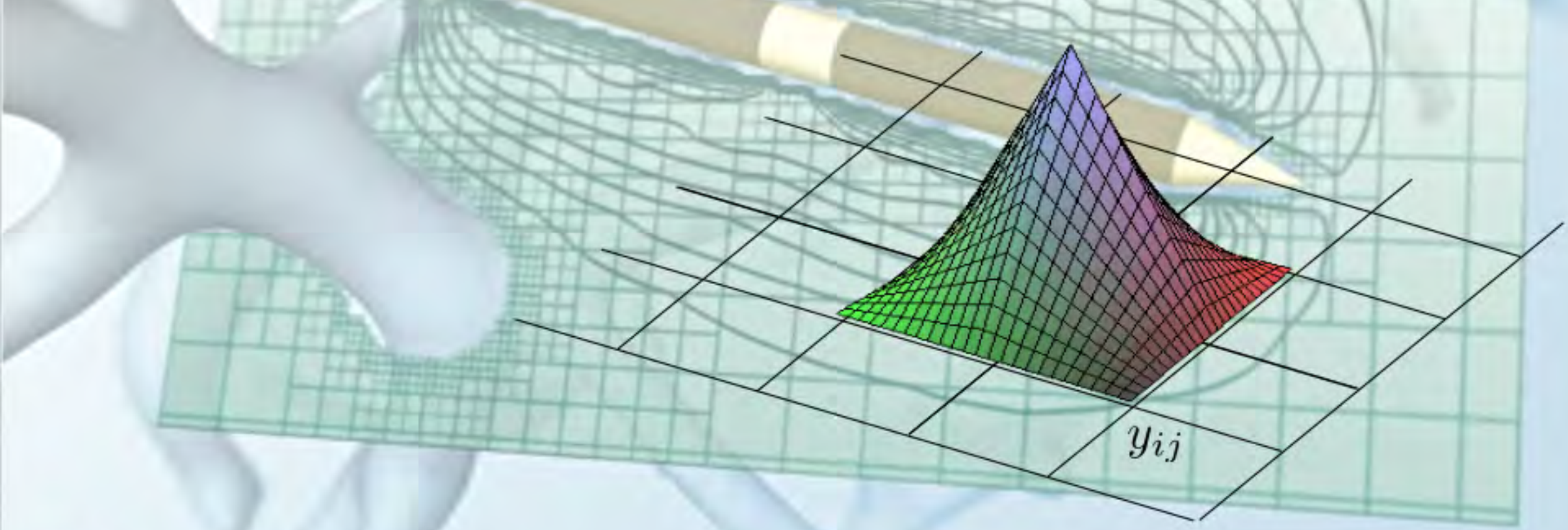
Piecewise Polynomial Shape Functions

- Piecewise polynomial shape functions are not differentiable at the grid nodes



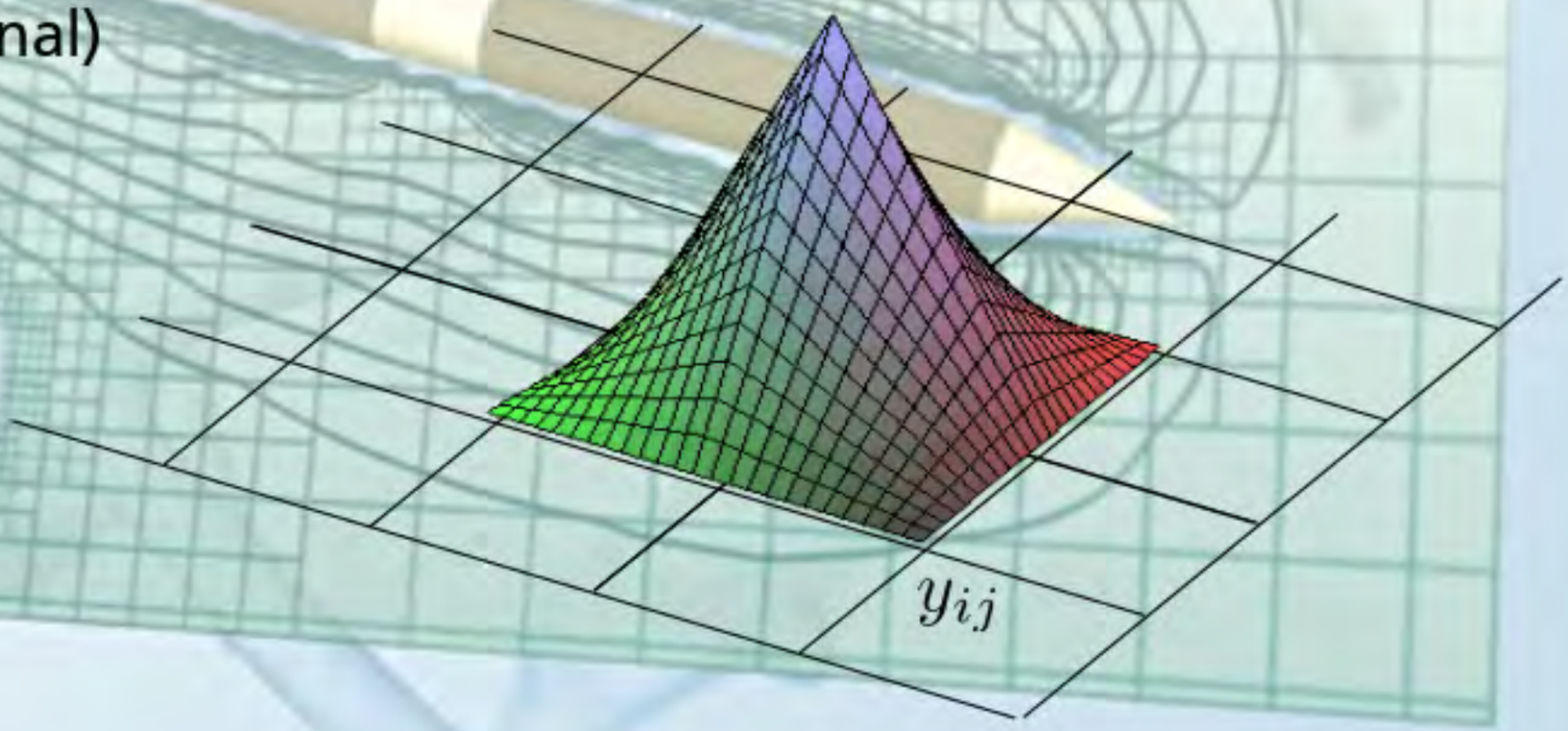
Piecewise Polynomial Shape Functions

- Piecewise polynomial shape functions are not differentiable at the grid nodes
- Need a different notion of differentiability



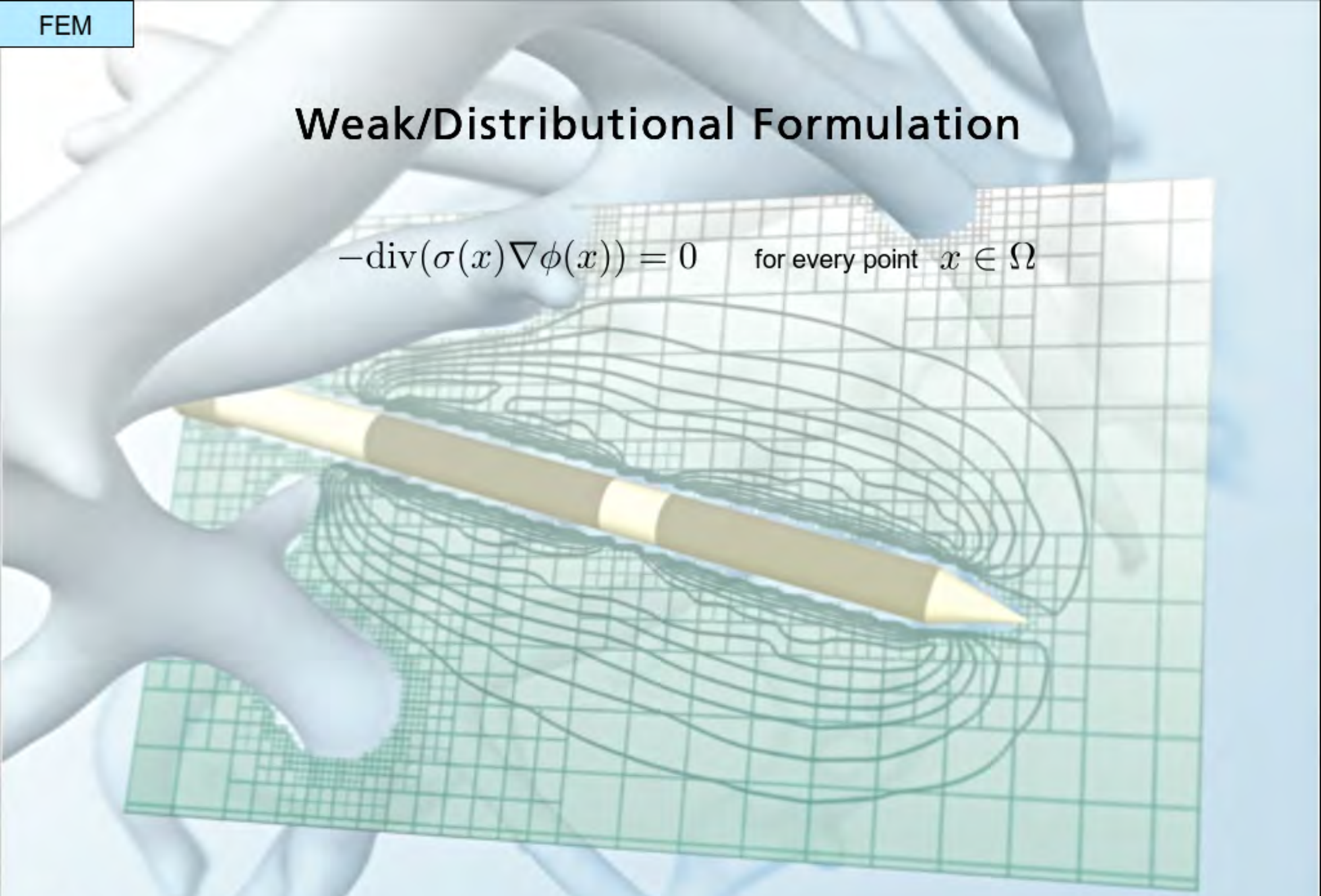
Piecewise Polynomial Shape Functions

- Piecewise polynomial shape functions are not differentiable at the grid nodes
- Need a different notion of differentiability
- Weak (distributional) derivative



Weak/Distributional Formulation

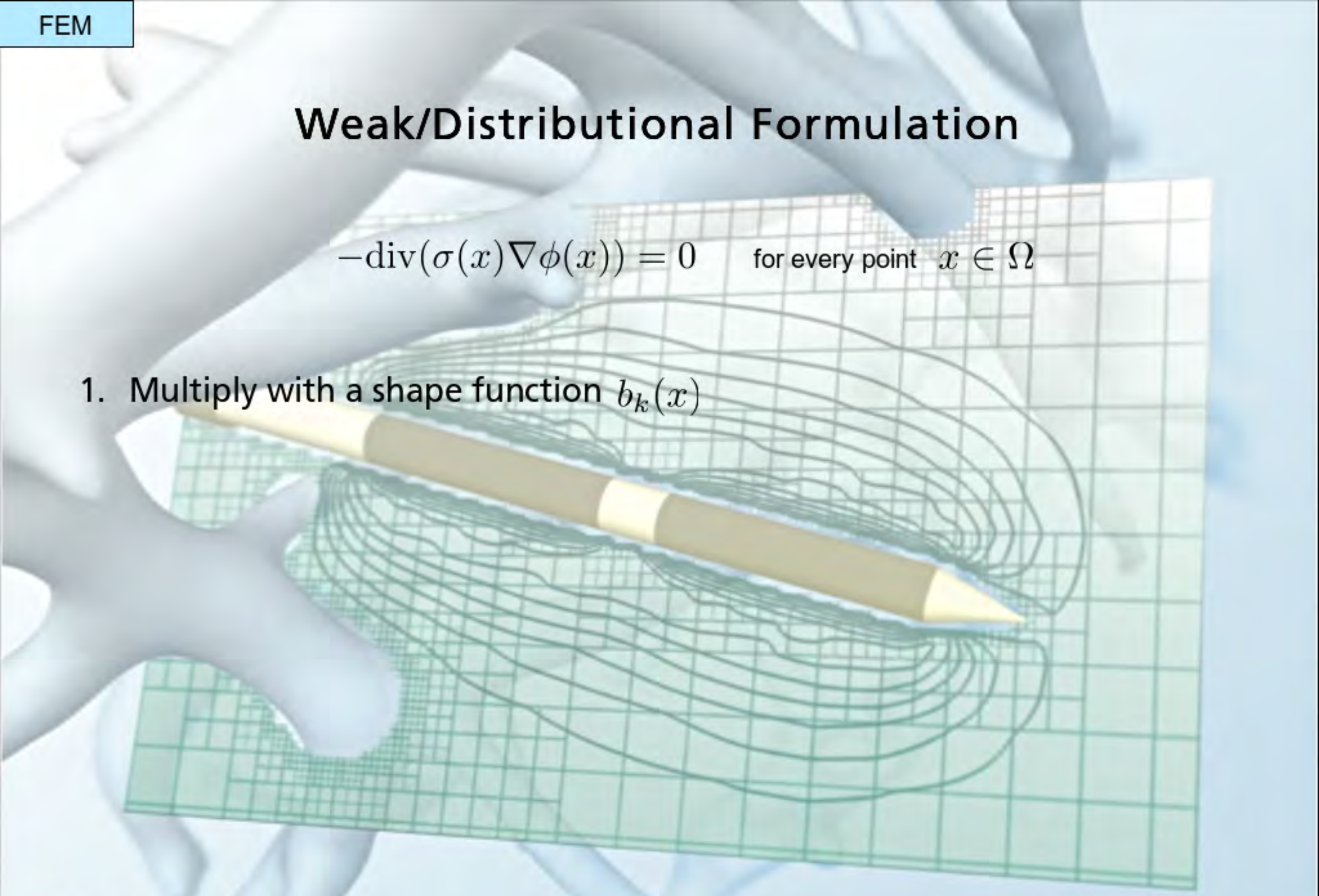
$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$



