Enkelt DOLFIN värmeledning demo

```python
from dolfin import *

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

# Define FEM formulation
u = TrialFunction(V)
v = TestFunction(V)
f = Expression("0.0")
k = 0.001
u0 = Function(V)
u1 = Function(V)
# Heat equation with backward
# Euler timestepping
a = (u * v + k * inner(grad(u), grad(v))) * dx
L = (u0 * v + k * f * v) * dx

problem = VariationalProblem(a, L)


# Set initial condition
for v in vertices(mesh):
     p = v.point()[0]
     i = v.index()
     if (p > 0.5):
          u1.vector()[i] = 1.0
     else:
          u1.vector()[i] = 0.0

file = File("heat.pvd")

# Timestep
T = 2.0
t = 0.0
while (t < T):
     u0.assign(u1)
     u1 = problem.solve()

     t += k

     # Save solution in VTK format
     file << u1
```

Projektion (L2)

```
# Define basis functions
V = FunctionSpace(mesh, "CG", 1)
Pf = TrialFunction(V)
v = TestFunction(V)

# Function to project
f = dolfin.Expression("exp(-10*x[0])", degree = 5)

# FEM formulation
a = Pf * v * dx
L = f * v * dx

# Assemble (build) matrix and vector,
# solve linear system and definefunction
# as linear combination
problem = dolfin.VariationalProblem(a, L)
Pf = problem.solve()
```

Projektion (L2) expanded

```
# Define basis functions
V = FunctionSpace(mesh, "CG", 1)
Pf = TrialFunction(V)
v = TestFunction(V)

# Function to project
f = dolfin.Expression("exp(-10*x[0])", degree = 5)

# FEM formulation
a = Pf * v * dx
L = f * v * dx

# Define solution functionas
# linear combination of basis
# functions
Pf = Function(V)
x = Pf.vector()

# Assemble (build) matrix and vector
A = assemble(a)
b = assemble(L)

# Solve for x^i
solver = LinearSolver()
solver.solve(A, xi, b)
```

Andraderivata — matris

```
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()
```