Lecture 04

Solving Time-Dependent Problems

Anders Logg

May 18, 2018

What will you learn?

- ***How to solve time-dependent problems***
- *How to formulate the initial–boundary value problem*
- *How to derive the variational problem*
- *How to discretize in time*
- *How to solve systems of time-dependent problems*
- *FEniCS programming*
    - Creating system (mixed) function spaces
    - Accessing system components
    - Time-stepping
    - Using constants to wrap numeric values
    - Assigning function values
    - Saving time-series to file
- *Exercise*

How to formulate the initial–boundary value problem

Partial differential equation

Abstract time-dependent PDE

$$\dot{u} + \mathcal{A}(u) = f \quad \text{in } \Omega \times (0, T)$$

Partial differential equation

Abstract time-dependent PDE

$$\dot{u} + \mathcal{A}(u) = f \quad \text{in } \Omega \times (0, T)$$

u is the solution to be computed

$\dot{u} = \frac{\partial u}{\partial t}$ is the time-derivative of u

f is a given source term

Ω is the computational domain

$\mathcal{A}$ is a nonlinear differential operator

T is the final time

Examples

The heat equation

$$\dot{u} - \epsilon \Delta u = f$$

Examples

The heat equation

$$\dot{u} - \epsilon \Delta u = f$$

The advection–diffusion equation

$$\dot{u} + \beta \cdot \nabla u - \epsilon \Delta u = f$$

Examples

The heat equation

$$\dot{u} - \epsilon \Delta u = f$$

The advection–diffusion equation

$$\dot{u} + \beta \cdot \nabla u - \epsilon \Delta u = f$$

The advection–diffusion–reaction equation

$$\dot{u} + \beta \cdot \nabla u - \epsilon \Delta u + u = f$$

Initial and boundary conditions

Initial conditions

$$u = u_0 \quad \text{on } \Omega \times \{0\}$$

Initial and boundary conditions

Initial conditions

$$u = u_0 \quad \text{on } \Omega \times \{0\}$$

Dirichlet boundary condition

$$u = u_{\mathrm{D}} \quad \text{on } \Gamma_{\mathrm{D}} \times (0, T)$$

Initial and boundary conditions

Initial conditions

$$u = u_0 \quad \text{on } \Omega \times \{0\}$$

Dirichlet boundary condition

$$u = u_{\mathrm{D}} \quad \text{on } \Gamma_{\mathrm{D}} \times (0, T)$$

Neumann boundary condition

$$-\epsilon \partial_n u = g \quad \text{on } \Gamma_{\mathrm{N}} \times (0, T)$$

Initial and boundary conditions

Initial conditions

$$u = u_0 \quad \text{on } \Omega \times \{0\}$$

Dirichlet boundary condition

$$u = u_{\mathrm{D}} \quad \text{on } \Gamma_{\mathrm{D}} \times (0, T)$$

Neumann boundary condition

$$-\epsilon \partial_n u = g \quad \text{on } \Gamma_{\mathrm{N}} \times (0, T)$$

Note that both u_{D} and g may be time-dependent

How to derive the variational problem

Recall from Lecture 3: The FEM cookbook (for a nonlinear PDE)

Partial differential equation (strong form)

$$\mathcal{A}(u) = f \tag{1}$$

Continuous variational problem (weak form)

Find $u \in V$ such that

$$F(u; v) = 0 \quad \forall v \in V \tag{2}$$

Discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$F(u_h; v) = 0 \quad \forall v \in V_h \tag{3}$$

Discrete system of equations (nonlinear system)

$$R(U) = 0 \tag{4}$$

From strong to weak form: (1) → (2)

Partial differential equation (strong form)

$$\dot{u} + \mathcal{A}(u) - f = 0$$

From strong to weak form: (1) → (2)

Partial differential equation (strong form)

$$\dot{u} + \mathcal{A}(u) - f = 0$$

Multiply by a test function v and integrate over the domain Ω:

$$\underbrace{\int_\Omega (\dot{u} + \mathcal{A}(u) - f)\, v \,\mathrm{d}x}_{=F_t(u;v)} = 0$$

From strong to weak form: (1) $\rightarrow$ (2)

Partial differential equation (strong form)

$$\dot{u} + \mathcal{A}(u) - f = 0$$

Multiply by a test function v and integrate over the domain Ω:

$$\underbrace{\int_\Omega (\dot{u} + \mathcal{A}(u) - f)\, v \,\mathrm{d}x}_{=F_t(u;v)} = 0$$