Weak/Distributional Formulation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$

1. Multiply with a shape function $b_k(x)$



Weak/Distributional Formulation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$

1. Multiply with a shape function $b_k(x)$
2. Intergrate over the domain $\int_\Omega dx$

Weak/Distributional Formulation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$

1. Multiply with a shape function $b_k(x)$
2. Intergrate over the domain $\int_\Omega dx$
3. Use integration by parts (Green's identity)

$$\int_\Omega \sigma(x)\nabla\phi(x)\cdot\nabla b_k(x)\,dx = f \qquad \text{for every shape function } b_k(x)$$

Weak/Distributional Formulation

$$-\mathrm{div}(\sigma(x)\nabla\phi(x)) = 0 \qquad \text{for every point } x \in \Omega$$

1. Multiply with a shape function $b_k(x)$
2. Intergrate over the domain $\int_\Omega dx$
3. Use integration by parts (Green's identity)

$$\int_\Omega \sigma(x)\nabla\phi(x)\cdot\nabla b_k(x)\,dx = f \qquad \text{for every shape function } b_k(x)$$

$$\int_\Omega \sigma(x)\Big(\sum_i \phi_i \nabla b_i(x)\Big)\cdot\nabla b_k(x)\,dx = f$$

Systems of Linear Equations

$$\int_\Omega \sigma(x)\Big(\sum_i \phi_i \nabla b_i(x)\Big) \cdot \nabla b_k(x)\, dx = f$$

for every shape function $b_k(x)$

› Matrix

$$A_{ik} = \int_\Omega \sigma \nabla \phi_i(x) \cdot \nabla \phi_j(x)\, dx$$

Systems of Linear Equations

$$\int_\Omega \sigma(x)\Big(\sum_i \phi_i \nabla b_i(x)\Big) \cdot \nabla b_k(x)\, dx = f$$

for every shape function $b_k(x)$

- Discrete equation

$$\sum_i \alpha_{i,k}\phi_i = f$$ for every shape function $b_k(x)$

- Matrix

$$A_{ik} = \int_\Omega \sigma \nabla \phi_i(x) \cdot \nabla \phi_j(x)\, dx$$

Systems of Linear Equations

$$\int_\Omega \sigma(x)\Big(\sum_i \phi_i \nabla b_i(x)\Big) \cdot \nabla b_k(x)\, dx = f$$

for every shape function $b_k(x)$

- Discrete equation

$$\sum_i \alpha_{i,k}\phi_i = f \qquad \text{for every shape function } b_k(x)$$

- Linear system of equations

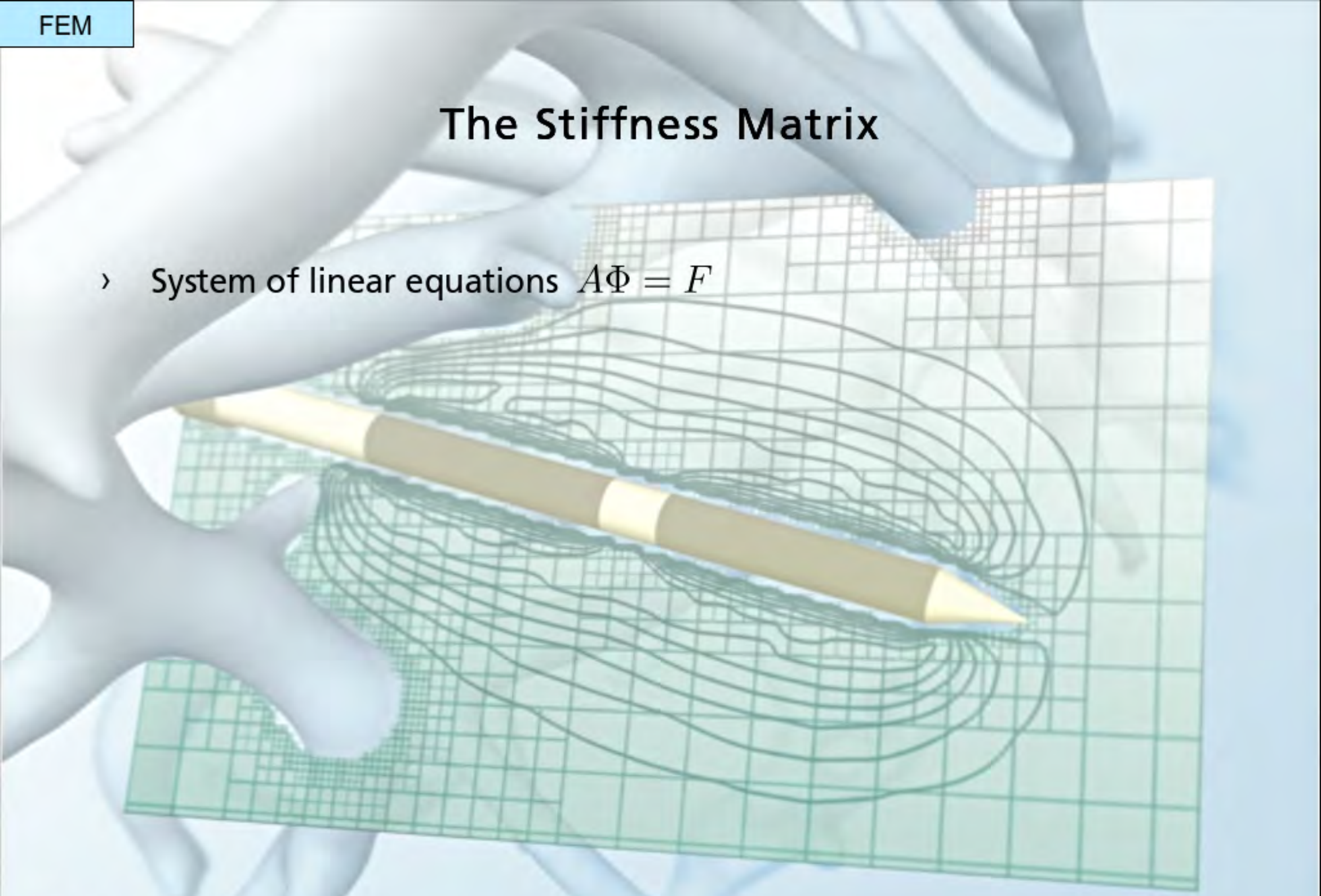
$$A\Phi = F \qquad A \in I\!R^{(N+1)\times(N+1)} \qquad \Phi, F \in I\!R^{(N+1)}$$

- Matrix

$$A_{ik} = \int_\Omega \sigma \nabla\phi_i(x) \cdot \nabla\phi_j(x)\, dx$$

The Stiffness Matrix

- System of linear equations $A\Phi = F$



The Stiffness Matrix

- System of linear equations $A\Phi = F$
- Entries of Stiffness Matrix A

$$A_{ik} = \int_\Omega \sigma \nabla \phi_i(x) \cdot \nabla \phi_j(x)\, dx$$

compute these entries by traversing through the elements of the grid

The Stiffness Matrix

- System of linear equations $A\Phi = F$
- Entries of Stiffness Matrix A

$$A_{ik} = \int_\Omega \sigma \nabla \phi_i(x) \cdot \nabla \phi_j(x)\, dx$$

compute these entries by traversing through the elements of the grid

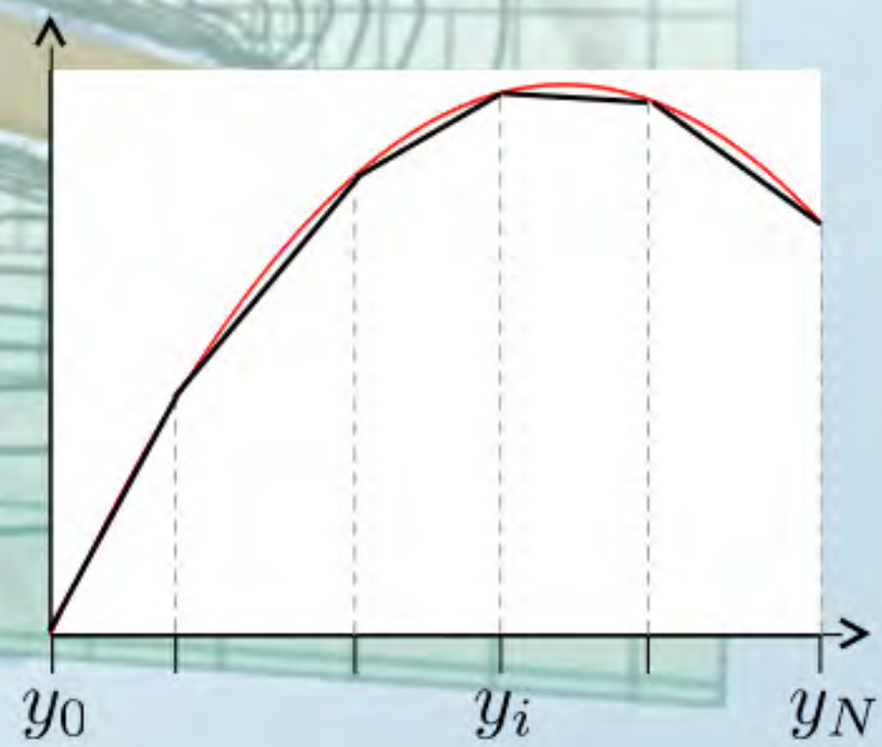
- Matrix inversion:
  - Iterative Method, e.g. Gauss–Seidel, conjugate gradient method
  - But also: Multigrid method

Summary FEM

- Compose functions by shape functions
- Shape functions = piecewise polynomial functions
- Discretization of domain by simplices of variable shape and size
- Weak formulation of differential equation
- Solve by iterative or direct method
- Obtain a numerical solution ϕ_h of the PDE

