Modeling and Simulation Seminar

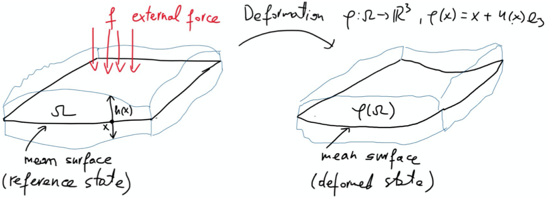
Group-Shape Optimization

Anurag Trivedi, Mohammed Parvez
anurag.trivedi@mailbox.tu-dresden.de
Parvez.Mohammed@mailbox.tu-dresden.de

July 17, 2020

Motivation

Modelling of thin membrane



Content

Problem Statement

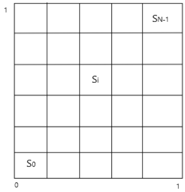
consider the following membrane model

$$\begin{aligned} -div(h\nabla u) &= \text{f in } \Omega \\ \text{u} &= 0 \text{ on } d\Omega \end{aligned}$$

(1)

→ 2D problem with $\Omega = [0, 1] \times [0, 1]$.

→ 'h ' to be piecewise constant on grid.



Admissible thickness

$$\mathrm{U}_{ad} = \{h \in L^\infty; 0 < 0.1 \leq h(x) \leq 5, \int_\Omega h(x)dx = 1\}$$

possible choice of admissible set $U_{ad} = h : \Omega \rightarrow R$.

$$U_{ad} = \{h(x) = \sum_{i=0}^{N-1} h_i . I_{s_i}(x); 0.1 \leq h_i \leq 5, \sum_{i=0}^{N-1} h_i(x) = N\}$$

Cost functional

To be minimized subject to a partial differential equation problem posed on a domain $\Omega \subset R^2$

$$min_h(J(h) = \int_\Omega j(u)dx)$$

Subject to - div(h ∇u) = f in Ω

with u = 0 on dΩ

We choose the optimization functional to be for compliance(rigidity) $\rightarrow$ j(u) = f u

Iterative Scheme

$$h^{n+1} = h^n - \mu J'(u)$$

Now the gradient of J is given by

$$\nabla J(u) = \nabla u . \nabla p$$

where 'u' is the solution to equation(1) with h = h^n

find 'p' is the ad joint state given by the solution to

$$-div(h^n \nabla p) = \text{j'(u) in } \Omega$$
$$\text{p= 0 on } d\Omega$$

$$-div(h^n \nabla p) = \text{-f in } \Omega$$
$$\text{p= 0 on } d\Omega$$

$\rightarrow$ p = -u

Hence

$$\text{J'(u)} = -\nabla u * \nabla u$$
$$h^{n+1} = h^n + \mu(\nabla u * \nabla u)$$

So projection is a must.

$\mathbf{L}^2$*Projection*

we are considering the following norm
i.e

$$\|f(x)\|^2_{L^2\Omega} = \sum_{i=1}^{N} \int_{s_i} |f(x)|^2 dx$$

Hence the projection should be such that

$$\|h^{n+1} - \tilde{h}^{n+1}\|^2_{L^2} = \sum_{i=0}^{N} \int_{s_i} |h_i^{n+1} - \tilde{h}^{n+1}|^2 dx \rightarrow \text{minimum}$$

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Approach

shapeOpt.cc ×

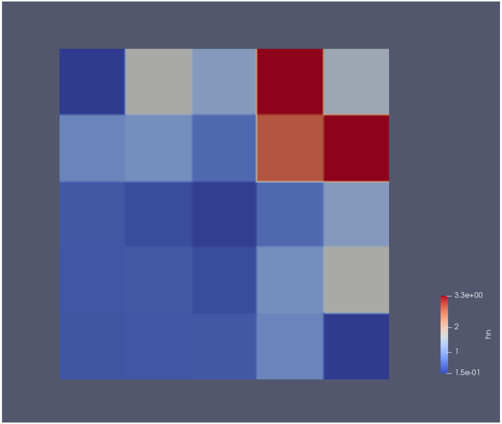
so_preparation > src > shapeOpt.cc > ShapeOptimization > closeTimestep(AdaptInfo *)

```
// ===== ProblemInstatBase methods ===================================


void closeTimestep(AdaptInfo *adaptInfo)
{
  /*
    Here we have to do magic - use the computed solution to compute the new
    iterate hn_tilde and then form the projection ...
  */
  *hn_old_<< valueOf(hn_);
  *hn_tilde_ << valueOf(hn_old_) + 0.05*(gradientOf(this->getOldSolution(0))*gradientOf(this->getOldSolution(0
  *hn_ << 0.0;

  double lambda_=2*(integrate(valueOf(hn_tilde_))-1.0)/(N_x*N_y);

  double integral= 0.0;
  cout<<N_x<<endl;
  cout<<N_y<<endl;

  for(int i=0; i<N_x; i++){
    for(int j=0; j<N_y; j++){
      WorldVector<double> center_;
      center_[0]= (0.5 + i)/N_x;
      center_[1]= (0.5 + j)/N_y;

      *indicator_ << function_(rectangle(center_, 1.0/N_x, 1.0/N_y), X());
      integral = integrate(valueOf(hn_tilde_)*valueOf(indicator_));

      *hn_<< valueOf(hn_) + 0.5*(N_x*N_y)*(lambda_+2*integral)*valueOf(indicator_);
    }
  }
```

