AMCS 394E: Contemp. Topics in Computational Science.
Computing with the finite element method

David I. Ketcheson and Manuel Quezada de Luna

King Abdullah University of Science and Technology

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Objectives

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Objectives

* Learn the basics of the theory behind the FE method.

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Objectives

* Learn the basics of the theory behind the FE method.
* Learn how to implement the FE method from scratch.

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Objectives

* Learn the basics of the theory behind the FE method.
* Learn how to implement the FE method from scratch.
* Learn to use a FE library to solve real applications.

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Objectives

* Learn the basics of the theory behind the FE method.
* Learn how to implement the FE method from scratch.
* Learn to use a FE library to solve real applications.

Prerequisites

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Objectives

* Learn the basics of the theory behind the FE method.
* Learn how to implement the FE method from scratch.
* Learn to use a FE library to solve real applications.

Prerequisites

* Theoretical side: have exposure to PDEs and numerical methods.

Overview of the course

Our goal is to learn the basics of the finite element (FE) method and its practical implementation.

Objectives

* Learn the basics of the theory behind the FE method.
* Learn how to implement the FE method from scratch.
* Learn to use a FE library to solve real applications.

Prerequisites

* Theoretical side: have exposure to PDEs and numerical methods.
* Computational side: have exposure to Python or Matlab and C++.

Overview of the course: method of evaluation

Homeworks (40%)

* Programming assignments with a short report.

Overview of the course: method of evaluation

Homeworks (40%)

* Programming assignments with a short report.

Final project (60%)

Overview of the course: method of evaluation

Homeworks (40%)

* Programming assignments with a short report.

Final project (60%)

* Objective: solve a practical application.

Overview of the course: method of evaluation

Homeworks (40%)

* Programming assignments with a short report.

Final project (60%)

* Objective: solve a practical application.
* Examples:
  - Solve the Euler equations in multiple dimensions.
  - Test different h-adaptivity criteria during the sol. of the shallow water equations.
  - Solve a problem of a floating cube via ALE.
  - Implement and test a numerical scheme from some publication.
  - Solve a equation or test a method from your own research.

Overview of the course: method of evaluation

Homeworks (40%)

* Programming assignments with a short report.

Final project (60%)

* Objective: solve a practical application.
* Examples:
  - Solve the Euler equations in multiple dimensions.
  - Test different h-adaptivity criteria during the sol. of the shallow water equations.
  - Solve a problem of a floating cube via ALE.
  - Implement and test a numerical scheme from some publication.
  - Solve a equation or test a method from your own research.
* Format: report (around 10 pages) with a description of the problem, equations, numerical methods, details of the implementation and numerical results.

Review of PDEs

A PDE is an equation involving multiple independent variables and their derivatives; e.g., consider

$$u = u(x, y, t), \qquad F(u, u_t, u_x, u_{xx}, \ldots, u_y, u_{yy}, \ldots, x, y, t) = 0.$$

Review of PDEs

A PDE is an equation involving multiple independent variables and their derivatives; e.g., consider

$$u = u(x, y, t), \qquad F(u, u_t, u_x, u_{xx}, \ldots, u_y, u_{yy}, \ldots, x, y, t) = 0.$$

"Since Newton, mankind has come to realize that the laws of physics are always expressed in the language of differential equations." Steven Strogatz

Review of PDEs

A PDE is an equation involving multiple independent variables and their derivatives; e.g., consider

$$u = u(x, y, t), \qquad F(u, u_t, u_x, u_{xx}, \ldots, u_y, u_{yy}, \ldots, x, y, t) = 0.$$

"Since Newton, mankind has come to realize that the laws of physics are always expressed in the language of differential equations." Steven Strogatz

(0:45-2:04) `https://www.youtube.com/watch?v=p_di4Zn4wz4`

(0:11-1:36, 3:32-5:58) `https://www.youtube.com/watch?v=ly4S0oi3Yz8`

Review of PDEs

A PDE is an equation involving multiple independent variables and their derivatives; e.g., consider

$$u = u(x, y, t), \qquad F(u, u_t, u_x, u_{xx}, \ldots, u_y, u_{yy}, \ldots, x, y, t) = 0.$$

PDEs model the bahavior of functions of multiple variables. The applications are vast:

* Weather forecast
* Electromagnetism
* Combustion
* Distribution of stress in a structure
* Finance
* Nuclear physics
* Blood flow
* Elasticity
* Fluid flows
* Multiphase flows
* Floating objects (ships, etc)
* Propagation of a tsunami
* Quantum mechanics
* Spacetime and matter relation
* Heat distribution and evolution
* Etcetera

Review of PDEs

Some important concepts:

Review of PDEs

Some important concepts:

* Order of a PDE.