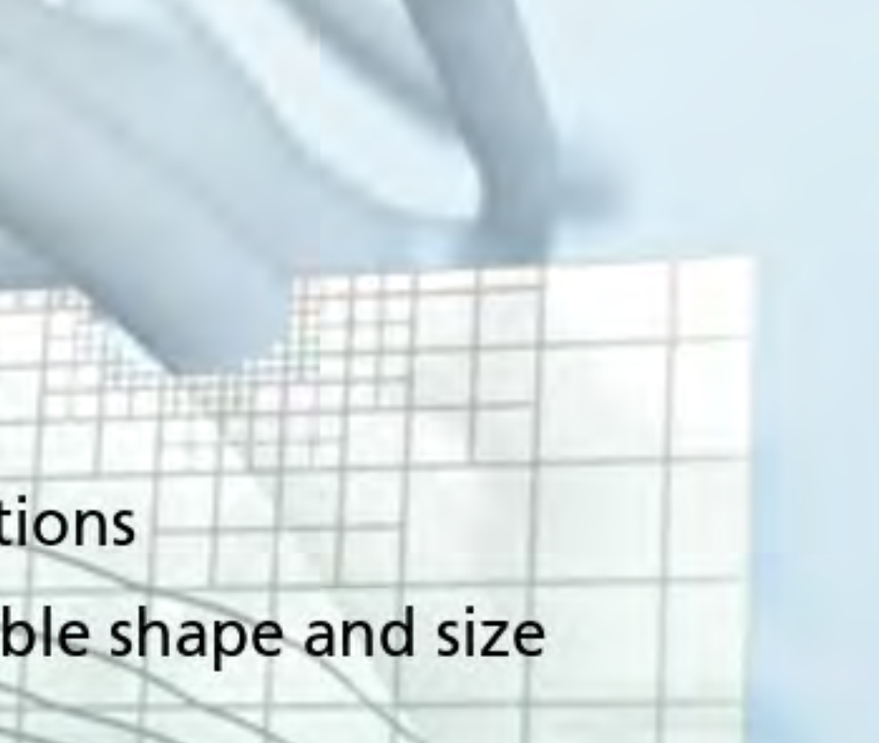
Summary FEM

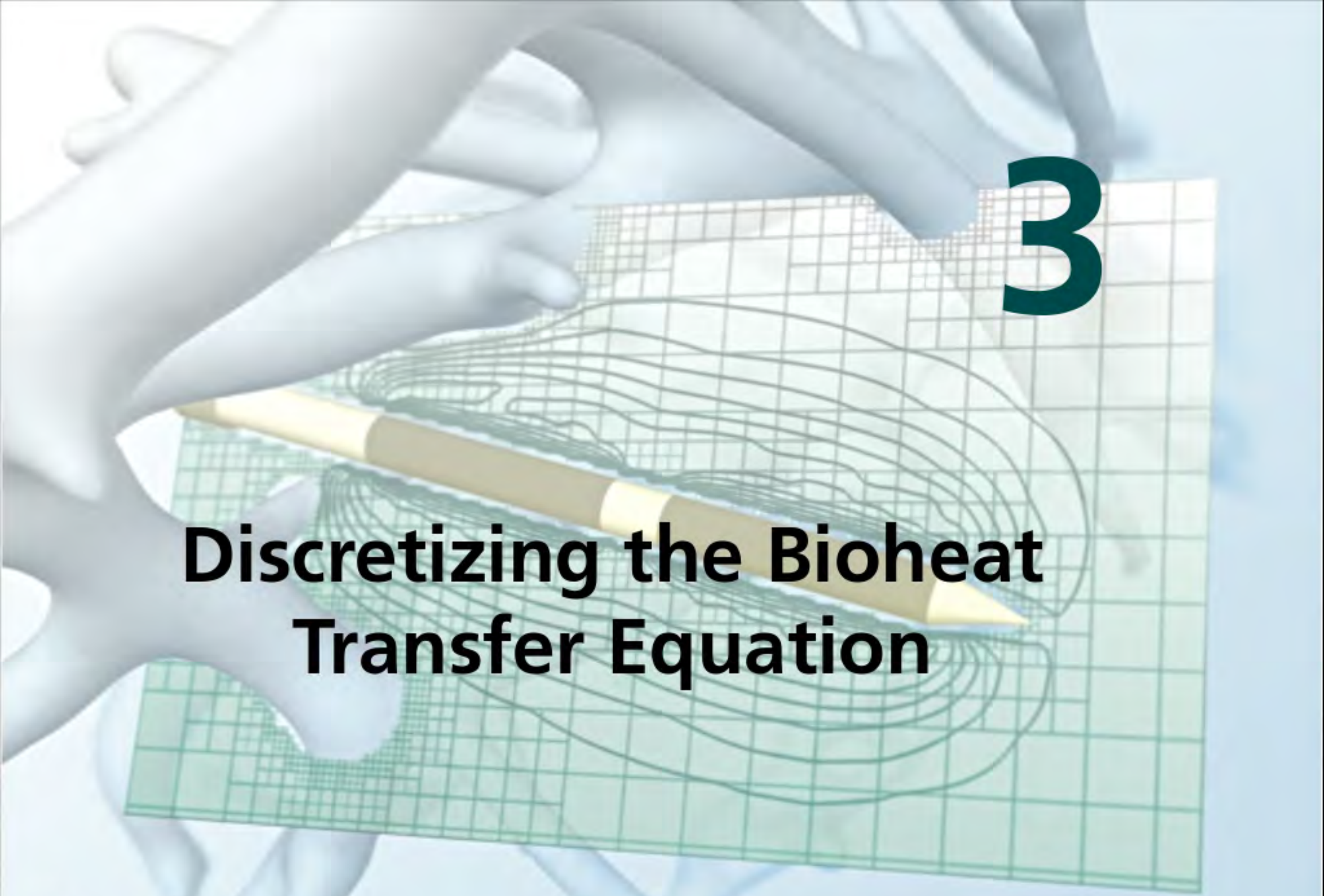
- Compose functions by shape functions
- Shape functions = piecewise polynomial functions
- Discretization of domain by simplices of variable shape and size
- Weak formulation of differential equation
- Solve by iterative or direct method
- Obtain a numerical solution ϕ_h of the PDE



Summary FEM

- Compose functions by shape functions
- Shape functions = piecewise polynomial functions
- Discretization of domain by simplices of variable shape and size
- Weak formulation of differential equation
- Solve by iterative or direct method
- Obtain a numerical solution ϕ_h of the PDE
- Accuracy $\|\phi - \phi_h\|_{L^2} = Ch^{p+1}$



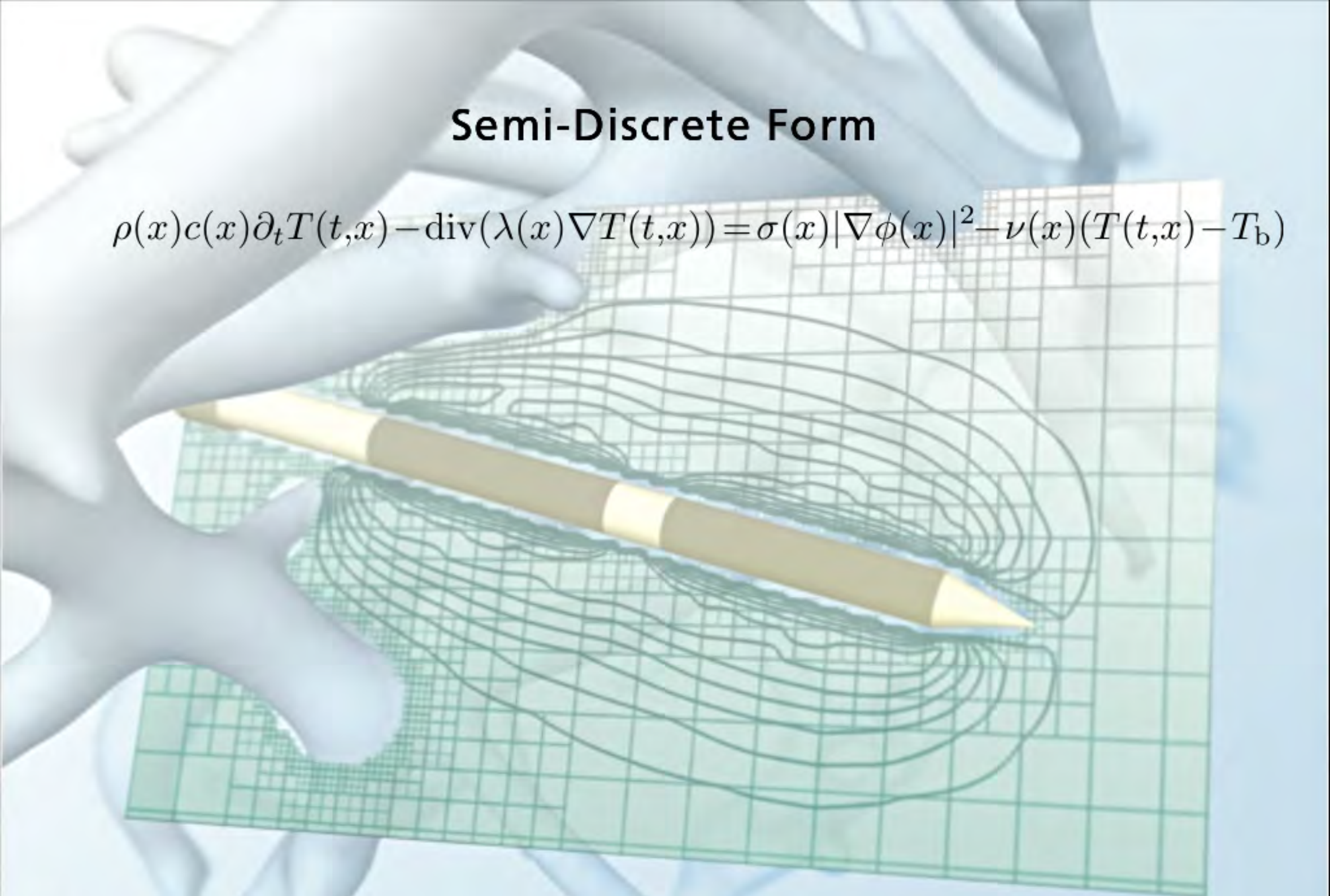


3

Discretizing the Bioheat Transfer Equation

Semi-Discrete Form

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$



Semi-Discrete Form

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

Simplification

$$\partial_t T - \mathrm{div}(\lambda \nabla T) = 0$$

Semi-Discrete Form

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

Simplification

$$\partial_t T - \mathrm{div}(\lambda \nabla T) = 0$$

FDM or FEM discretization

$$\partial_t T - AT = 0$$

Semi-Discrete Form

$$\rho(x)c(x)\partial_t T(t,x) - \mathrm{div}(\lambda(x)\nabla T(t,x)) = \sigma(x)|\nabla\phi(x)|^2 - \nu(x)(T(t,x) - T_\mathrm{b})$$

Simplification

$$\partial_t T - \mathrm{div}(\lambda \nabla T) = 0$$

FDM or FEM discretization

$$\partial_t T - AT = 0$$

$$\dot{T} - AT = 0$$

Matrix Exponentials

- Ordinary differential equation in the nodal values

$$\dot{T} - AT = 0$$

Matrix Exponentials

- Ordinary differential equation in the nodal values

$$\dot{T} - AT = 0$$

- Solution $T(t) = \exp(tA)\, T_0$

Note that this is the exponential function for matrices!

Matrix Exponentials

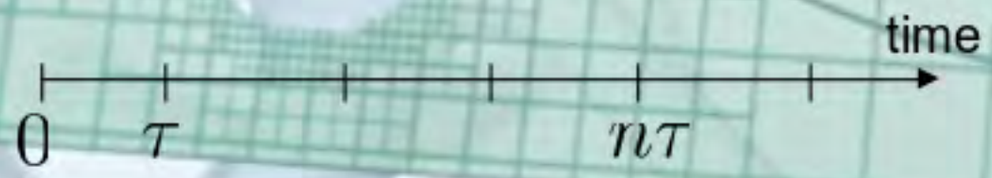
- Ordinary differential equation in the nodal values

$$\dot{T} - AT = 0$$

- Solution $T(t) = \exp(tA)\, T_0$

  Note that this is the exponential function for matrices!

- Go small time steps of size tau



Matrix Exponentials

- Ordinary differential equation in the nodal values

$$\dot{T} - AT = 0$$

- Solution $T(t) = \exp(tA)\, T_0$

  Note that this is the exponential function for matrices!

- Go small time steps of size tau

Matrix Exponentials

- Ordinary differential equation in the nodal values

$$\dot{T} - AT = 0$$

- Solution $T(t) = \exp(tA)\, T_0$

  Note that this is the exponential function for matrices!

- Go small time steps of size tau $\exp(n\tau A) = \exp(\tau A) \cdots \exp(\tau A)$

Matrix Exponentials

- Ordinary differential equation in the nodal values

$$\dot{T} - AT = 0$$

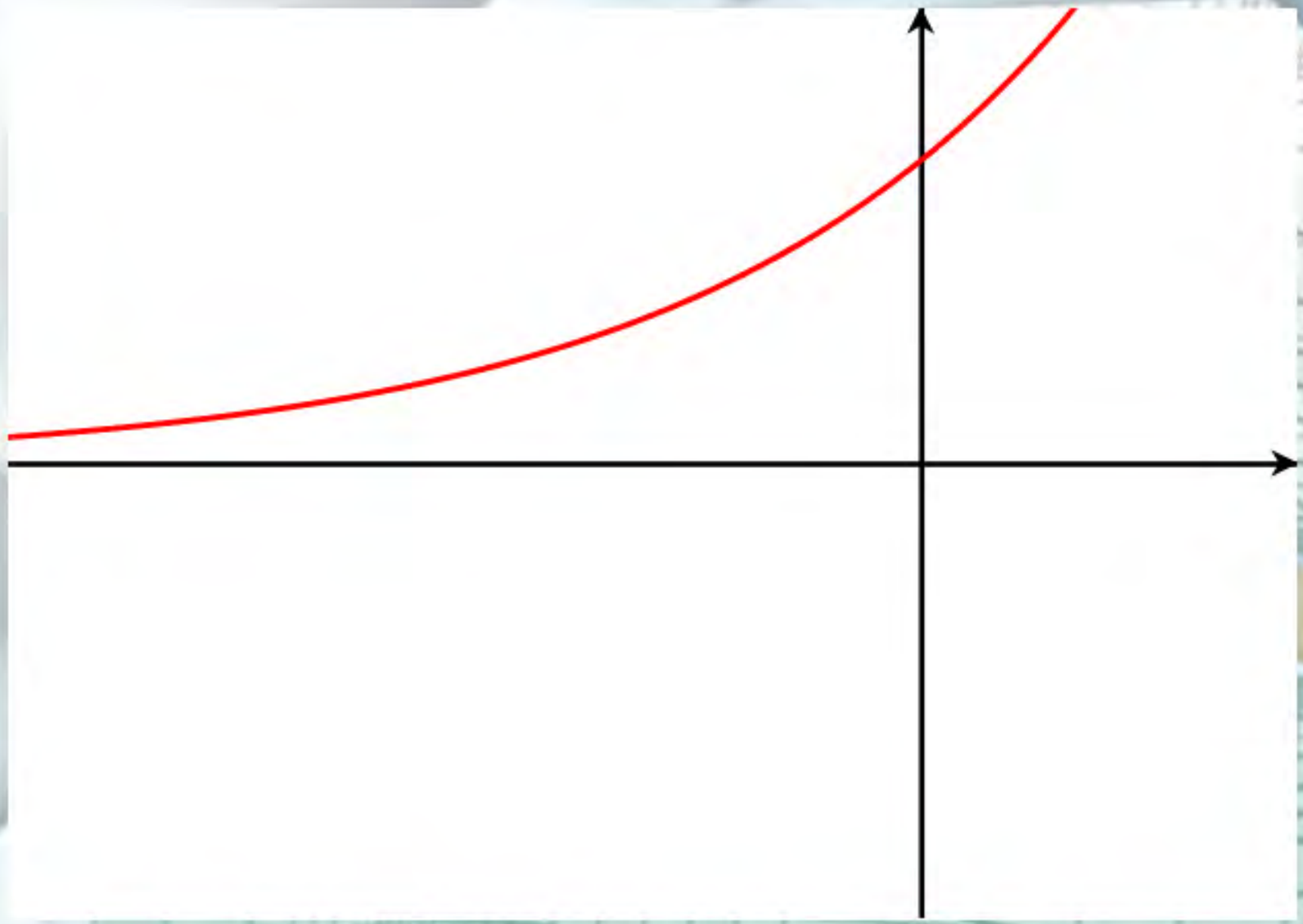
- Solution $T(t) = \exp(tA)\, T_0$

  Note that this is the exponential function for matrices!