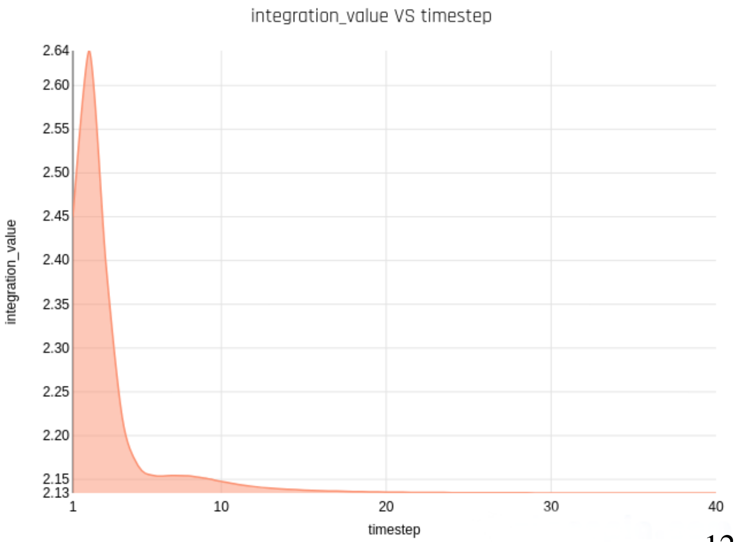
Test Result-1

f = 10*x*y

$N = N_x * N_y = 25$



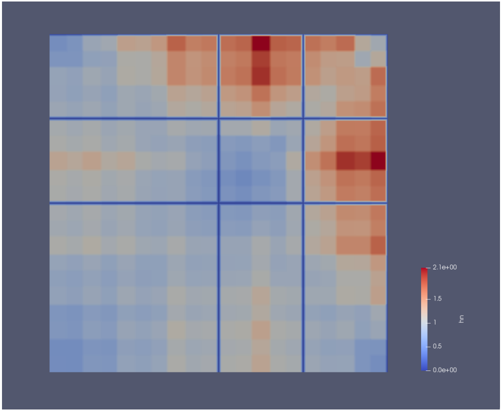
Convergence



Test Result-2

f = 100*x*y

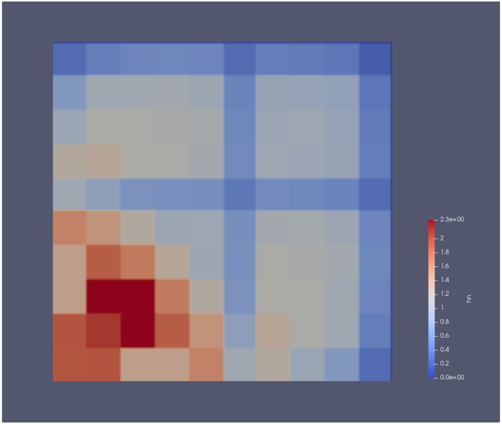
$N = N_x * N_y = 20 * 20 = 400$



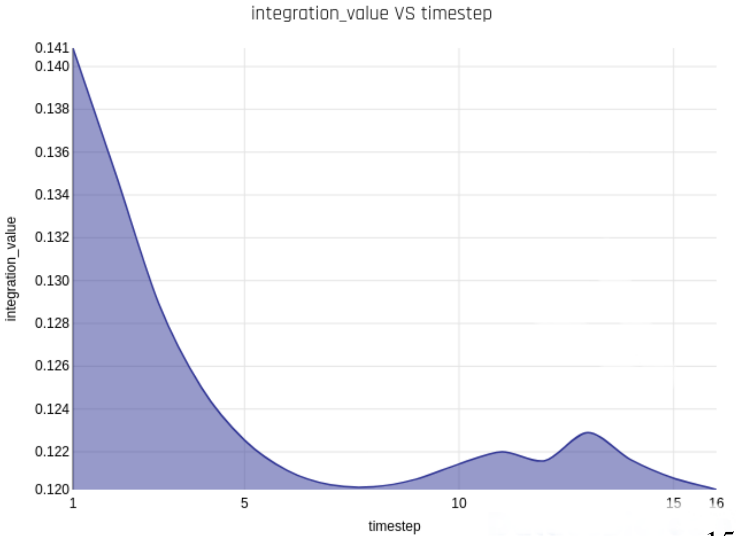
Test Result-3

$f = -(400x^2 - 20d)e^{-}10x^2$

$N = N_x * N_y = 10 * 10 = 100$



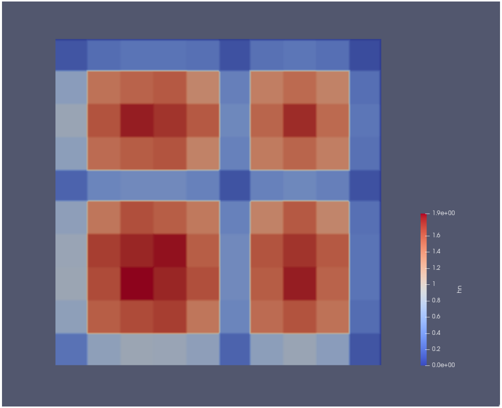
Convergence



Test Result-4

$$f = \sqrt{\frac{(x-0.5)^2+(y-0.5)^2}{4}}$$

$$N = N_x * N_y = 10 * 10 = 100$$



Test Result-5

$$f = e^{\frac{-(x-0.5)^2+(y-0.5)^2}{0.2}}$$

$$N = N_x * N_y = 10 * 10 = 100$$