Object-oriented scientific computing

Pras Pathmanathan

Summer 2012

Classes

```
class Academic inherits from Human:
    Data:
        mNumPapers
    Methods:
        PublishPaper()          (increments mNumPapers by one)
        GetNumPapers()
```

Key concepts in object-oriented programming

1. Encapsulation
2. Security (private member variables)
3. Interface
4. Inheritence
5. Abstract methods

Second-order PDEs

PDEs

Second-order PDEs commonly arise in physical models. There are three archetypal second-order PDEs

1. **Elliptic PDEs**, for example, Poisson's equation

$$\nabla^2 u + f = 0$$

2. **Parabolic PDEs**, for example, the heat equation

$$u_t = \nabla^2 u + f$$

3. **Hyperbolic PDEs**, for example, the wave equation

$$u_{tt} = \nabla^2 u$$

We will only consider **elliptic and parabolic** PDEs..

PDEs

Second-order PDEs commonly arise in physical models. There are three archetypal second-order PDEs

1. **Elliptic PDEs**, for example, Poisson's equation

$$\nabla^2 u + f = 0$$

2. **Parabolic PDEs**, for example, the heat equation

$$u_t = \nabla^2 u + f$$

3. **Hyperbolic PDEs**, for example, the wave equation

$$u_{tt} = \nabla^2 u$$

We will only consider **elliptic and parabolic** PDEs..

PDEs

Second-order PDEs commonly arise in physical models. There are three archetypal second-order PDEs

1. **Elliptic PDEs**, for example, Poisson's equation

$$\nabla^2 u + f = 0$$

2. **Parabolic PDEs**, for example, the heat equation

$$u_t = \nabla^2 u + f$$

3. **Hyperbolic PDEs**, for example, the wave equation

$$u_{tt} = \nabla^2 u$$

We will only consider **elliptic and parabolic** PDEs..

PDEs

Second-order PDEs commonly arise in physical models. There are three archetypal second-order PDEs

1. **Elliptic PDEs**, for example, Poisson's equation

$$\nabla^2 u + f = 0$$

2. **Parabolic PDEs**, for example, the heat equation

$$u_t = \nabla^2 u + f$$

3. **Hyperbolic PDEs**, for example, the wave equation

$$u_{tt} = \nabla^2 u$$

We will only consider **elliptic and parabolic** PDEs..

PDEs

Second-order PDEs commonly arise in physical models. There are three archetypal second-order PDEs

1. **Elliptic PDEs**, for example, Poisson's equation

$$\nabla^2 u + f = 0$$

2. **Parabolic PDEs**, for example, the heat equation

$$u_t = \nabla^2 u + f$$

3. **Hyperbolic PDEs**, for example, the wave equation

$$u_{tt} = \nabla^2 u$$

We will only consider **elliptic and parabolic** PDEs..

Defining PDEs in an object oriented manner

First, an abstract class defining a general linear elliptic PDE

$$\nabla \cdot D \boldsymbol{\nabla} u = f,$$

where D is a matrix-valued function of position (the diffusion tensor):

`AbstractLinearEllipticPde`:
Abs. method: `GetDiffusionTensor(x)`
Abs. method: `GetForceTerm(x)`

`MyEllipticPde`: *inherits from* `AbstractLinearEllipticPde`
Implemented method: `GetDiffusionTensor(x)`
Implemented method: `GetForceTerm(x)`

For example $\nabla^2 u = 0$

`LaplacesEquation`: *inherits from* `AbstractLinearEllipticPde`
Implemented method: `GetDiffusionTensor(x)`
▷ *return identity matrix*
Implemented method: `GetForceTerm(x)`
▷ *return zero*

Defining PDEs in an object oriented manner

First, an abstract class defining a general linear elliptic PDE

$$\nabla \cdot D \boldsymbol{\nabla} u = f,$$

where D is a matrix-valued function of position (the diffusion tensor):