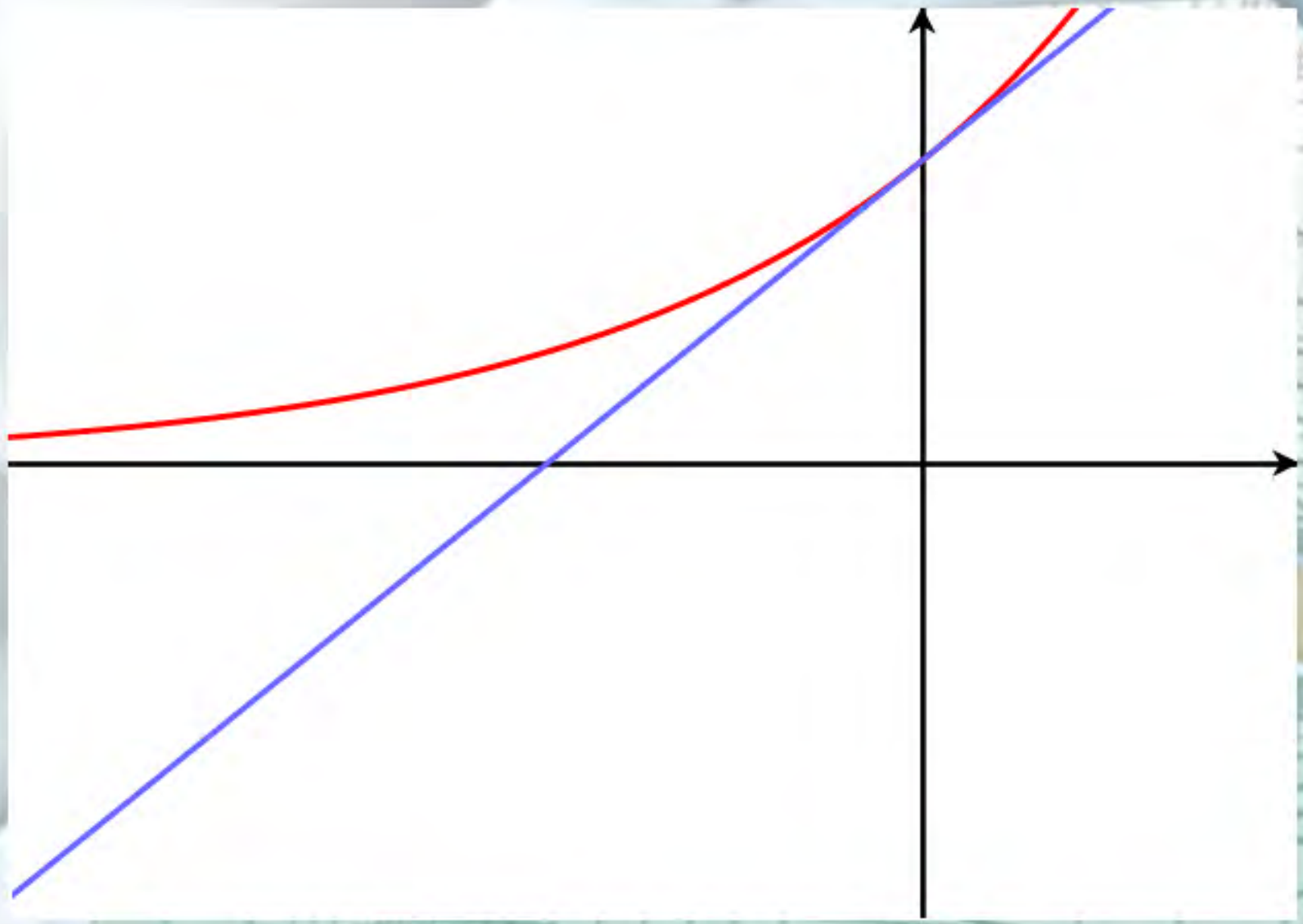
- Go small time steps of size tau $\exp(n\tau A) = \exp(\tau A) \cdots \exp(\tau A)$

$$T_{n+1} = \exp(\tau A) T_n$$

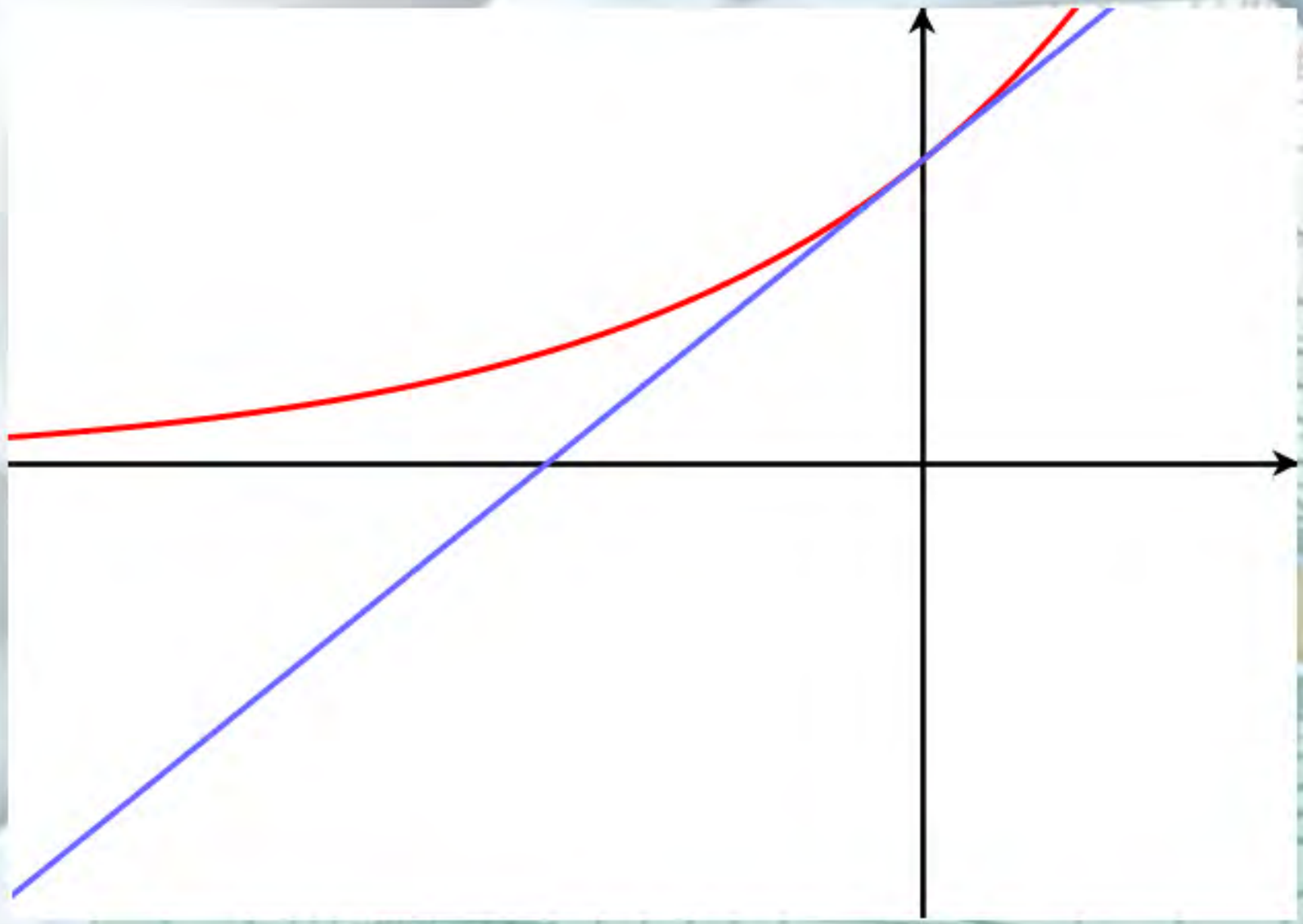
Approxmating $\exp(\tau A)$ by Rational Functions