Review of PDEs

Some important concepts:

* Order of a PDE.
* Linear vs nonlinear PDEs.

Review of PDEs

Some important concepts:

* Order of a PDE.
* Linear vs nonlinear PDEs.
* Constant or variable coefficient PDEs.

Review of PDEs

Some important concepts:

* Order of a PDE.
* Linear vs nonlinear PDEs.
* Constant or variable coefficient PDEs.

Types of PDEs based on:

Review of PDEs

Some important concepts:

* Order of a PDE.
* Linear vs nonlinear PDEs.
* Constant or variable coefficient PDEs.

Types of PDEs based on:

* Order: first-order, second-order, etc.

Review of PDEs

Some important concepts:

* Order of a PDE.
* Linear vs nonlinear PDEs.
* Constant or variable coefficient PDEs.

Types of PDEs based on:

* Order: first-order, second-order, etc.
* Depending on the linearity: linear, quasilinear or nonlinear.

Review of PDEs

Some important concepts:

* Order of a PDE.
* Linear vs nonlinear PDEs.
* Constant or variable coefficient PDEs.

Types of PDEs based on:

* Order: first-order, second-order, etc.
* Depending on the linearity: linear, quasilinear or nonlinear.
* Depending on the force term: homogeneous or non-homogeneous.

Review of PDEs

Classification of second order PDEs:

Review of PDEs

Classification of second order PDEs:

* Hyperbolic

Review of PDEs

Classification of second order PDEs:

* Hyperbolic
* Elliptic

Review of PDEs

Classification of second order PDEs:

* Hyperbolic
* Elliptic
* Parabolic

Numerical methods for PDEs

Most common numerical methods:

Numerical methods for PDEs

Most common numerical methods:

* Finite differences

Numerical methods for PDEs

Most common numerical methods:

* Finite differences
* Finite volumes

Numerical methods for PDEs

Most common numerical methods:

* Finite differences
* Finite volumes
* Finite elements

Numerical methods for PDEs

Most common numerical methods:

* Finite differences
* Finite volumes
* Finite elements

Method of lines:

Popular strategy to solve time dependent PDEs. The spatial derivatives are discretized directly which leads to system of (non)linear ordinary differential equations.

The system of ODEs is solved at different time intervals.

Finite difference method (FDM)

The FDM solves for point values at given nodes by approximating the derivatives through finite differences.

Finite difference method (FDM)

The FDM solves for point values at given nodes by approximating the derivatives through finite differences.

Some characteristics are:

Finite difference method (FDM)

The FDM solves for point values at given nodes by approximating the derivatives through finite differences.

Some characteristics are:

* derivation and error analysis are done via Taylor series

Finite difference method (FDM)

The FDM solves for point values at given nodes by approximating the derivatives through finite differences.

Some characteristics are:

* derivation and error analysis are done via Taylor series
* easy to implement for structured grids.

Finite difference method (FDM)

The FDM solves for point values at given nodes by approximating the derivatives through finite differences.

Some characteristics are:

* derivation and error analysis are done via Taylor series
* easy to implement for structured grids.
* fast and efficient

Finite difference method (FDM)

The FDM solves for point values at given nodes by approximating the derivatives through finite differences.

Some characteristics are:

* derivation and error analysis are done via Taylor series
* easy to implement for structured grids.
* fast and efficient
* cumbersome for unstructured grids

Finite difference method (FDM)

Approximation of first- and second-order derivatives.

Finite difference method (FDM)

Approximation of first- and second-order derivatives.

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Finite difference method (FDM)

Approximation of first- and second-order derivatives.

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

Finite difference method (FDM)

Approximation of first- and second-order derivatives.

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Consider the spatial discretization for a given node i.

Finite difference method (FDM)

Approximation of first- and second-order derivatives.

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Consider the spatial discretization for a given node i.
* Consider a matrix form of the semi-discretization.

Finite difference method (FDM)

Approximation of first- and second-order derivatives.

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Consider the spatial discretization for a given node i.
* Consider a matrix form of the semi-discretization.
* Discretize in time using e.g. BE, FE, RK methods, etc.