`AbstractLinearEllipticPde`:
- *Abs. method:* `GetDiffusionTensor(x)`
- *Abs. method:* `GetForceTerm(x)`

`MyEllipticPde`: *inherits from* `AbstractLinearEllipticPde`
- *Implemented method:* `GetDiffusionTensor(x)`
- *Implemented method:* `GetForceTerm(x)`

For example $\nabla^2 u = 0$

`LaplacesEquation`: *inherits from* `AbstractLinearEllipticPde`
- *Implemented method:* `GetDiffusionTensor(x)`
  - ▷ *return identity matrix*
- *Implemented method:* `GetForceTerm(x)`
  - ▷ *return zero*

Defining PDEs in an object oriented manner

First, an abstract class defining a general linear elliptic PDE

$$\nabla \cdot D\boldsymbol{\nabla} u = f,$$

where D is a matrix-valued function of position (the diffusion tensor):

`AbstractLinearEllipticPde`:
    Abs. method: `GetDiffusionTensor(x)`
    Abs. method: `GetForceTerm(x)`

`MyEllipticPde`: *inherits from* `AbstractLinearEllipticPde`
    Implemented method: `GetDiffusionTensor(x)`
    Implemented method: `GetForceTerm(x)`

For example $\nabla^2 u = 0$

`LaplacesEquation`: *inherits from* `AbstractLinearEllipticPde`
    Implemented method: `GetDiffusionTensor(x)`
        ▷ *return identity matrix*
    Implemented method: `GetForceTerm(x)`
        ▷ *return zero*

Defining PDEs in an object oriented manner

Next, an abstract class defining a general linear parabolic PDE

$$\alpha u_t = \nabla \cdot D \boldsymbol{\nabla} u + f$$

where α, D and f are functions of space and time.

`AbstractLinearParabolicPde`:

Abs. method: `GetDuDtCoefficientTerm(t,x)`
Abs. method: `GetDiffusionTensor(t,x)`
Abs. method: `GetForceTerm(t,x)`

For example $u_t = \nabla^2 u$

`HeatEquation`: *inherits from* `AbstractLinearParabolicPde`

Implemented method: `GetDuDtCoefficientTerm(t,x)`
▷ *return 1*
Implemented method: `GetDiffusionTensor(x)`
▷ *return identity matrix*
Implemented method: `GetForceTerm(x)`
▷ *return zero*

Defining PDEs in an object oriented manner

Next, an abstract class defining a general linear parabolic PDE

$$\alpha u_t = \nabla \cdot D \boldsymbol{\nabla} u + f$$

where α, D and f are functions of space and time.

`AbstractLinearParabolicPde`:
- *Abs. method:* `GetDuDtCoefficientTerm(t,x)`
- *Abs. method:* `GetDiffusionTensor(t,x)`
- *Abs. method:* `GetForceTerm(t,x)`

For example $u_t = \nabla^2 u$

`HeatEquation`: *inherits from* `AbstractLinearParabolicPde`
- *Implemented method:* `GetDuDtCoefficientTerm(t,x)`
  - ▷ *return 1*
- *Implemented method:* `GetDiffusionTensor(x)`
  - ▷ *return identity matrix*
- *Implemented method:* `GetForceTerm(x)`
  - ▷ *return zero*

Defining PDEs in an object oriented manner

Doing this allows the possibility of writing solvers of *generic* PDEs, eg

`EllipticPdeSolver`:
Method: `Solve(AbstractLinearEllipticPde)`

`ParabolicPdeSolver`:
Method: `Solve(AbstractLinearParabolicPde,t0,t1,initCond)`

The finite difference method

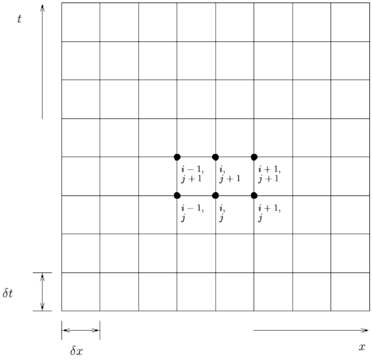
Heat equation on a regular geometry

Consider the PDE

$$u_t = u_{xx}$$

on $[0, 1]$, with boundary conditions $u = 0$ and initial condition u_0.