Approxmating $\exp(\tau A)$ by Rational Functions



Approxmating $\exp(\tau A)$ by Rational Functions

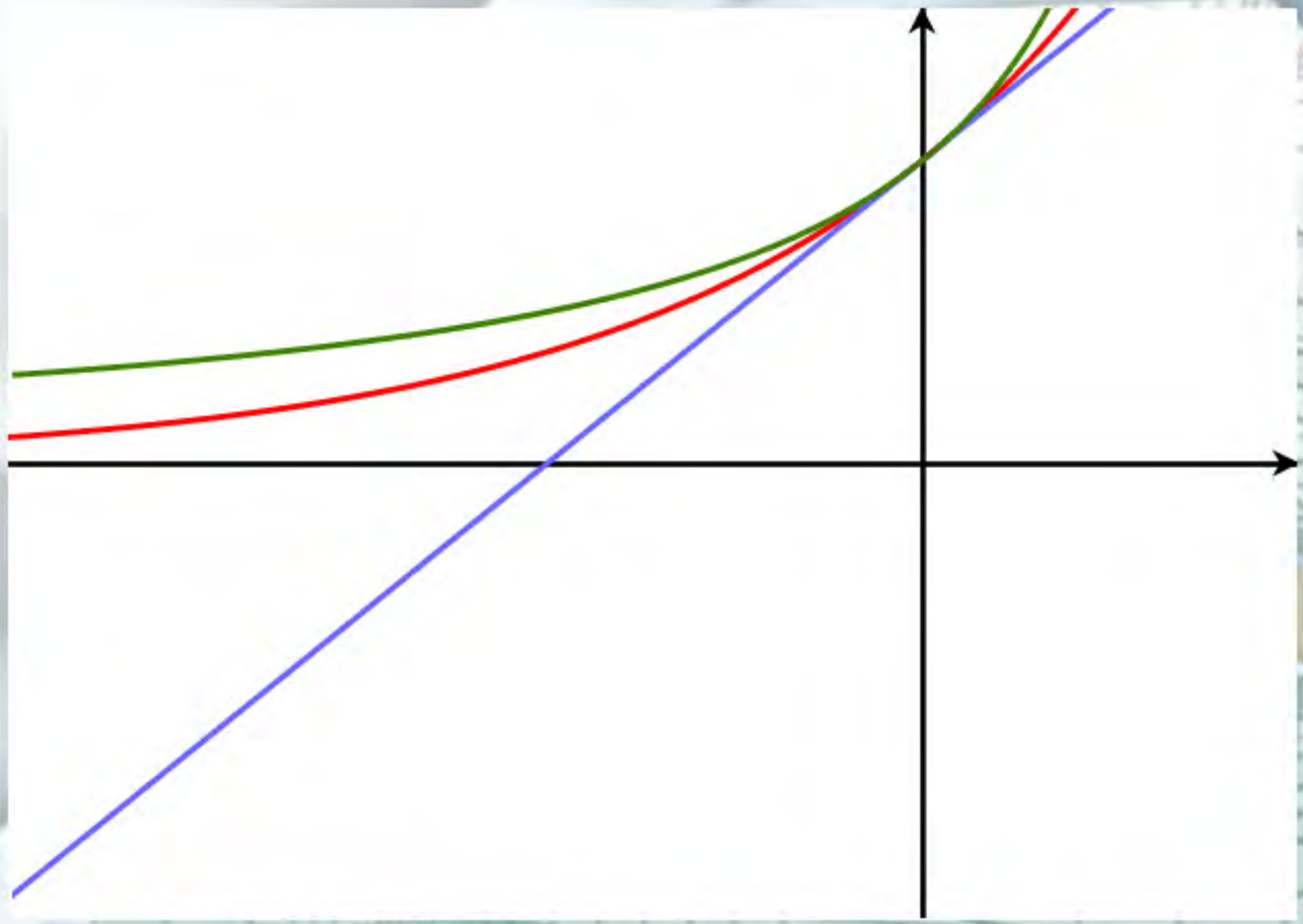


Exponential

Forward Euler

$$\exp(\tau A) \approx 1 + \tau A$$

Approxmating $\exp(\tau A)$ by Rational Functions

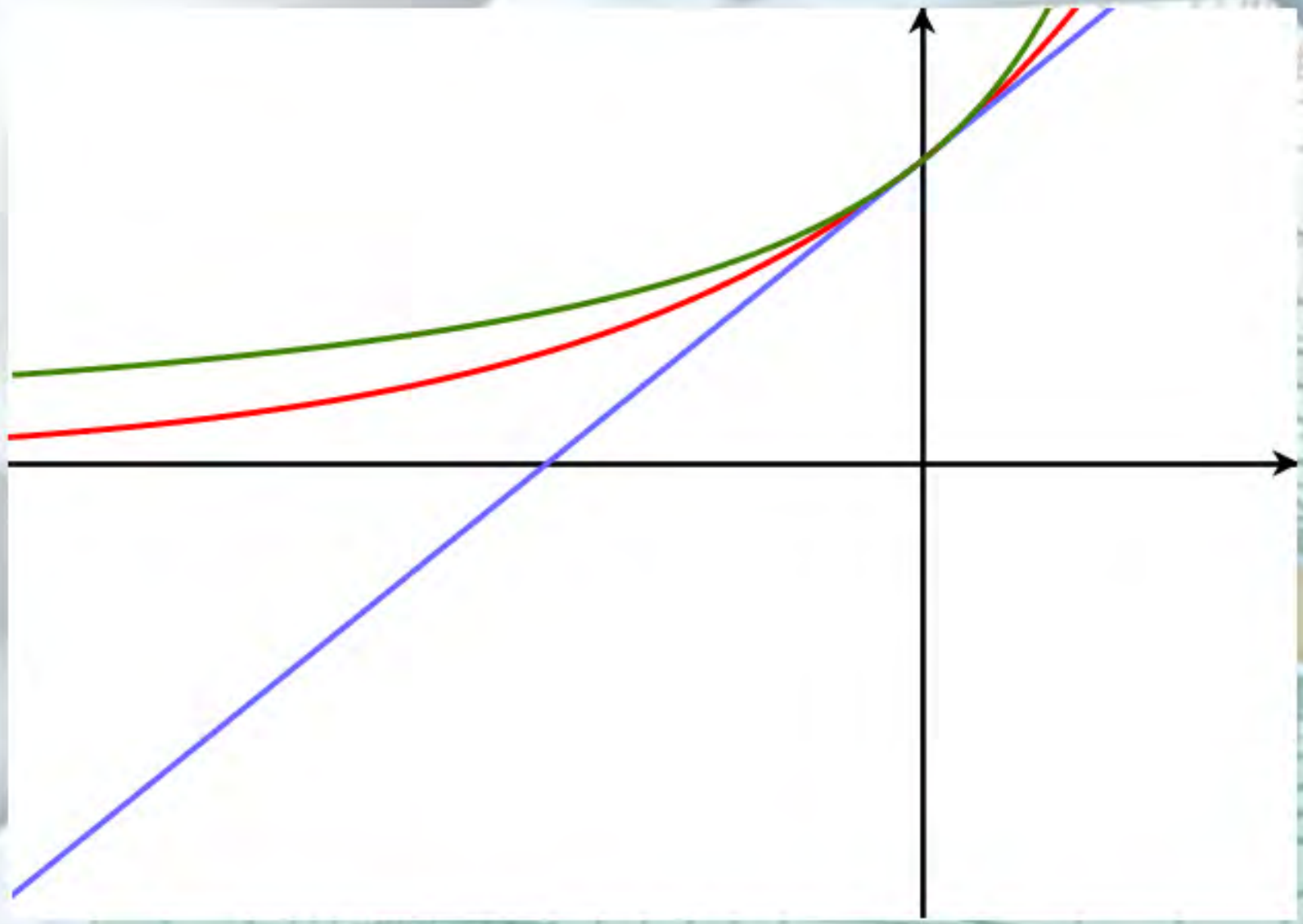


Exponential

Forward Euler

$$\exp(\tau A) \approx 1 + \tau A$$

Approxmating $\exp(\tau A)$ by Rational Functions



Exponential

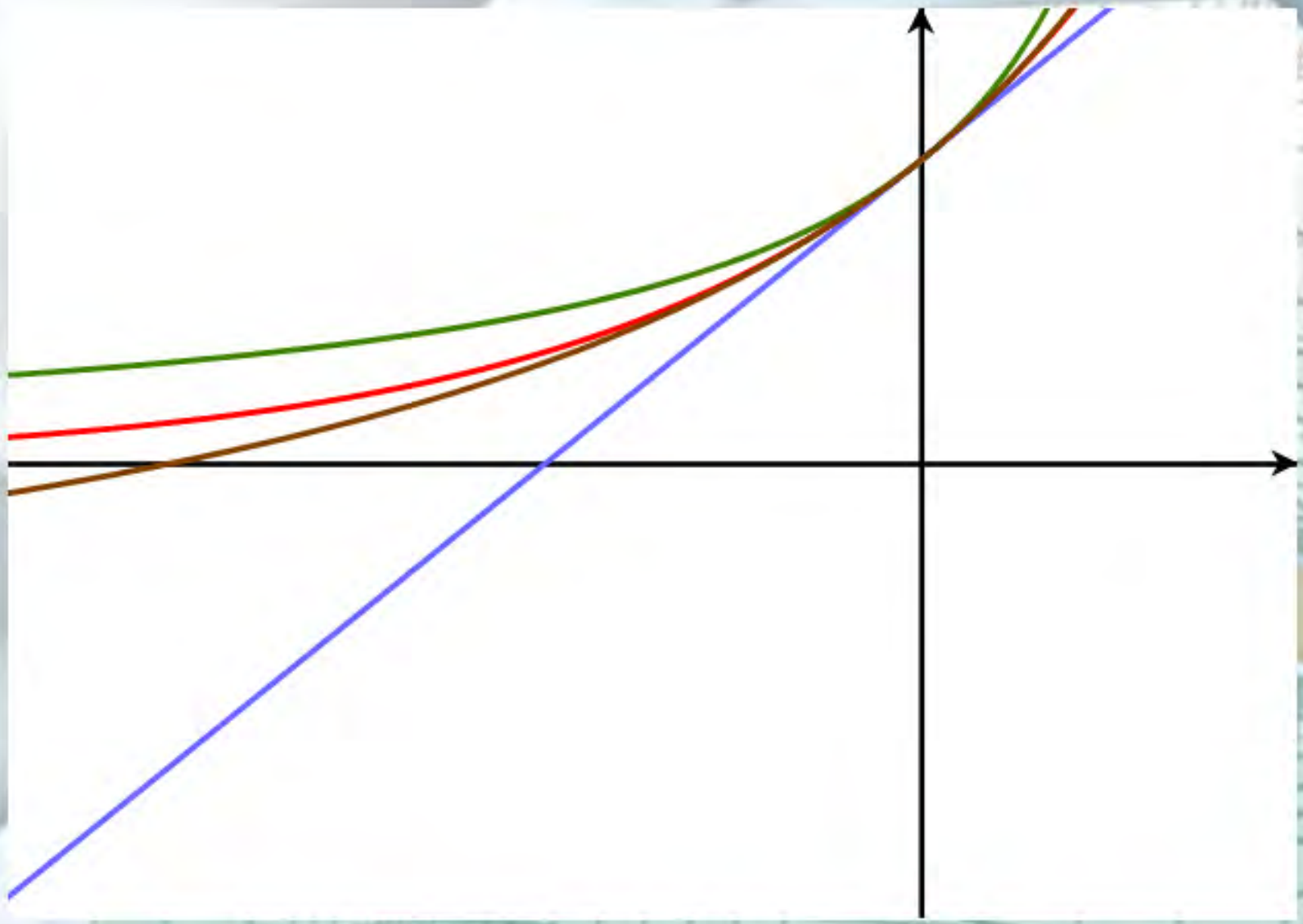
Forward Euler

$$\exp(\tau A) \approx 1 + \tau A$$

Backward Euler

$$\exp(\tau A) \approx (1 - \tau A)^{-1}$$

Approxmating $\exp(\tau A)$ by Rational Functions



Exponential

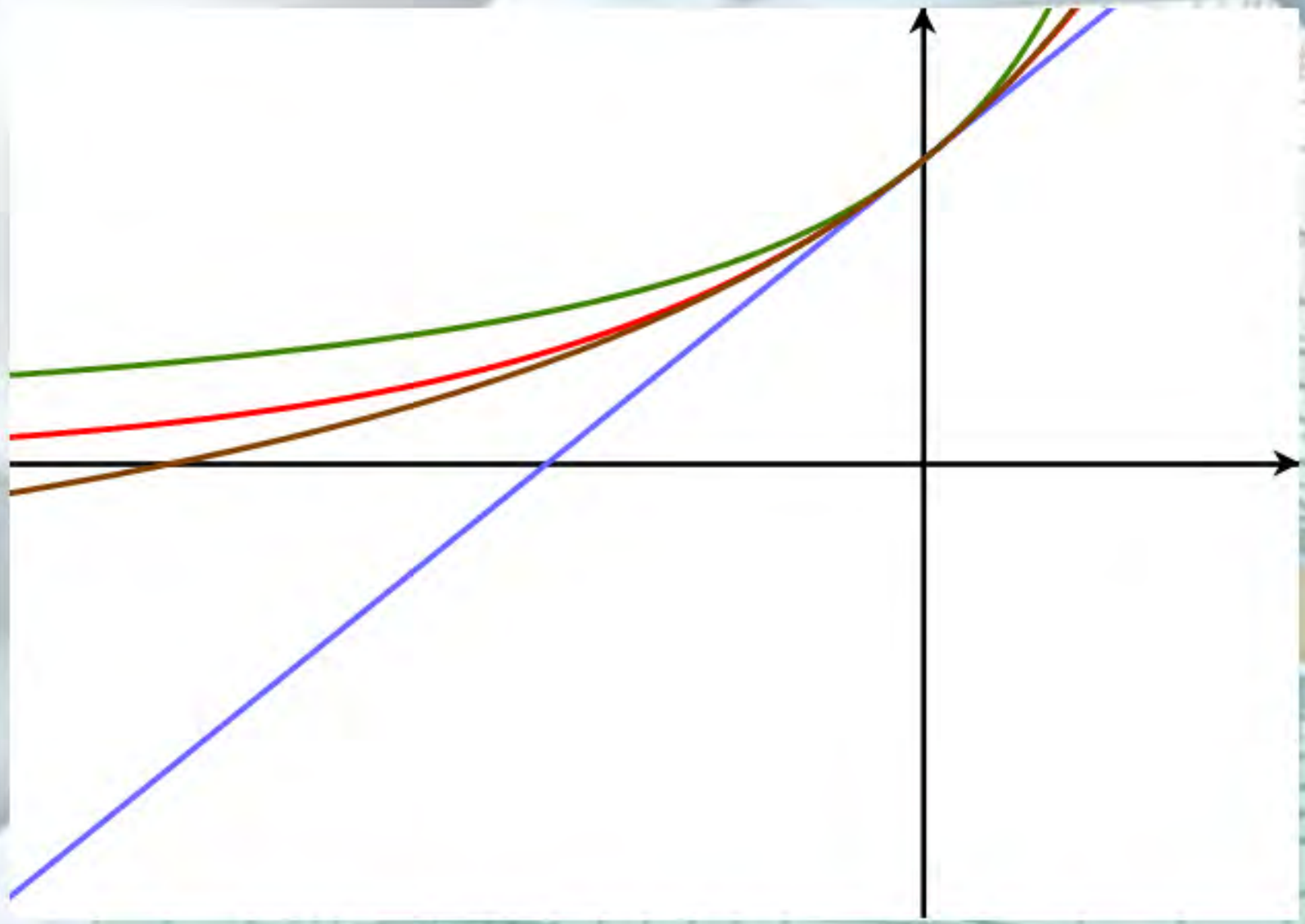
Forward Euler

$$\exp(\tau A) \approx 1 + \tau A$$

Backward Euler

$$\exp(\tau A) \approx (1 - \tau A)^{-1}$$

Approxmating $\exp(\tau A)$ by Rational Functions



Exponential

Forward Euler

$$\exp(\tau A) \approx 1 + \tau A$$

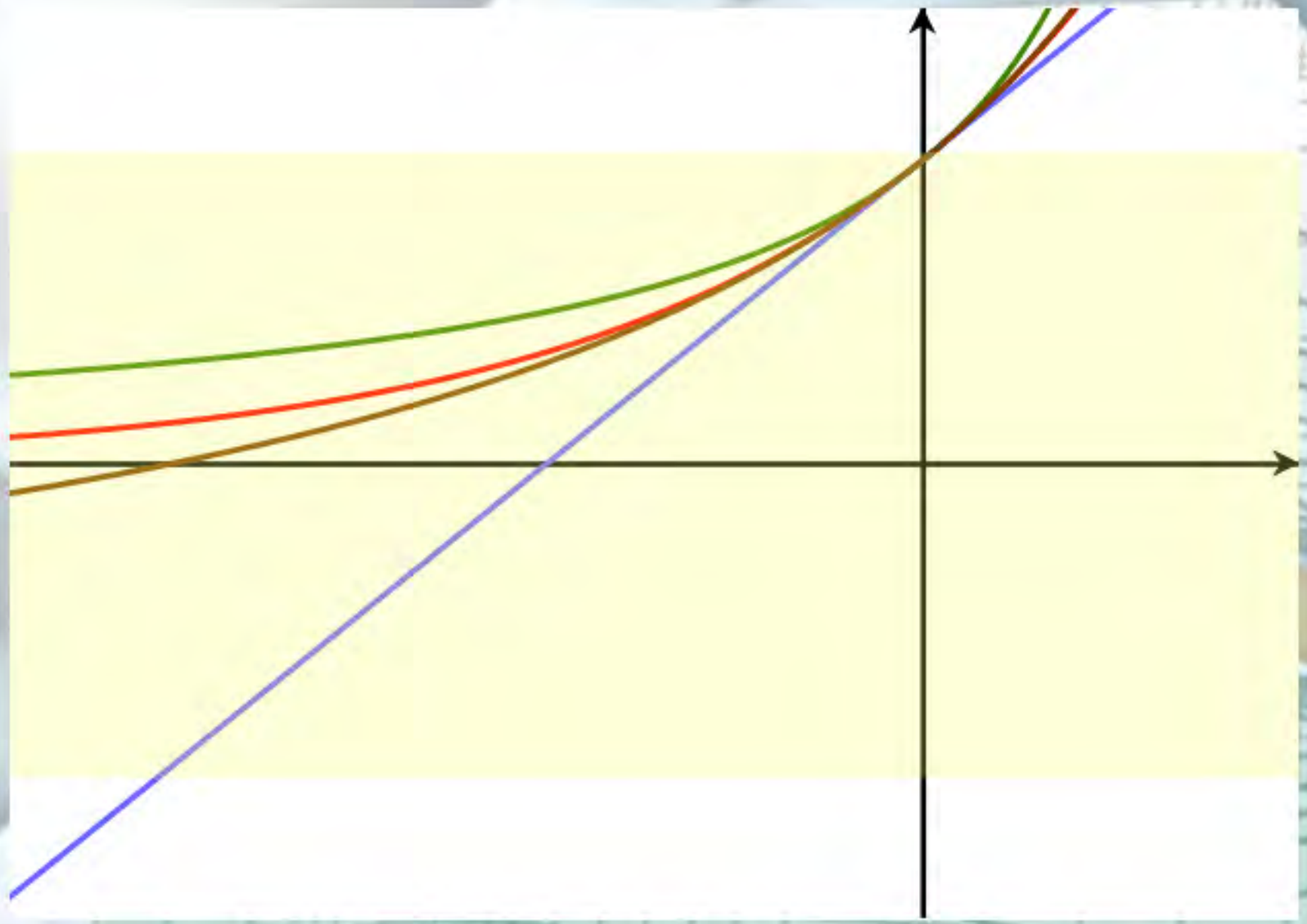
Backward Euler

$$\exp(\tau A) \approx (1 - \tau A)^{-1}$$

Crank-Nicholson

$$\exp(\tau A) \approx \left(1 + \frac{1}{2}\tau A\right)\left(1 - \frac{1}{2}\tau A\right)^{-1}$$