Finite volume method (FVM)

The FVM solves for cell averages.
The derivatives are commonly approximated via numerical fluxes.

Finite volume method (FVM)

The FVM solves for cell averages.
The derivatives are commonly approximated via numerical fluxes.

Some characteristics are:

Finite volume method (FVM)

The FVM solves for cell averages.
The derivatives are commonly approximated via numerical fluxes.

Some characteristics are:

* easy to implement for structured grids

Finite volume method (FVM)

The FVM solves for cell averages.
The derivatives are commonly approximated via numerical fluxes.

Some characteristics are:

* easy to implement for structured grids
* still relatively fast and efficient

Finite volume method (FVM)

The FVM solves for cell averages.
The derivatives are commonly approximated via numerical fluxes.

Some characteristics are:

* easy to implement for structured grids
* still relatively fast and efficient
* high-order approximations are cumbersome for unstructured grids

Finite volume method (FVM)

The FVM solves for cell averages.
The derivatives are commonly approximated via numerical fluxes.

Some characteristics are:

* easy to implement for structured grids
* still relatively fast and efficient
* high-order approximations are cumbersome for unstructured grids
* widely used for hyperbolic equations

Finite volume method (FVM)

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Finite volume method (FVM)

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

Finite volume method (FVM)

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Integrate the PDE over cell K_i.

Finite volume method (FVM)

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Integrate the PDE over cell K_i.
* Integrate by parts, use the divergence theorem, etc.

Finite volume method (FVM)

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Integrate the PDE over cell K_i.
* Integrate by parts, use the divergence theorem, etc.
* Replace the fluxes through the boundary by numerical fluxes.

Finite volume method (FVM)

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Integrate the PDE over cell K_i.
* Integrate by parts, use the divergence theorem, etc.
* Replace the fluxes through the boundary by numerical fluxes.
* Consider a matrix form of the semi-discretization.

Finite volume method (FVM)

Example. Consider the following advection reaction diffusion equation:

$$u_t + u_x - \mu u_{xx} = r(x), \qquad \mu > 0$$

Basic steps:

* Integrate the PDE over cell K_i.
* Integrate by parts, use the divergence theorem, etc.
* Replace the fluxes through the boundary by numerical fluxes.
* Consider a matrix form of the semi-discretization.
* Discretize in time using e.g. BE, FE, RK methods, etc.

When to use each type of method?

Finite differences:

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient

When to use each type of method?

Finite differences:

- Easiest to implement and the most efficient
- Difficult to generalize for unstructured grids

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM
* Difficult to generalize to unstructured grids

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM
* Difficult to generalize to unstructured grids
* In some codes it is difficult to consider diffusive terms

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM
* Difficult to generalize to unstructured grids
* In some codes it is difficult to consider diffusive terms

Finite elements:

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM
* Difficult to generalize to unstructured grids
* In some codes it is difficult to consider diffusive terms

Finite elements:

* FEM is the slowest and harder to understand and implement.

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM
* Difficult to generalize to unstructured grids
* In some codes it is difficult to consider diffusive terms

Finite elements:

* FEM is the slowest and harder to understand and implement.
* Easy to work with unstructured grids.

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM
* Difficult to generalize to unstructured grids
* In some codes it is difficult to consider diffusive terms

Finite elements:

* FEM is the slowest and harder to understand and implement.
* Easy to work with unstructured grids.
* Easy to consider complex geometries.

When to use each type of method?

Finite differences:

* Easiest to implement and the most efficient
* Difficult to generalize for unstructured grids
* Easy to consider different type of operators
* Good to test different models with simple geometries

Finite volumes:

* FVM is widely used and developed for hyperbolic equations
* Easy to implement and more efficient than FEM
* Difficult to generalize to unstructured grids
* In some codes it is difficult to consider diffusive terms

Finite elements:

* FEM is the slowest and harder to understand and implement.
* Easy to work with unstructured grids.
* Easy to consider complex geometries.
* Easy to consider different type of operators.

Scientific libraries for finite elements

Finite element libraries versus personalized codes.

See `https://www.math.colostate.edu/~bangerth/videos.676.1.html`

Scientific libraries for finite elements

Finite element libraries versus personalized codes.

See `https://www.math.colostate.edu/~bangerth/videos.676.1.html`

Some things to consider:

Scientific libraries for finite elements

Finite element libraries versus personalized codes.