We discretise space and time with a spacestep δx and a timestep δt:



Heat equation on a regular geometry

Let u_i^n represent the numerical solution at (x_i, t_n). We can use the approximations:

$$
\begin{aligned}
\frac{\partial u}{\partial t}(x_i, t_n) &\approx \frac{u_i^{n+1} - u_i^n}{\Delta t} \\
\frac{\partial^2 u}{\partial x^2}(x_i, t_n) &\approx \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}
\end{aligned}
$$

which gives the numerical scheme

$$
\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}
$$

or, alternatively stated

$$
u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x^2}\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right)
$$

Given the solution $(u_0^n, u_1^n, \ldots, u_M^n)$ at time $t = t_n$, we can directly compute the solution at the next timestep, i.e. the scheme is **explicit**

Heat equation on a regular geometry

Let u_i^n represent the numerical solution at (x_i, t_n). We can use the approximations:

$$\frac{\partial u}{\partial t}(x_i, t_n) \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}$$
$$\frac{\partial^2 u}{\partial x^2}(x_i, t_n) \approx \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}$$

which gives the numerical scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}$$

or, alternatively stated

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x^2}\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right)$$

Given the solution $(u_0^n, u_1^n, \ldots, u_M^n)$ at time $t = t_n$, we can directly compute the solution at the next timestep, i.e. the scheme is **explicit**

Heat equation on a regular geometry

Let u_i^n represent the numerical solution at (x_i, t_n). We can use the approximations:

$$\frac{\partial u}{\partial t}(x_i, t_n) \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}$$
$$\frac{\partial^2 u}{\partial x^2}(x_i, t_n) \approx \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}$$

which gives the numerical scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2}$$

or, alternatively stated

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x^2}\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right)$$

Given the solution $(u_0^n, u_1^n, \ldots, u_M^n)$ at time $t = t_n$, we can directly compute the solution at the next timestep, i.e. the scheme is **explicit**

Heat equation on a regular geometry

The scheme is:

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x^2}\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right)$$

with $u_0 = 0$ and $u_M = 0$

Let $\mathbf{u}^n = (u_0^n, u_1^n, \ldots, u_M^n)$. (Excluding first and last row) the above can be re-written as

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{\Delta t}{\Delta x^2} A\mathbf{u}^n$$

where A is the matrix

$$A = \begin{pmatrix} -2 & 1 & 0 & 0 & \ldots & 0 \\ 1 & -2 & 1 & 0 & \ldots & 0 \\ 0 & 1 & -2 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & -1 & 2 & -1 \\ 0 & \ldots & 0 & 0 & -1 & 2 \end{pmatrix}$$

Heat equation on a regular geometry

Suppose $\mathbf{u}^0$ is given (the initial condition).

Forward Euler (explicit)

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{\Delta t}{\Delta x^2} A \mathbf{u}^n$$

Backward Euler (implicit)

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{\Delta t}{\Delta x^2} A \mathbf{u}^{n+1}$$

i.e.

$$\left(I - \frac{\Delta t}{\Delta x^2} A \right) \mathbf{u}^{n+1} = \mathbf{u}^n$$

This requires the solution of a linear system to get from $\mathbf{u}^n$ to $\mathbf{u}^{n+1}$

Heat equation on a regular geometry

Suppose $\mathbf{u}^0$ is given (the initial condition).