Approxmating $\exp(\tau A)$ by Rational Functions



Exponential

Forward Euler

$$\exp(\tau A) \approx 1 + \tau A$$

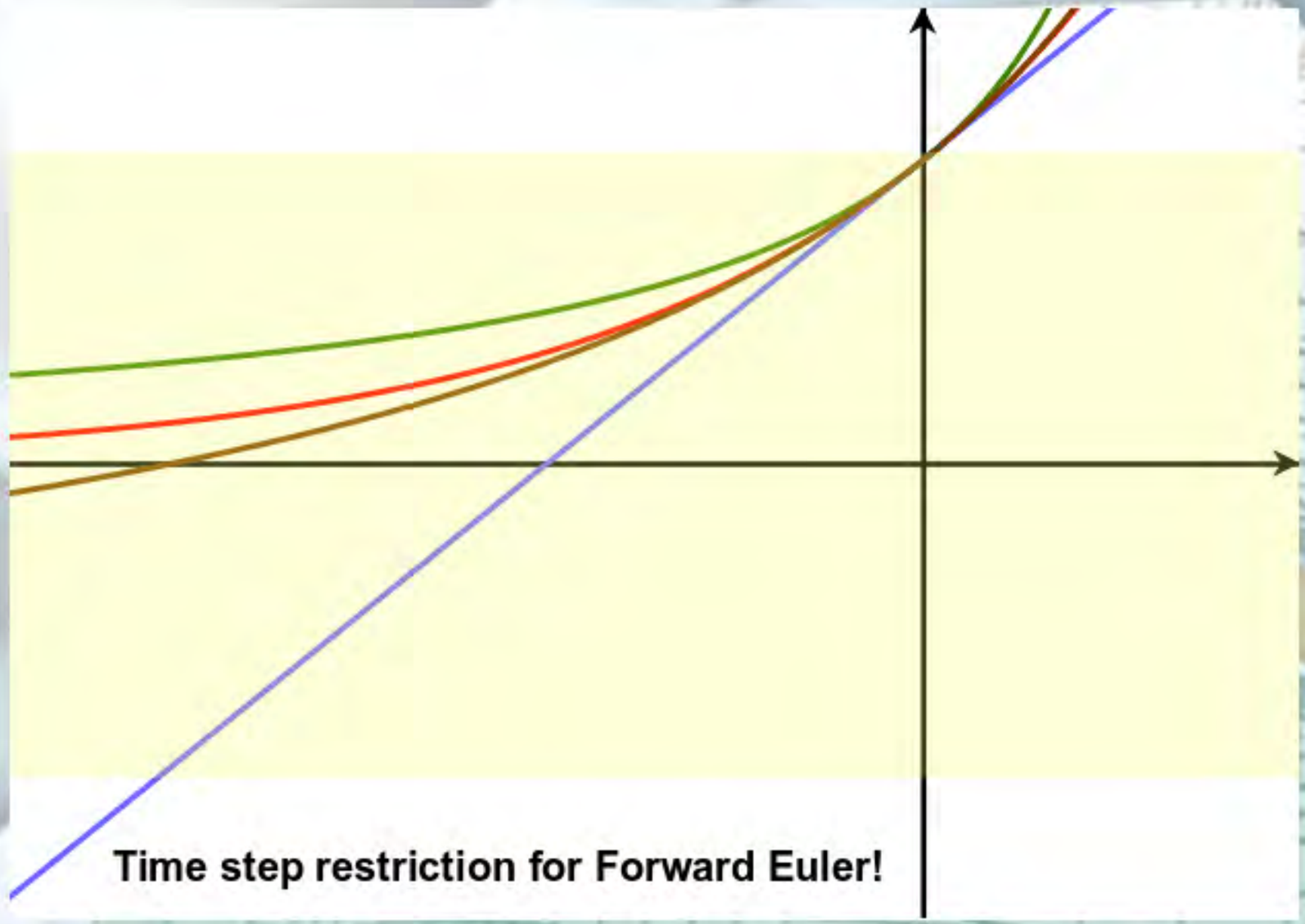
Backward Euler

$$\exp(\tau A) \approx (1 - \tau A)^{-1}$$

Crank-Nicholson

$$\exp(\tau A) \approx \left(1 + \frac{1}{2}\tau A\right)\left(1 - \frac{1}{2}\tau A\right)^{-1}$$

Approxmating $\exp(\tau A)$ by Rational Functions



Exponential

Forward Euler

$$\exp(\tau A) \approx 1 + \tau A$$

Backward Euler

$$\exp(\tau A) \approx (1 - \tau A)^{-1}$$

Crank-Nicholson

$$\exp(\tau A) \approx \left(1 + \frac{1}{2}\tau A\right)\left(1 - \frac{1}{2}\tau A\right)^{-1}$$

Fully Discrete Scheme

$$T_{n+1} = \exp(\tau A) T_n$$

Fully Discrete Scheme

$$T_{n+1} = \exp(\tau A) T_n$$

› Forward Euler (explicit Euler)

$$T_{n+1} = (1 + \tau A) T_n$$

- No matrix inversion necessary!
- But time step restriction

Fully Discrete Scheme

$$T_{n+1} = \exp(\tau A) T_n$$

- Forward Euler (explicit Euler)

$$T_{n+1} = (1 + \tau A) T_n$$

  - No matrix inversion necessary!
  - But time step restriction

- Backward Euler (implicit Euler)

$$T_{n+1} = (1 - \tau A)^{-1} T_n$$

  - Matrix inversion necessary

4

Pros and Cons: FDM vs. FEM

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement	

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement › Very efficient implementation on GPU possible	

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement › Very efficient implementation on GPU possible › Not very flexible w.r.t geometry of domain	

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement › Very efficient implementation on GPU possible › Not very flexible w.r.t geometry of domain › Only 2nd order convergence	

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement › Very efficient implementation on GPU possible › Not very flexible w.r.t geometry of domain › Only 2nd order convergence	› Very flexible w.r.t geometry of domain

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement › Very efficient implementation on GPU possible › Not very flexible w.r.t geometry of domain › Only 2nd order convergence	› Very flexible w.r.t geometry of domain › hp-refinement possible

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement › Very efficient implementation on GPU possible	› Very flexible w.r.t geometry of domain › hp-refinement possible
› Not very flexible w.r.t geometry of domain › Only 2nd order convergence	› More difficult to implement

Pros and Cons: FDM vs. FEM

Finite Difference Method (FDM)	Finite Element Method (FEM)
› Easy to implement › Very efficient implementation on GPU possible	› Very flexible w.r.t geometry of domain › hp-refinement possible
› Not very flexible w.r.t geometry of domain › Only 2nd order convergence	› More difficult to implement

For 1D regular grids and piecewise linear shape functions: FDM = FEM

5

Examples

Image-Based Computing Pipeline

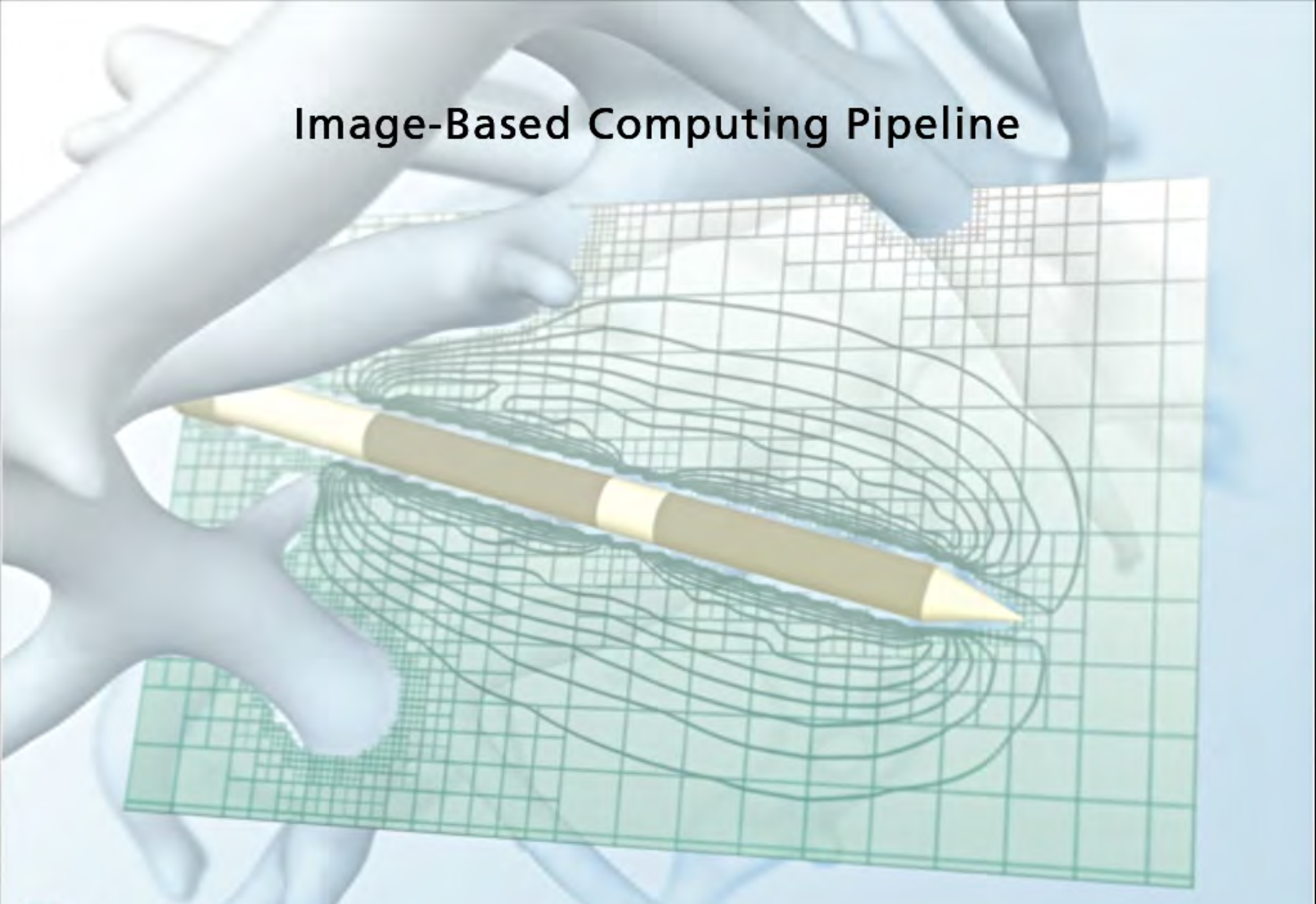


Image-Based Computing Pipeline

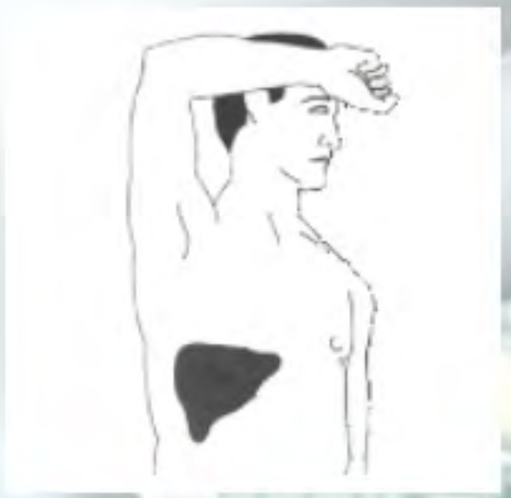


Image-Based Computing Pipeline

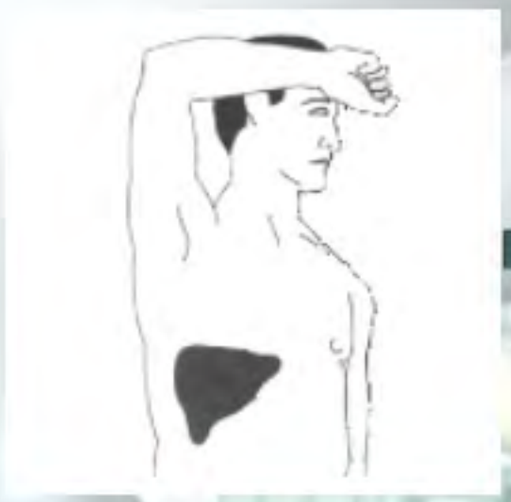
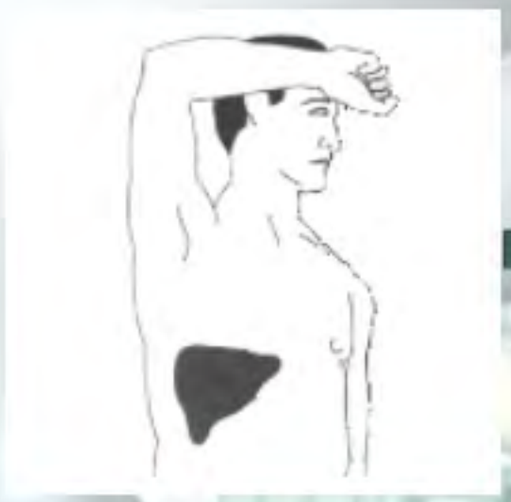