See `https://www.math.colostate.edu/~bangerth/videos.676.1.html`

Some things to consider:

* A personalized solver might be considerably faster.

Scientific libraries for finite elements

Finite element libraries versus personalized codes.

See `https://www.math.colostate.edu/~bangerth/videos.676.1.html`

Some things to consider:

* A personalized solver might be considerably faster.
* Coding a fast solver from scratch requires considerable experience and knowledge.

Scientific libraries for finite elements

Finite element libraries versus personalized codes.

See `https://www.math.colostate.edu/~bangerth/videos.676.1.html`

Some things to consider:

* A personalized solver might be considerably faster.
* Coding a fast solver from scratch requires considerable experience and knowledge.
* Some FE libraries have been largely tested and many bugs have been found and fixed.

Scientific libraries for finite elements

Finite element libraries versus personalized codes.

See https://www.math.colostate.edu/~bangerth/videos.676.1.html

Some things to consider:

* A personalized solver might be considerably faster.
* Coding a fast solver from scratch requires considerable experience and knowledge.
* Some FE libraries have been largely tested and many bugs have been found and fixed.
* Libraries have many (difficult to implement) tools like:
  - multigrid methods
  - parallelized solvers
  - h-adaptivity
  - moving meshes
  - etc

Scientific libraries for finite elements

Finite element libraries versus personalized codes.

See https://www.math.colostate.edu/~bangerth/videos.676.1.html

Some things to consider:

* A personalized solver might be considerably faster.
* Coding a fast solver from scratch requires considerable experience and knowledge.
* Some FE libraries have been largely tested and many bugs have been found and fixed.
* Libraries have many (difficult to implement) tools like:
    - multigrid methods
    - parallelized solvers
    - h-adaptivity
    - moving meshes
    - etc
* In general it might take years to code all the tools available in a good FE library.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.
* Maps from local to global entries in the algebraic systems.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.
* Maps from local to global entries in the algebraic systems.
* Tools to perform quadratures, access data, etc.
* etc.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.
* Maps from local to global entries in the algebraic systems.
* Tools to perform quadratures, access data, etc.
* etc.

Main differences:

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.
* Maps from local to global entries in the algebraic systems.
* Tools to perform quadratures, access data, etc.
* etc.

Main differences:

* Extra tools to mesh, solve the linear systems, visualization, etc.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.
* Maps from local to global entries in the algebraic systems.
* Tools to perform quadratures, access data, etc.
* etc.

Main differences:

* Extra tools to mesh, solve the linear systems, visualization, etc.
* Pre-define functions to assemble common operators.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.
* Maps from local to global entries in the algebraic systems.
* Tools to perform quadratures, access data, etc.
* etc.

Main differences:

* Extra tools to mesh, solve the linear systems, visualization, etc.
* Pre-define functions to assemble common operators.
* How exposed the indices, data structures, conectivity matrix, etc are.

Scientific libraries for finite elements

Examples of finite element libraries:

deal.II, MFEM, FEniCS Project, FreeFEM, Proteus, etc.

Common things between libraries:

* Finite element loop: elements, quad points, shape functions, etc.
* Finite element transformations.
* Maps from local to global entries in the algebraic systems.
* Tools to perform quadratures, access data, etc.
* etc.

Main differences:

* Extra tools to mesh, solve the linear systems, visualization, etc.
* Pre-define functions to assemble common operators.
* How exposed the indices, data structures, conectivity matrix, etc are.
* Tools to handle matrices in different forms (CSR, standard row-column indices, etc).
* etc.

Scientific libraries for finite elements

What to look to choose a finite element library:

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies
* parallelization

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies
* parallelization
* standard versus non-standard implementations
* etc

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies
* parallelization
* standard versus non-standard implementations
* etc

Other libraries to work with finite elements:

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies
* parallelization
* standard versus non-standard implementations
* etc

Other libraries to work with finite elements:

* meshing

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies
* parallelization
* standard versus non-standard implementations
* etc

Other libraries to work with finite elements:

* meshing
* partition of the domain

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies
* parallelization
* standard versus non-standard implementations
* etc

Other libraries to work with finite elements:

* meshing
* partition of the domain
* linear algebra

Scientific libraries for finite elements

What to look to choose a finite element library:

* programming language
* software dependencies
* parallelization
* standard versus non-standard implementations
* etc

Other libraries to work with finite elements:

* meshing
* partition of the domain
* linear algebra
* visualization
* etc