Forward Euler (explicit)

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{\Delta t}{\Delta x^2} A \mathbf{u}^n$$

Backward Euler (implicit)

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{\Delta t}{\Delta x^2} A \mathbf{u}^{n+1}$$

i.e.

$$\left(I - \frac{\Delta t}{\Delta x^2} A \right) \mathbf{u}^{n+1} = \mathbf{u}^n$$

This requires the solution of a linear system to get from $\mathbf{u}^n$ to $\mathbf{u}^{n+1}$

Theory for finite differences

The same concepts that were discussed for ODEs can be applied to these methods for solving PDEs

Truncation error: defined analogously to with ODEs, and is $\mathcal{O}(\delta t + \delta x^2)$ for both forward and backward Euler

Convergence: does numerical solution converge to the true solution as $\delta t, \delta x \to 0$?

Stability: Different method of definition, similar conclusions to those for ODEs

- Forward Euler: require $\frac{\delta t}{\delta x^2} < \frac{1}{2}$ (conditionally stable)
    - For 2D heat equation $u_t = u_{xx} + u_{yy}$ this generalises to $\frac{\delta t}{\delta x^2 + \delta y^2} < \frac{1}{8}$
    - Refine mesh by factor of 10 $\Rightarrow \delta t$ needs to get 100 times smaller..
    - These are known as **CFL conditions** (Courant-Friedrichs-Lewy conditions)
- Backward Euler: unconditionally stable

Theory for finite differences

The same concepts that were discussed for ODEs can be applied to these methods for solving PDEs

Truncation error: defined analogously to with ODEs, and is $\mathcal{O}(\delta t + \delta x^2)$ for both forward and backward Euler

Convergence: does numerical solution converge to the true solution as $\delta t, \delta x \to 0$?

Stability: Different method of definition, similar conclusions to those for ODEs

- Forward Euler: require $\frac{\delta t}{\delta x^2} < \frac{1}{2}$ (conditionally stable)
  - For 2D heat equation $u_t = u_{xx} + u_{yy}$ this generalises to $\frac{\delta t}{\delta x^2 + \delta y^2} < \frac{1}{8}$
  - Refine mesh by factor of 10 $\Rightarrow$ δt needs to get 100 times smaller..
  - These are known as **CFL conditions** (Courant-Friedrichs-Lewy conditions)
- Backward Euler: unconditionally stable

Theory for finite differences

The same concepts that were discussed for ODEs can be applied to these methods for solving PDEs

Truncation error: defined analogously to with ODEs, and is $\mathcal{O}(\delta t + \delta x^2)$ for both forward and backward Euler

Convergence: does numerical solution converge to the true solution as $\delta t, \delta x \to 0$?

Stability: Different method of definition, similar conclusions to those for ODEs

- Forward Euler: require $\frac{\delta t}{\delta x^2} < \frac{1}{2}$ (conditionally stable)
  - For 2D heat equation $u_t = u_{xx} + u_{yy}$ this generalises to $\frac{\delta t}{\delta x^2 + \delta y^2} < \frac{1}{8}$
  - Refine mesh by factor of 10 $\Rightarrow$ δt needs to get 100 times smaller..
  - These are known as **CFL conditions** (Courant-Friedrichs-Lewy conditions)
- Backward Euler: unconditionally stable

Theory for finite differences

The same concepts that were discussed for ODEs can be applied to these methods for solving PDEs

Truncation error: defined analogously to with ODEs, and is $\mathcal{O}(\delta t + \delta x^2)$ for both forward and backward Euler

Convergence: does numerical solution converge to the true solution as $\delta t, \delta x \to 0$?