Image Acquisition

Image-Based Computing Pipeline



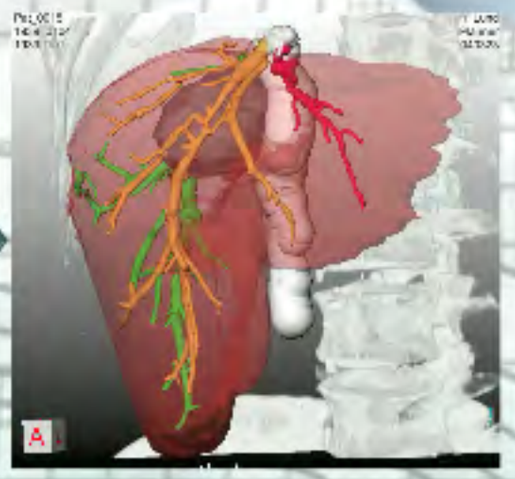
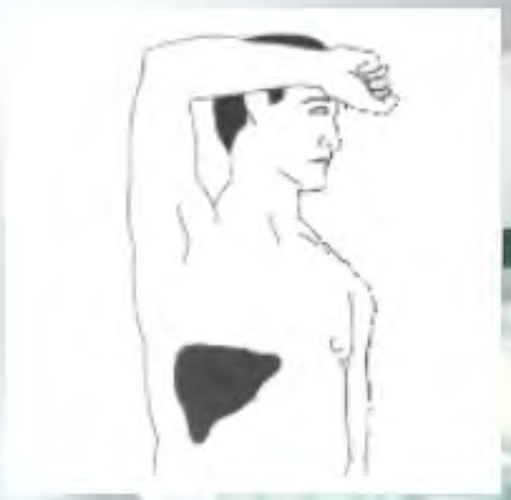
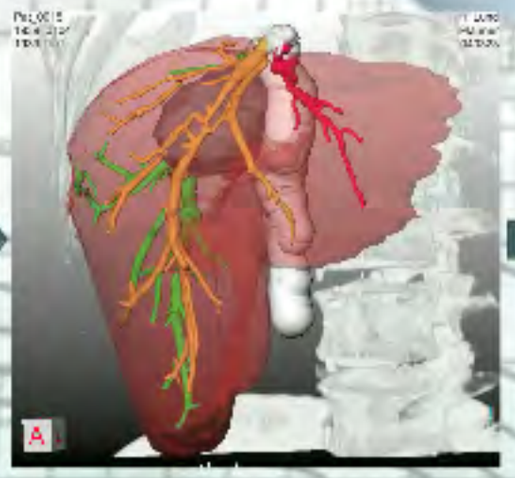


Image Acquisition

Image Processing

Image-Based Computing Pipeline





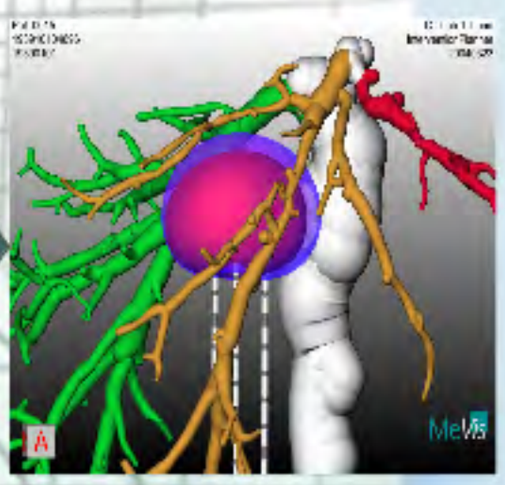
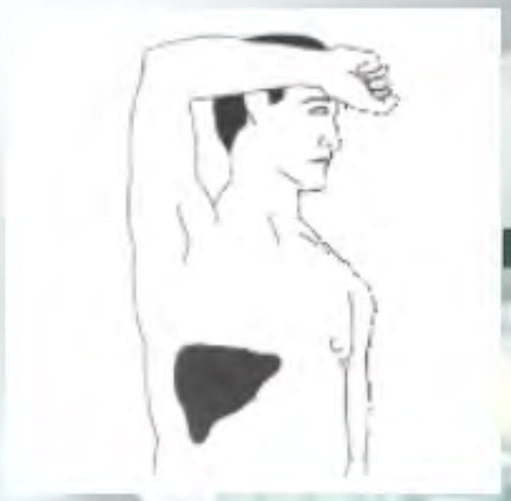


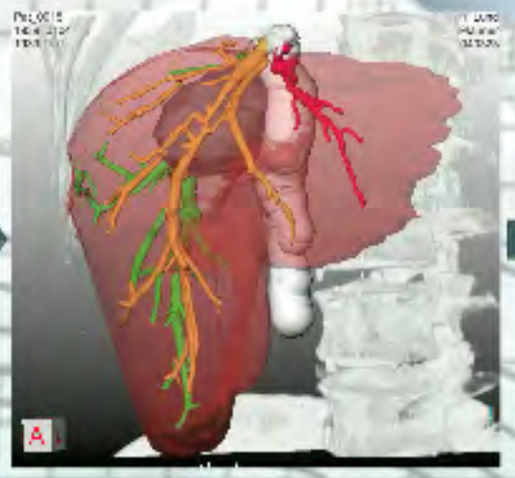
Image Acquisition

Image Processing

Simulation

Image-Based Computing Pipeline





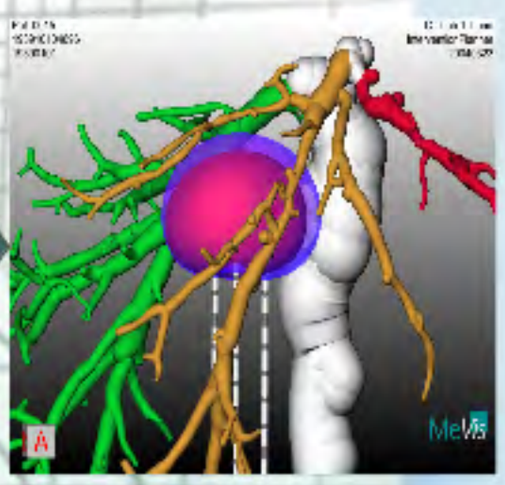
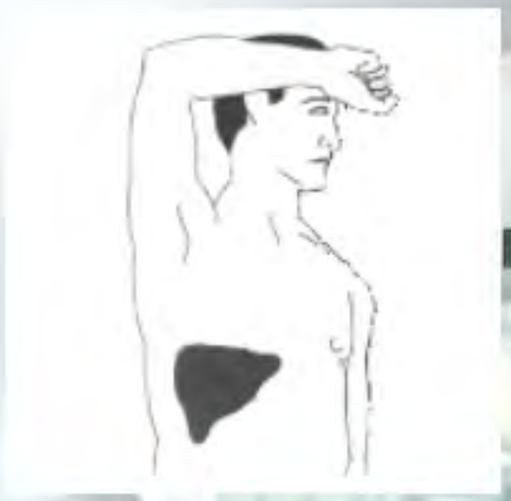


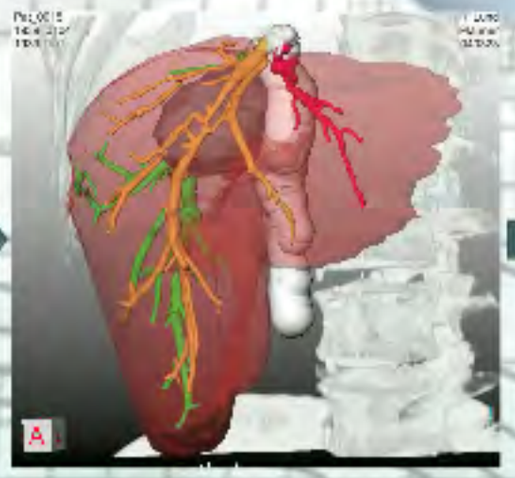
Image Acquisition

Image Processing

Simulation

Image-Based Computing Pipeline





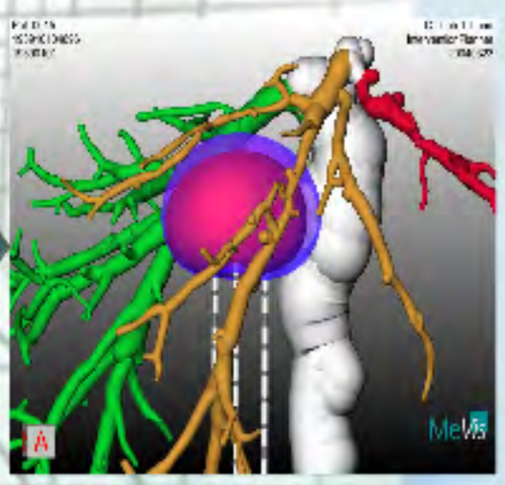


Image Acquisition

Image Processing

Simulation

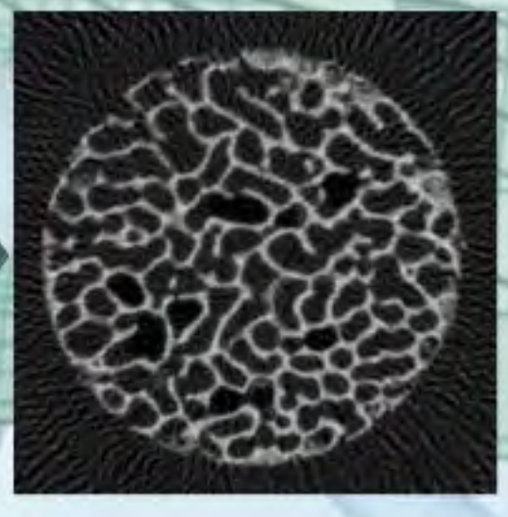
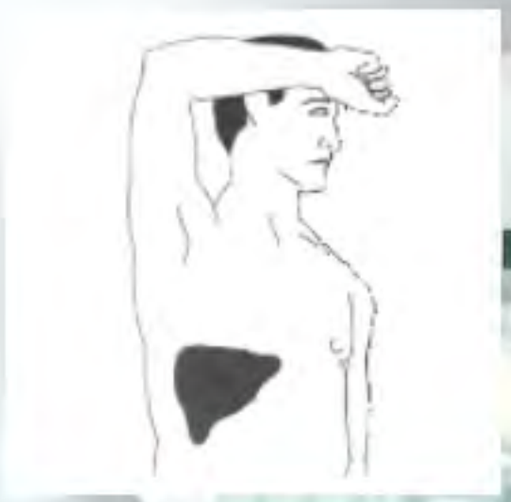
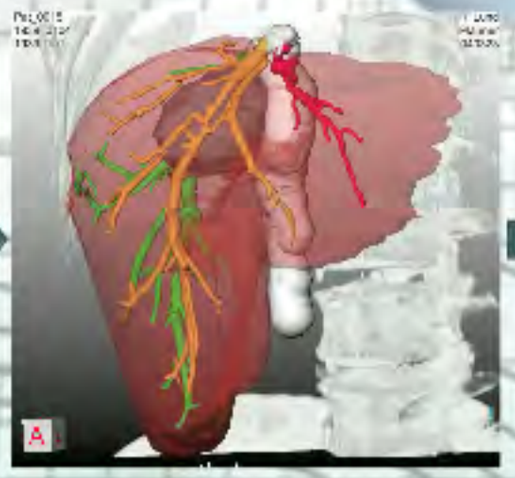