$\Rightarrow$ Continuous variational problem (weak form)

Find $u \in V$ such that

$$F_t(u; v) = 0 \quad \forall v \in V \quad \forall t \in (0, T)$$

From weak form to finite element method: (2) $\to$ (3)

Continuous variational problem (weak form)

Find $u \in V$ such that

$$F_t(u; v) = 0 \quad \forall v \in V \quad \forall t \in (0, T)$$

From weak form to finite element method: (2) $\rightarrow$ (3)

Continuous variational problem (weak form)

Find $u \in V$ such that

$$F_t(u; v) = 0 \quad \forall v \in V \quad \forall t \in (0, T)$$

Let V_h be a discrete finite element subspace of V

From weak form to finite element method: (2) $\to$ (3)

Continuous variational problem (weak form)

Find $u \in V$ such that

$$F_t(u; v) = 0 \quad \forall v \in V \quad \forall t \in (0, T)$$

Let V_h be a discrete finite element subspace of V

$\Rightarrow$ Semi-discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$F_t(u_h; v) = L(v) \quad \forall v \in V_h \quad \forall t \in (0, T)$$

How to discretize in time

The method of lines

The semi-discrete variational problem defines an ODE (ordinary differential equation) for the vector of degrees of freedom $U = U(t)$

The method of lines

The semi-discrete variational problem defines an ODE (ordinary differential equation) for the vector of degrees of freedom $U = U(t)$

Can be solved using an appropriate numerical integration scheme

The method of lines

The semi-discrete variational problem defines an ODE (ordinary differential equation) for the vector of degrees of freedom $U = U(t)$

Can be solved using an appropriate numerical integration scheme

Examples:

- Forward (explicit) Euler (avoid)
- Backward (implicit) Euler
- Midpoint method
- Runge–Kutta methods
- Multistep methods
- Space-time FEM

Backward Euler method

- Partition $(0, T)$ into time intervals of length Δt
- Approximate $\dot{u}$ by $(u_h^{n+1} - u_h^n)/\Delta t$
- Approximate u by u_h^{n+1}

Backward Euler method

- Partition $(0, T)$ into time intervals of length Δt
- Approximate $\dot{u}$ by $(u_h^{n+1} - u_h^n)/\Delta t$
- Approximate u by u_h^{n+1}

Fully-discrete variational problem (time-stepping scheme)

Find $u_h^{n+1} \in V_h$ such that

$$\int_\Omega (\Delta t^{-1}(u_h^{n+1} - u_h^n) + \mathcal{A}(u_h^{n+1}) - f^{n+1})\, v \,\mathrm{d}x = 0 \quad \forall v \in V$$

Backward Euler method

- Partition $(0, T)$ into time intervals of length Δt
- Approximate $\dot{u}$ by $(u_h^{n+1} - u_h^n)/\Delta t$
- Approximate u by u_h^{n+1}

Fully-discrete variational problem (time-stepping scheme)

Find $u_h^{n+1} \in V_h$ such that

$$\int_\Omega (\Delta t^{-1}(u_h^{n+1} - u_h^n) + \mathcal{A}(u_h^{n+1}) - f^{n+1})\, v \,\mathrm{d}x = 0 \quad \forall v \in V$$

A linear problem if $\mathcal{A}$ is linear

A nonlinear problem if $\mathcal{A}$ is nonlinear

Solve for the degrees of freedom $U \in \mathbb{R}^N$ in each time step

Example: The heat equation

Fully-discrete variational problem

$$\int_\Omega \Delta t^{-1}(u_h^{n+1} - u_h^n)\, v + \epsilon \nabla u_h^{n+1} \cdot \nabla v - f^{n+1}\, v \,\mathrm{d}x = 0$$

Example: The heat equation

Fully-discrete variational problem

$$\int_\Omega \Delta t^{-1}(u_h^{n+1} - u_h^n)\, v + \epsilon \nabla u_h^{n+1} \cdot \nabla v - f^{n+1}\, v \,\mathrm{d}x = 0$$

Rearrange:

Fully-discrete variational problem

$$\underbrace{\int_\Omega u_h^{n+1} v + \Delta t\, \epsilon\, \nabla u_h^{n+1} \cdot \nabla v \,\mathrm{d}x}_{=a(u_h^{n+1}, v)} = \underbrace{\int_\Omega u_h^n v + \Delta t\, f^{n+1}\, v \,\mathrm{d}x}_{=L_{n+1}(v)}$$