Stability: Different method of definition, similar conclusions to those for ODEs

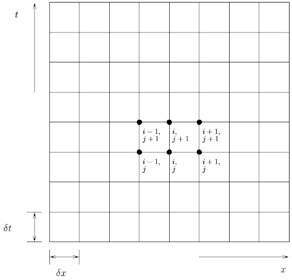
- Forward Euler: require $\frac{\delta t}{\delta x^2} < \frac{1}{2}$ (conditionally stable)
  - For 2D heat equation $u_t = u_{xx} + u_{yy}$ this generalises to $\frac{\delta t}{\delta x^2 + \delta y^2} < \frac{1}{8}$
  - Refine mesh by factor of 10 $\Rightarrow \delta t$ needs to get 100 times smaller..
  - These are known as **CFL conditions** (Courant-Friedrichs-Lewy conditions)
- Backward Euler: unconditionally stable

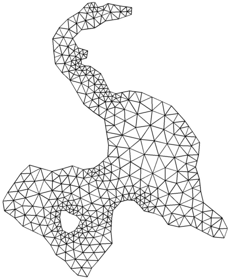
The finite element method

Advantages of the FE method over the FD method

Main advantages of FE over FD

1. Deal with Neumann boundary conditions in a natural (systematic) way
2. Deals with irregular geometries much more easily





The finite element method

Stages

1. Convert equation from **strong form** to **weak form**
2. Convert infinite-dimensional problem into a finite dimensional one
3. Set up the finite element linear system to be solved

Weak form of Poisson's equation

Consider Poisson's equation:

$$\nabla^2 u + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 \qquad \text{on } \Gamma_1 \\ \boldsymbol{\nabla} u \cdot \mathbf{n} &= g \qquad \text{on } \Gamma_2 \end{aligned}$$

Weak form

Multiply by a test function v satisfying $v = 0$ on Γ_1, and integrate:

$$\begin{aligned} v\left(\nabla^2 u\right) &= -fv \\ \int_\Omega v\left(\nabla^2 u\right) \mathrm{d}V &= -\int_\Omega fv \,\mathrm{d}V \\ \int_{\partial\Omega} v\,(\boldsymbol{\nabla} u \cdot \mathbf{n})\,\mathrm{d}S - \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}V &= -\int_\Omega fv \,\mathrm{d}V \\ \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}V &= \int_\Omega fv \,\mathrm{d}V + \int_{\Gamma_2} gv \,\mathrm{d}S \end{aligned}$$

Weak form of Poisson's equation

Consider Poisson's equation:

$$\nabla^2 u + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 \qquad \text{on } \Gamma_1 \\ \boldsymbol{\nabla} u \cdot \mathbf{n} &= g \qquad \text{on } \Gamma_2 \end{aligned}$$

Weak form

Multiply by a test function v satisfying $v = 0$ on Γ_1, and integrate:

$$\begin{aligned} v\left(\nabla^2 u\right) &= -fv \\ \int_\Omega v\left(\nabla^2 u\right) \mathrm{d}V &= -\int_\Omega fv \,\mathrm{d}V \\ \int_{\partial\Omega} v\left(\boldsymbol{\nabla} u \cdot \mathbf{n}\right) \mathrm{d}S - \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}V &= -\int_\Omega fv \,\mathrm{d}V \\ \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}V &= \int_\Omega fv \,\mathrm{d}V + \int_{\Gamma_2} gv \,\mathrm{d}S \end{aligned}$$

Weak form of Poisson's equation

Let $\mathcal{V}$ be the space of all differentiable functions on Ω (more precisely, $\mathcal{V}$ is the Sobolev space $H^1(\Omega)$). Let

$$\mathcal{V}_0 = \{v \in \mathcal{V} \,:\, v = 0 \text{ on } \Gamma_1\}$$

Weak form

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}V = \int_\Omega fv \, \mathrm{d}V + \int_{\Gamma_2} gv \, \mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

Example

Solve $\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 1$, $u(0) = u(1) = 0$ vs Find differentiable u satisfying $u(0) = u(1) = 0$ and: $\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = -\int_0^1 v\,\mathrm{d}x$ for all v s.t. $v(0) = v(1) = 0$