Image-Based Computing Pipeline





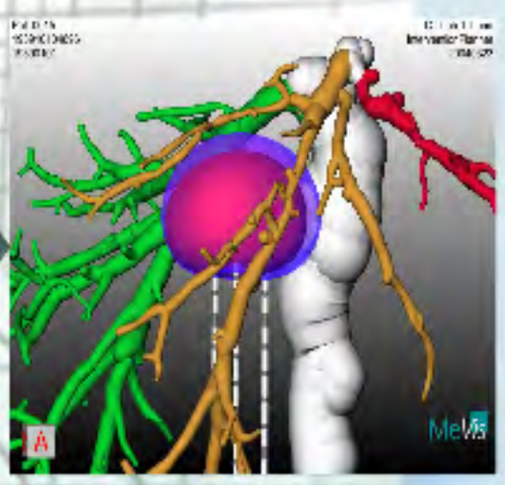


Image Acquisition

Image Processing

Simulation

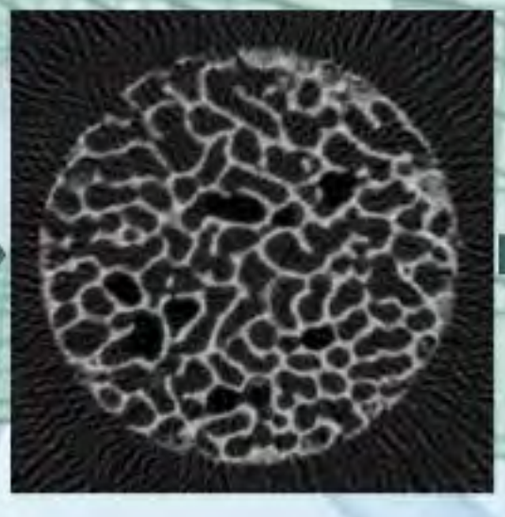


Image-Based Computing Pipeline

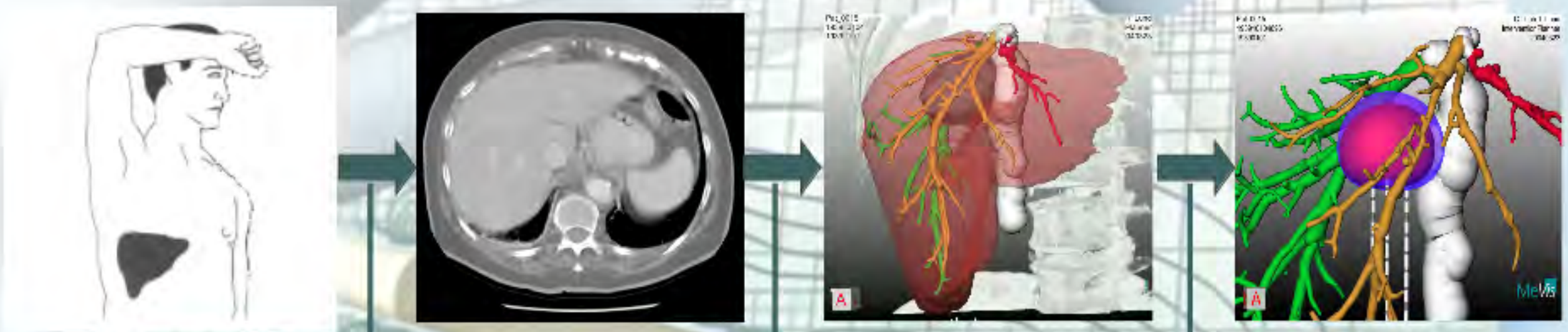


Image Acquisition

Image Processing

Simulation

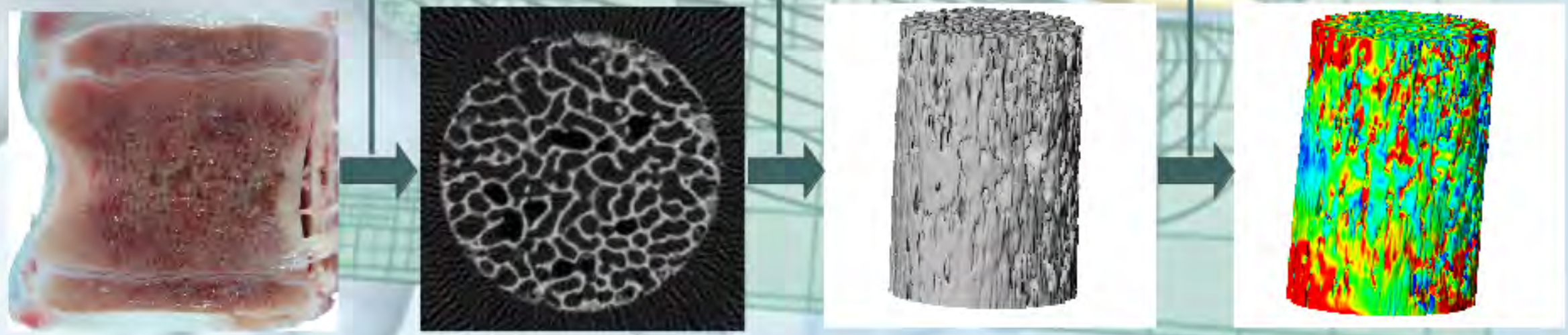


Image-Based Computing Pipeline

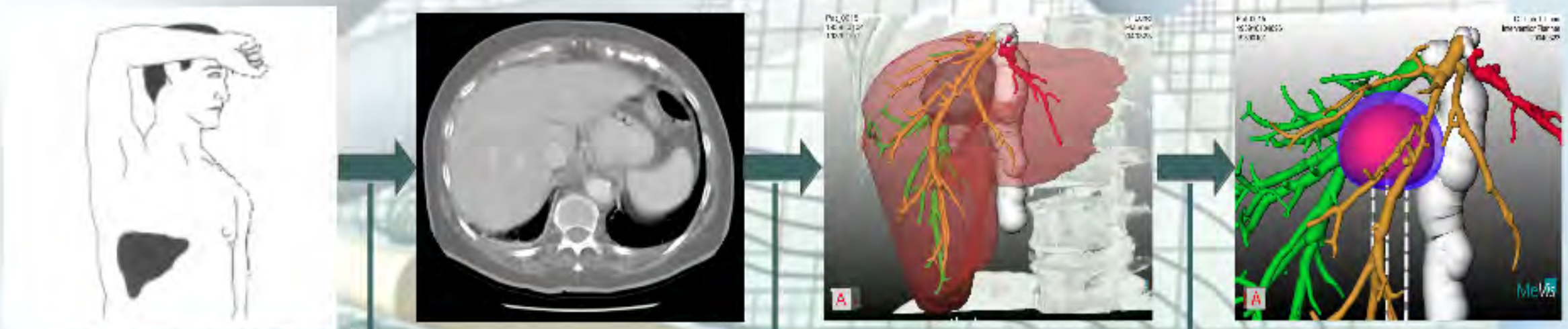
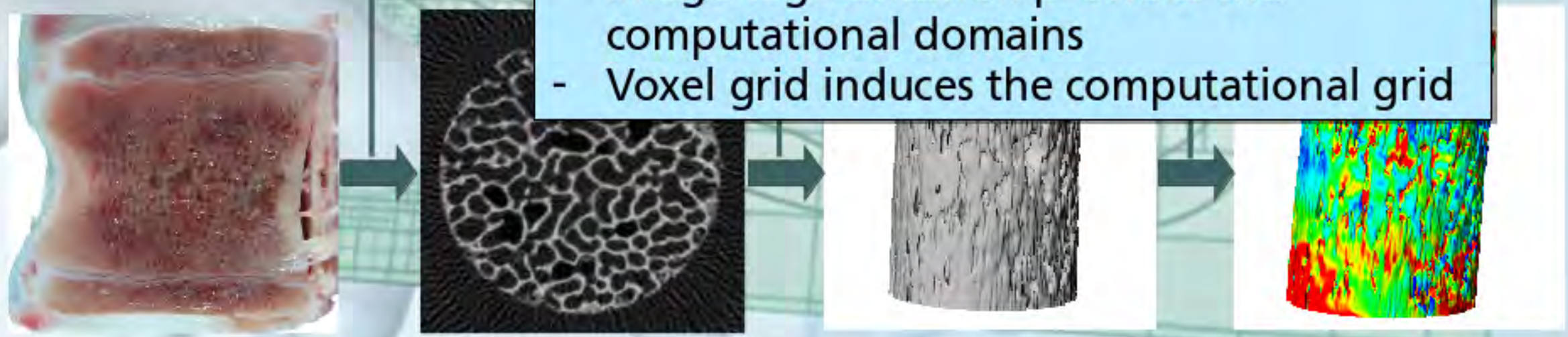


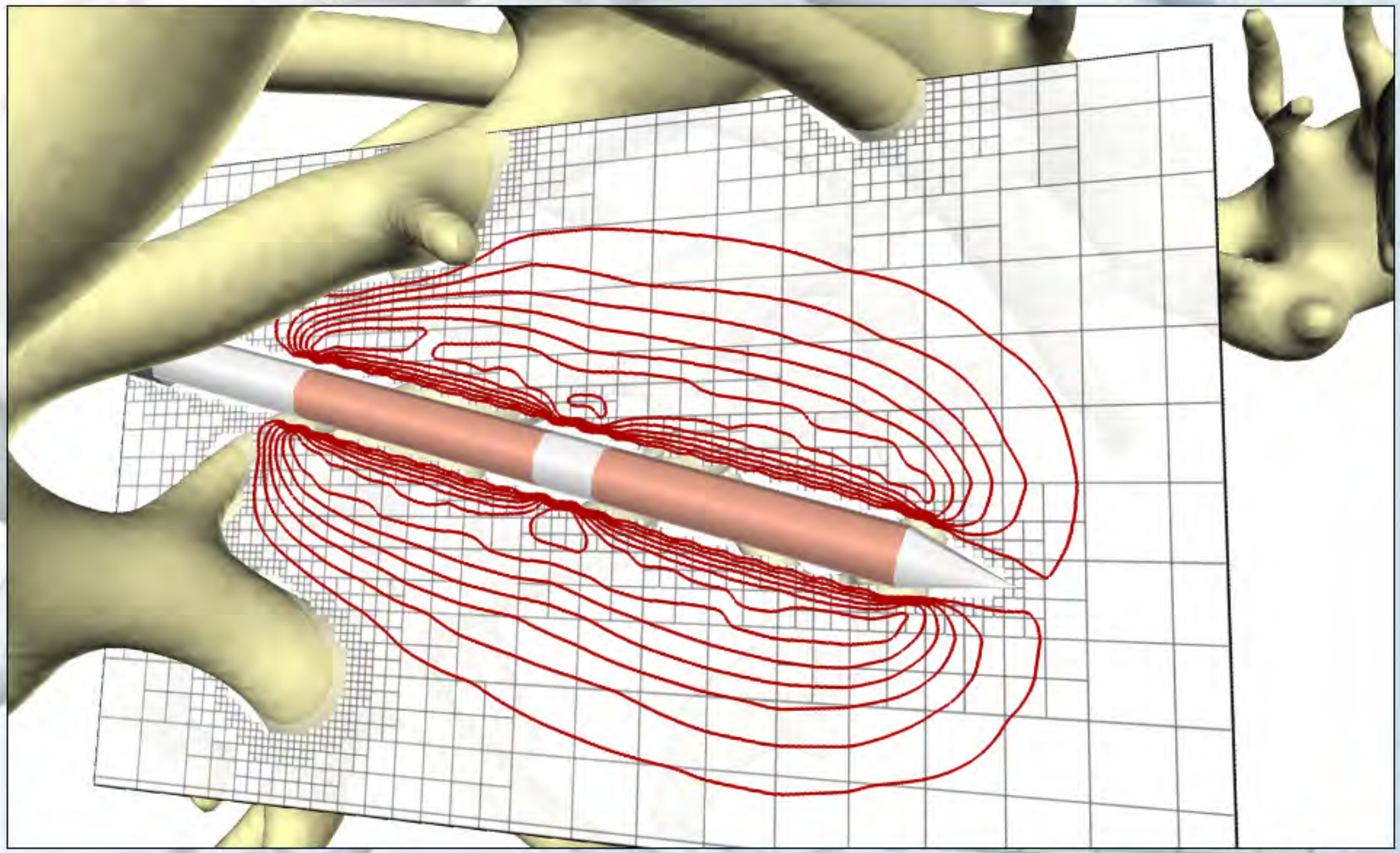
Image Acquisition | **Image Processing** | **Simulation**



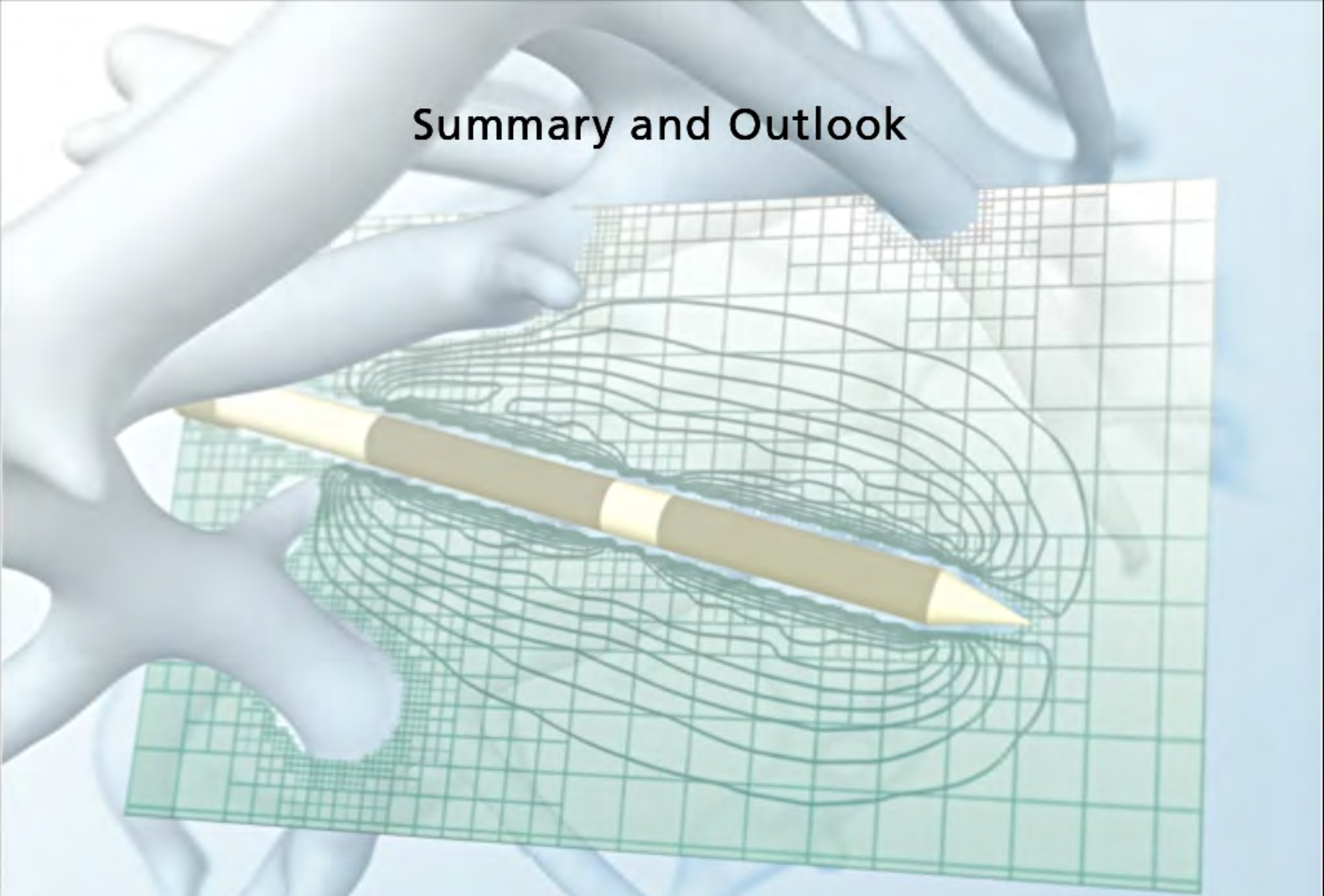
- Image segmentation provides the computational domains
- Voxel grid induces the computational grid

Simulation of RF Ablation

Simulation of RF Ablation



Summary and Outlook



Summary and Outlook

Simulation of bio-physical processes:
What happens under the hood of simulation software?

- Finite Difference Methods
- Finite Element Methods

Summary and Outlook

Simulation of bio-physical processes:
What happens under the hood of simulation software?

- Finite Difference Methods
- Finite Element Methods

- Bio-heat model problem
- Works similarly (but more technical) for elasticity

Summary and Outlook

Simulation of bio-physical processes:
What happens under the hood of simulation software?

- Finite Difference Methods
- Finite Element Methods
- Bio-heat model problem
- Works similarly (but more technical) for elasticity

Contact:

ole.schwen@mevis.fraunhofer.de