```julia
using DCISolver, Gridap, PDENLPModels
Ω = (-1, 1, -1, 1) # Cartesian discretization of Ω=(-1,1)^2 in 100^2 squares.
model = CartesianDiscreteModel(Ω, (100, 100))
fe_y = ReferenceFE(lagrangian, Float64, 2) # Finite-elements for the state
Xpde = TestFESpace(model, fe_y; dirichlet_tags = "boundary")
Ypde = TrialFESpace(Xpde, x -> 0.0) # y is 0 over ∂Ω
fe_u = ReferenceFE(lagrangian, Float64, 1) # Finite-elements for the control
Xcon = TestFESpace(model, fe_u)
Ycon = TrialFESpace(Xcon)
dΩ = Measure(Triangulation(model), 1) # Gridap's integration machinery
# Define the objective function f
yd(x) = -x[1]^2
f(y, u) = ∫(0.5 * (yd - y) * (yd - y) + 0.5 * 1e-2 * u * u) * dΩ
# Define the constraint operator in weak form
h(x) = -sin(7π / 8 * x[1]) * sin(7π / 8 * x[2])
c(y, u, v) = ∫(∇(v) ⊙ ∇(y) - v * u - v * h) * dΩ
# Define an initial guess for the discretized problem
x0 = zeros(num_free_dofs(Ypde) + num_free_dofs(Ycon))
# Build a GridapPDENLPModel, which implements the NLPModel API.
name = "Control elastic membrane"
nlp = GridapPDENLPModel(x0, f, dΩ, Ypde, Ycon, Xpde, Xcon, c, name = name)
dci(nlp, verbose = 1) # solve the problem with DCI
```