Weak form of Poisson's equation

Let $\mathcal{V}$ be the space of all differentiable functions on Ω (more precisely, $\mathcal{V}$ is the Sobolev space $H^1(\Omega)$). Let

$$\mathcal{V}_0 = \{v \in \mathcal{V} \,:\, v = 0 \text{ on } \Gamma_1\}$$

Weak form

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}V = \int_\Omega f v \,\mathrm{d}V + \int_{\Gamma_2} g v \,\mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

Example

Solve $\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 1$, $u(0) = u(1) = 0$

vs

Find differentiable u satisfying $u(0) = u(1) = 0$ and: $\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = -\int_0^1 v\,\mathrm{d}x$ for all v s.t. $v(0) = v(1) = 0$

Weak form of Poisson's equation

Let $\mathcal{V}$ be the space of all differentiable functions on Ω (more precisely, $\mathcal{V}$ is the Sobolev space $H^1(\Omega)$). Let

$$\mathcal{V}_0 = \{v \in \mathcal{V} \ : \ v = 0 \text{ on } \Gamma_1\}$$

Weak form

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}V = \int_\Omega f v \, \mathrm{d}V + \int_{\Gamma_2} g v \, \mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

Example

Solve $\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 1$, $u(0) = u(1) = 0$

vs

Find differentiable u satisfying $u(0) = u(1) = 0$ and: $\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x} \frac{\mathrm{d}v}{\mathrm{d}x} \, \mathrm{d}x = -\int_0^1 v \, \mathrm{d}x$ for all v s.t. $v(0) = v(1) = 0$

FEM discretisation

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}V = \int_\Omega f v \, \mathrm{d}V + \int_{\Gamma_2} g v \, \mathrm{d}S \qquad \text{for all } v \in \mathcal{V}_0$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\phi_1 \, \mathrm{d}V & \int_\Omega \nabla\phi_1 \cdot \nabla\phi_2 \, \mathrm{d}V \\ \int_\Omega \nabla\phi_2 \cdot \nabla\phi_1 \, \mathrm{d}V & \int_\Omega \nabla\phi_2 \cdot \nabla\phi_2 \, \mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f\phi_1 \, \mathrm{d}V + \int_{\Gamma_2} g\phi_1 \, \mathrm{d}S \\ \int_\Omega f\phi_2 \, \mathrm{d}V + \int_{\Gamma_2} g\phi_2 \, \mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla v \, \mathrm{d}V = \int_\Omega f v \, \mathrm{d}V + \int_{\Gamma_2} g v \, \mathrm{d}S \qquad \text{for all } v \in \mathcal{V}_0^h$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\phi_1 \, \mathrm{d}V & \int_\Omega \nabla\phi_1 \cdot \nabla\phi_2 \, \mathrm{d}V \\ \int_\Omega \nabla\phi_2 \cdot \nabla\phi_1 \, \mathrm{d}V & \int_\Omega \nabla\phi_2 \cdot \nabla\phi_2 \, \mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f\phi_1 \, \mathrm{d}V + \int_{\Gamma_2} g\phi_1 \, \mathrm{d}S \\ \int_\Omega f\phi_2 \, \mathrm{d}V + \int_{\Gamma_2} g\phi_2 \, \mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla v \,\mathrm{d}V = \int_\Omega f v \,\mathrm{d}V + \int_{\Gamma_2} g v \,\mathrm{d}S \qquad \text{for all } v \in \mathcal{V}_0^h$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_1 \cdot \nabla\phi_2 \,\mathrm{d}V \\ \int_\Omega \nabla\phi_2 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_2 \cdot \nabla\phi_2 \,\mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f\phi_1 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_1 \,\mathrm{d}S \\ \int_\Omega f\phi_2 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_2 \,\mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla \phi_j \,\mathrm{d}V = \int_\Omega f \phi_j \,\mathrm{d}V + \int_{\Gamma_2} g \phi_j \,\mathrm{d}S \qquad \text{for } j = 1, 2$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_1 \cdot \nabla\phi_2 \,\mathrm{d}V \\ \int_\Omega \nabla\phi_2 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_2 \cdot \nabla\phi_2 \,\mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f\phi_1 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_1 \,\mathrm{d}S \\ \int_\Omega f\phi_2 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_2 \,\mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla \phi_j \, \mathrm{d}V = \int_\Omega f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S \qquad \text{for } j = 1, 2$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

