Randvärdesproblem:

Ekvation: $-u''(x) = f(x), \quad x \in [a; b]$

Randvillkor typ 1: $u(a) = 0, \quad u(b) = 0$

Randvillkor typ 2: $u'(a) = 0, \quad u'(b) = 0$

Vi kan skriva ett generellt randvillkor som :

$$u'(a) = \kappa u(a), \qquad u'(b) = \kappa u(b)$$

$\kappa = 0$ blir av typ 2, $\kappa \to \infty$ blir av typ 1.

Partialintegration (kom ihåg):

$$\int_a^b w''v \, dx = -\int_a^b w'v' \, dx + w'(b)v(b) - w'(a)v(a)$$

$$R(u) = -u'' - f = 0$$

$\Downarrow$ Finita element metoden

$$\int_a^b R(u)\varphi_j \, dx = 0$$

$$\Updownarrow$$

$$\int_a^b (-u'' - f)\varphi_j \, dx = \int_a^b \left(-u''\varphi_j - f\varphi_j\right) dx = \{\text{partialintegration}\} =$$

$$= \int_a^b \left(u'\varphi'_j - f\varphi_j\right) dx + u'(b)\varphi_j(b) - u'(a)\varphi_j(a) = 0$$

Använd generellt randvillkor $u'(a) = \kappa u(a)$:

$$\int_a^b u'\varphi'_j \, dx + \kappa u(b)\varphi_j(b) - \kappa u(a)\varphi_j(a) = \int_a^b f\varphi_j \, dx$$

Randvillkor i Python med DOLFIN:

```python
from dolfin import *

# Coefficient for boundary condition
class Kappa(Expression):
     def eval(self, values, x):
          if (x[0] < 0.5):
               values[0] = 0.0
          else:
               values[0] = 1.0e6

# Mesh and basis functions (function space)
mesh = UnitInterval(8)
V = FunctionSpace(mesh, "CG", 1)

# FEM formulation
u = TrialFunction(V)
v = TestFunction(V)
f = Expression("1.0")
kappa = Kappa(V)
# Equation: -u'' = f
a = (inner(grad(u), grad(v))) * dx + kappa * u * v * ds
L = (f * v) * dx

# Build linear system and solve for
# coefficients uh in linear combination
problem = VariationalProblem(a, L)
uh = problem.solve()
plot(uh)
```

Andraderivata – matris:

Använd definition av u som linjärkombination av basfunktioner:

$$u = \sum_{i=1}^{J} u_i \varphi_i(x)$$

$$\int_a^b u' \varphi'_j \, dx = \sum_{i=1}^{J} \int_a^b u_i \varphi'_i \varphi'_j \, dx, \quad j = 1, 2, \ldots, J$$

Alltså, för matriselement $\mathbf{A}_{ij}$:

$$\mathbf{A}_{ij} = \int_a^b \varphi'_j \varphi'_i \, dx$$

```
from dolfin import *

# Create mesh and define function space
mesh = UnitInterval(2)
V = FunctionSpace(mesh, "CG", 1)

# Define FEM formulation
u = TrialFunction(V)
v = TestFunction(V)
a = (inner(grad(u), grad(v))) * dx

A = assemble(a)
print A.array()
```

```
...
```

```
Matrix of size 3 x 3 has 7 nonzero entries .
[[ 2.  -2.   0.]
 [-2.   4.  -2.]
 [ 0.  -2.   2.]]
```

Konvektion-diffusion-reaktionen av en koncentration, u_1, av en art, som äts upp av en annan art med koncentrationen u_2 kan beskrivas med foljade ODE- och PDE-modeller:

ODE:

$$\begin{cases} \dot{u}_1 = -\alpha_1 u_1 u_2 \\ \dot{u}_2 = \alpha_2 u_1 u_2 - \alpha_3 u_2 \end{cases}$$

PDE:

$$\dot{u} + \beta \cdot \nabla u - \epsilon \Delta u = f(u)$$
$$[\dot{u} + \beta u' - \epsilon u'' = f(u)]$$

Elasticitet kan modelleras med en mass–fjädermodell i en ODE eller genom att använda en PDE-vågekvation:

ODE:

$$\begin{cases} \dot{x}^i = v^i \\ \dot{v}^i = \dfrac{F^i}{m^i} \\ F^i = \sum\limits_{j=0}^{N} F^{ij} \\ F^{ij} = E\left(r^{ij} - L^{ij}\right)\vec{e}^{ij} \end{cases}$$

PDE:

$$\begin{cases} \dot{u}_1 = u_2 \\ \dot{u}_2 - \nabla \cdot \sigma = 0 \\ \dot{\sigma} = \dfrac{1}{2} E\left(\nabla u_2 + \nabla u_2^T\right) \end{cases}$$

Vågfenomen kan modelleras med en mass–fjädermodell i en ODE eller genom att använda en PDE-vågekvation:

Samma ODE som ovan.

PDE:

$$\begin{cases} \dot{u}_1 = u_2 \\ \dot{u}_2 = c^2 \Delta u_1 = 0 \end{cases}$$
$$\left[\dot{u}_2 = c^2 u''_1 = 0\right]$$