How to solve systems of time-dependent problems

Systems of partial differential equations

Consider the PDE system

$$\begin{aligned}\dot{u}_1 + \mathcal{A}_1(u_1, u_2) &= f_1\\ \dot{u}_2 + \mathcal{A}_2(u_1, u_2) &= f_2\end{aligned}$$

Systems of partial differential equations

Consider the PDE system

$$\begin{aligned}\dot{u}_1 + \mathcal{A}_1(u_1, u_2) &= f_1 \\ \dot{u}_2 + \mathcal{A}_2(u_1, u_2) &= f_2\end{aligned}$$

Multiply by test functions v_1 and v_2, integrate and sum up:

$$\int_\Omega (\dot{u}_1 + \mathcal{A}_1(u_1, u_2) - f_1)\, v_1 + (\dot{u}_2 + \mathcal{A}_2(u_1, u_2) - f_2)\, v_2 \,\mathrm{d}x = 0$$

Backward Euler method for the system

Fully-discrete variational problem for system

Find $(u_{h,1}^{n+1}, u_{h,2}^{n+1}) \in V_h \times V_h$ such that

$$\begin{aligned}
\int_\Omega & \Delta t^{-1}(u_{h,1}^{n+1} - u_{h,1}^{n})\, v_1 + \Delta t^{-1}(u_{h,2}^{n+1} - u_{h,2}^{n})\, v_2 \\
& + \mathcal{A}_1(u_{h,1}^{n+1}, u_{h,2}^{n+1})\, v_1 + \mathcal{A}_2(u_{h,1}^{n+1}, u_{h,2}^{n+1})\, v_2 \\
& - f_1^{n+1} v_1 - f_2^{n+1} v_2 \,\mathrm{d}x = 0
\end{aligned}$$

for all $(v_1, v_2) \in V_h \times V_h$

Backward Euler method for the system

Fully-discrete variational problem for system

Find $(u_{h,1}^{n+1}, u_{h,2}^{n+1}) \in V_h \times V_h$ such that

$$
\begin{aligned}
\int_\Omega & \Delta t^{-1}(u_{h,1}^{n+1} - u_{h,1}^{n})\, v_1 + \Delta t^{-1}(u_{h,2}^{n+1} - u_{h,2}^{n})\, v_2 \\
& + \mathcal{A}_1(u_{h,1}^{n+1}, u_{h,2}^{n+1})\, v_1 + \mathcal{A}_2(u_{h,1}^{n+1}, u_{h,2}^{n+1})\, v_2 \\
& - f_1^{n+1} v_1 - f_2^{n+1} v_2 \,\mathrm{d}x = 0
\end{aligned}
$$

for all $(v_1, v_2) \in V_h \times V_h$

A linear problem if $\mathcal{A}_1$ and $\mathcal{A}_2$ are linear

A nonlinear problem if $\mathcal{A}_1$ and $\mathcal{A}_2$ are nonlinear

Solve for the degrees of freedom $U \in \mathbb{R}^{2N}$ in each time step

FEniCS programming

Creating system (mixed) function spaces

System (mixed) function spaces can be created using `VectorFunctionSpace` instead of `FunctionSpace`:

```
V = VectorFunctionSpace(mesh, 'P', 1)
```

Creating system (mixed) function spaces

System (mixed) function spaces can be created using `VectorFunctionSpace` instead of `FunctionSpace`:

```
V = VectorFunctionSpace(mesh, 'P', 1)
```

The number of components is equal to the space dimension of the mesh but can also be chosen by an additional argument:

```
V = VectorFunctionSpace(mesh, 'P', 1, dim)
```

Creating system (mixed) function spaces

System (mixed) function spaces can be created using `VectorFunctionSpace` instead of `FunctionSpace`:

```
V = VectorFunctionSpace(mesh, 'P', 1)
```

The number of components is equal to the space dimension of the mesh but can also be chosen by an additional argument:

```
V = VectorFunctionSpace(mesh, 'P', 1, dim)
```

Mixed function spaces can also be created from mixed finite elements:

```
P1 = FiniteElement('P', triangle, 1)
V = FunctionSpace(mesh, P1*P1)
```

Accessing system components: `split(u)`

If `u` is a vector (mixed) function, its components can be accessed using the `split()` operator:

```
u1, u2 = split(u)
```

Accessing system components: `split(u)`

If `u` is a vector (mixed) function, its components can be accessed using the `split()` operator:

```
u1, u2 = split(u)
```