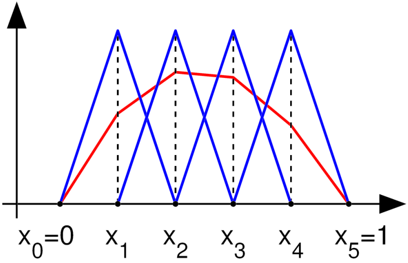
(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

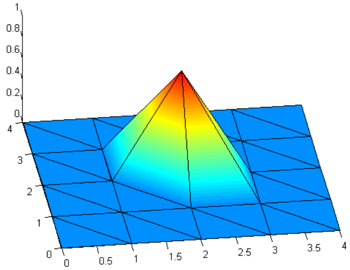
$$u_h = \alpha \phi_1 + \beta \phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla \phi_1 \cdot \nabla \phi_1 \, \mathrm{d}V & \int_\Omega \nabla \phi_1 \cdot \nabla \phi_2 \, \mathrm{d}V \\ \int_\Omega \nabla \phi_2 \cdot \nabla \phi_1 \, \mathrm{d}V & \int_\Omega \nabla \phi_2 \cdot \nabla \phi_2 \, \mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f \phi_1 \, \mathrm{d}V + \int_{\Gamma_2} g \phi_1 \, \mathrm{d}S \\ \int_\Omega f \phi_2 \, \mathrm{d}V + \int_{\Gamma_2} g \phi_2 \, \mathrm{d}S \end{bmatrix}$$

Basis functions





FEM discretisations

Take

$$V_h = \mathrm{span}\{\phi_1, \phi_2, \ldots, \phi_N\}$$

(satisfying $\phi_j = 0$ on Γ_1) so

$$u_h = \alpha_1\phi_1 + \ldots + \alpha_N\phi_N$$

Let the stiffness matrix and RHS vector be given by

$$\begin{aligned} K_{jk} &= \int_\Omega \nabla\phi_j \cdot \nabla\phi_k \,\mathrm{d}V \\ b_j &= \int_\Omega f\phi_j \,\mathrm{d}V + \int_{\Gamma_2} g\phi_j \,\mathrm{d}S \end{aligned}$$

and solve

$$K \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} = \mathbf{b}$$

FEM discretisations

Let

$$\begin{aligned} K_{jk} &= \int_\Omega \nabla\phi_j \cdot \nabla\phi_k \,\mathrm{d}V && \text{stiffness matrix} \\ M_{jk} &= \int_\Omega \phi_j\phi_k \,\mathrm{d}V && \text{mass matrix} \\ b_j &= \int_\Omega f\phi_j \,\mathrm{d}V + \int_{\Gamma_2} g\phi_j \,\mathrm{d}S \end{aligned}$$

FEM discretisations

Laplace's equation: $\nabla^2 u + f = 0 \quad \rightarrow \quad K\mathbf{U} = \mathbf{b}$

Heat equation:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f \quad \rightarrow \quad M\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} + K\mathbf{U} = \mathbf{b}$$

Time-discretised heat equation:

$$\frac{u^{n+1} - u^n}{\Delta t} = \nabla^2 u^{n+1} + f^{n+1} \quad \rightarrow \quad M\mathbf{U}^{n+1} + \Delta t\, K\mathbf{U}^{n+1} = M\mathbf{U}^n + \Delta t\,\mathbf{b}^{n+1}$$

Anisotropic diffusion

Suppose we have an anisotropic diffusion tensor D (symmetric, positive definite), for example, in Poisson's equation:

$$\nabla \cdot (D \boldsymbol{\nabla} u) + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 \qquad \text{on } \Gamma_1 \\ (D \boldsymbol{\nabla} u) \cdot \mathbf{n} &= g \qquad \text{on } \Gamma_2 \end{aligned}$$

The weak form is: find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega (D \boldsymbol{\nabla} u) \cdot \boldsymbol{\nabla} v \, \mathrm{d}V = \int_\Omega f v \, \mathrm{d}V + \int_{\Gamma_2} g v \, \mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

and the only change in the FEM discretisation is that the stiffness matrix becomes

$$K_{jk} = \int_\Omega \boldsymbol{\nabla} \phi_j \cdot (D \boldsymbol{\nabla} \phi_k) \, \mathrm{d}V$$

Anisotropic diffusion

Suppose we have an anisotropic diffusion tensor D (symmetric, positive definite), for example, in Poisson's equation:

$$\nabla \cdot (D\boldsymbol{\nabla} u) + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 \qquad \text{on } \Gamma_1 \\ (D\boldsymbol{\nabla} u)\cdot \mathbf{n} &= g \qquad \text{on } \Gamma_2 \end{aligned}$$

The weak form is: find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega (D\boldsymbol{\nabla} u)\cdot \boldsymbol{\nabla} v \,\mathrm{d}V = \int_\Omega fv \,\mathrm{d}V + \int_{\Gamma_2} gv \,\mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

and the only change in the FEM discretisation is that the stiffness matrix becomes

$$K_{jk} = \int_\Omega \boldsymbol{\nabla}\phi_j \cdot (D\boldsymbol{\nabla}\phi_k)\,\mathrm{d}V$$

Implementing Dirichlet boundary conditions

In practice, rather using the basis functions in $\mathcal{V}_0^h$ (i.e. bases satisfying $\phi_i = 0$ on Γ_1), we use $\mathcal{V}^h$, i.e. all the basis functions corresponding to all nodes in the mesh.

We then impose (any) Dirichlet boundary conditions by altering the appropriate rows of the linear system, for example, for $K\mathbf{U} = b$, if we want to impose $U_1 = c$

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1N} \\ K_{21} & K_{22} & \dots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

Implementing Dirichlet boundary conditions

In practice, rather using the basis functions in $\mathcal{V}_0^h$ (i.e. bases satisfying $\phi_i = 0$ on Γ_1), we use $\mathcal{V}^h$, i.e. all the basis functions corresponding to all nodes in the mesh.

We then impose (any) Dirichlet boundary conditions by altering the appropriate rows of the linear system, for example, for $K\mathbf{U} = b$, if we want to impose $U_1 = c$

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1N} \\ K_{21} & K_{22} & \dots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

Implementing Dirichlet boundary conditions

In practice, rather using the basis functions in $\mathcal{V}_0^h$ (i.e. bases satisfying $\phi_i = 0$ on Γ_1), we use $\mathcal{V}^h$, i.e. all the basis functions corresponding to all nodes in the mesh.

We then impose (any) Dirichlet boundary conditions by altering the appropriate rows of the linear system, for example, for $K\mathbf{U} = b$, if we want to impose $U_1 = c$

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ K_{21} & K_{22} & \dots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} c \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

FEM stages

Solve:

$$\nabla \cdot (D \boldsymbol{\nabla} u) + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= u^* && \text{on } \Gamma_1 \\ (D \boldsymbol{\nabla} u) \cdot \mathbf{n} &= g && \text{on } \Gamma_2 \end{aligned}$$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:

$$\begin{aligned} K_{jk} &= \int_\Omega \boldsymbol{\nabla} \phi_j \cdot (D \boldsymbol{\nabla} \phi_k) \, \mathrm{d}V \\ b_j &= \int_\Omega f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S \end{aligned}$$

3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system