This is equivalent to

```
u1 = u[0]
u2 = u[1]
```

Accessing system components: `u.split()`

Note that `split(u)` only creates `symbolic expressions` for the components of `u`

To extract the components of `u` as `Function`s that can be plotted or save to file, use `u.split()`:

```
u1, u2 = u.split()
```

Time-stepping

Time-stepping is performed using a standard Python loop:

```
t = 0.0
for n in range(num_steps):

    # Update current time
    t += dt
    print("t =", t)

    # Solve variational problem
    ...

    # Save solution to file
    ...

    # Update previous solution
    ...
```

Using constants to wrap numeric values

Changing numeric values (float variables) in forms triggers code generation and can be costly

Avoid by wrapping each numeric value as a `Constant`

```
k = Constant(dt)
```

Using constants to wrap numeric values

Changing numeric values (float variables) in forms triggers code generation and can be costly

Avoid by wrapping each numeric value as a `Constant`

```
k = Constant(dt)
```

Use `k` to define the variational problem

Use `dt` to update the current time

Also very useful for material parameters

Assigning function values

At the end of each time step, we need to make the assignment

$$\texttt{un} \quad \leftarrow \quad \texttt{u}$$

The `Function` `un` represents the previous value u_h^n

The `Function` `u` represents the next value u_h^{n+1}

Assigning function values

At the end of each time step, we need to make the assignment

$$\texttt{un} \quad \leftarrow \quad \texttt{u}$$

The `Function` `un` represents the previous value u_h^n

The `Function` `u` represents the next value u_h^{n+1}

Use the `assign` function:

```
un.assign(u)
```

Note that `un = u` does not work as expected!

Saving time-series to file

To save a time-series, create a VTK file:

```
vtkfile = File('solution.pvd')
```

Then write to the file in each time step:

```
for n in range(num_steps):

    ...

    # Save solution to file
    vtkfile << u
```

Exercise

Exercise 4: Solving Time-Dependent Problems

In this exercise, we will solve the following system of nonlinear advection-diffusion-reaction equations with FEniCS:

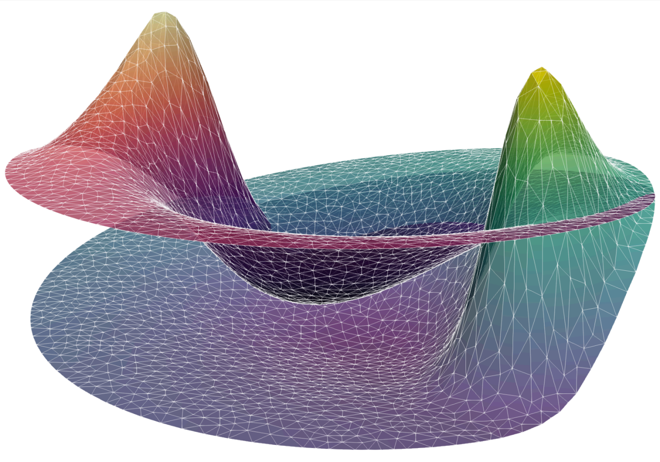
$$
\begin{aligned}
\frac{\partial u_1}{\partial t} + \beta \cdot \nabla u_1 - \epsilon \Delta u_1 = f_1 - K u_1 u_2^2 &\quad \text{in } \Omega \\
\frac{\partial u_2}{\partial t} + \beta \cdot \nabla u_2 - \epsilon \Delta u_2 = f_2 - 2K u_1 u_2^2 &\quad \text{in } \Omega \\
u_1 = u_2 = 0 &\quad \text{at } t = 0 \\
-\partial_n u_1 = -\partial_n u_2 = 0 &\quad \text{on } \partial\Omega
\end{aligned}
$$

This a model of the chemical reaction $A + 2B \to C$ where $u_1 = [A]$ is the concentration of A and $u_2 = [B]$ is the concentration of B. Both A and B are being continuously added to the system through the source terms f_1 and f_2, and mixed through diffusion (diffusivity ϵ) and advection (velocity β).

Exercise 4: Problem data

- Ω is the unit disc
- $\epsilon = 0.1$
- $\beta(x, y) = 5(-y, x)$
- $K = 10$
- $f_1(x, y) = \exp(-50((x + 0.75)^2 + y^2))$
- $f_2(x, y) = \exp(-50((x - 0.75)^2 + y^2))$
- 500 time steps of size $\Delta t = 0.01$

Exercise 4: Solution



Solution to Exercise 4 plotted in Paraview using the "Warp By Scalar" filter for the two components u_1 and u_2.