Multiphysics shape optimization based on a level set mesh evolution framework

Florian Feppon

Grégoire Allaire, Charles Dapogny
Julien Cortial, Felipe Bordeu

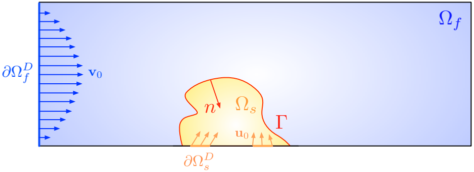
MMG Day – December 13, 2018

SAFRAN

l'X ÉCOLE POLYTECHNIQUE UNIVERSITÉ PARIS-SACLAY

Simplified weakly coupled three-physics setting

$$\min_{\Gamma} J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)).$$

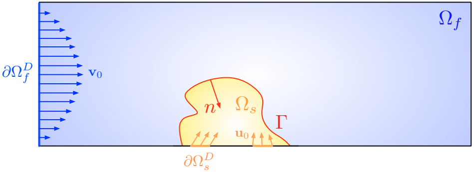


- Incompressible Navier-Stokes equations for $(\boldsymbol{v}, p)$ in Ω_f

$$-\mathrm{div}(\sigma_f(\boldsymbol{v}, p)) + \rho\nabla\boldsymbol{v}\,\boldsymbol{v} = \boldsymbol{f}_f \text{ in } \Omega_f$$

Simplified weakly coupled three-physics setting

$$\min_{\Gamma} J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)).$$



- Incompressible Navier-Stokes equations for $(\boldsymbol{v}, p)$ in Ω_f

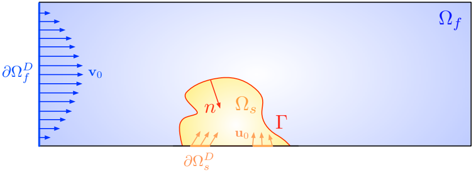
$$-\mathrm{div}(\sigma_f(\boldsymbol{v}, p)) + \rho \nabla \boldsymbol{v}\, \boldsymbol{v} = \boldsymbol{f}_f \text{ in } \Omega_f$$

- Steady-state convection-diffusion for T_f and T_s in Ω_f and Ω_s:

$$\begin{aligned} -\mathrm{div}(k_f \nabla T_f) + \rho \boldsymbol{v} \cdot \nabla T_f &= Q_f \quad \text{in } \Omega_f \\ -\mathrm{div}(k_s \nabla T_s) &= Q_s \quad \text{in } \Omega_s \end{aligned}$$

Simplified weakly coupled three-physics setting

$$\min_{\Gamma} J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)).$$



- Incompressible Navier-Stokes equations for $(\boldsymbol{v}, p)$ in Ω_f

$$-\mathrm{div}(\sigma_f(\boldsymbol{v}, p)) + \rho\nabla\boldsymbol{v}\,\boldsymbol{v} = \boldsymbol{f}_f \text{ in } \Omega_f$$

- Steady-state convection-diffusion for T_f and T_s in Ω_f and Ω_s:

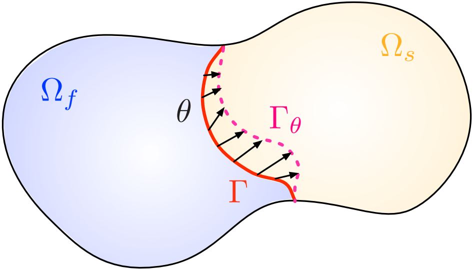
$$\begin{aligned} -\mathrm{div}(k_f\nabla T_f) + \rho\boldsymbol{v}\cdot\nabla T_f &= Q_f \quad \text{in } \Omega_f \\ -\mathrm{div}(k_s\nabla T_s) &= Q_s \quad \text{in } \Omega_s \end{aligned}$$

- Linearized thermoelasticity with fluid-structure interaction for $\boldsymbol{u}$ in Ω_s:

$$\begin{aligned} -\mathrm{div}(\sigma_s(\boldsymbol{u}, T_s)) &= \boldsymbol{f}_s && \text{in } \Omega_s \\ \sigma_s(\boldsymbol{u}, T_s)\cdot\boldsymbol{n} &= \sigma_f(\boldsymbol{v}, p)\cdot\boldsymbol{n} && \text{on } \Gamma. \end{aligned}$$

Hadamard's method of boundary variations

$$\min_{\Gamma} \; J(\Gamma)$$



$$\Gamma_{\boldsymbol{\theta}} = (I + \boldsymbol{\theta})\Gamma, \text{ where } \boldsymbol{\theta} \in W_0^{1,\infty}(\Omega, \mathbb{R}^d), \; ||\boldsymbol{\theta}||_{W^{1,\infty}(\mathbb{R}^d,\mathbb{R}^d)} < 1.$$

$$J(\Gamma_{\boldsymbol{\theta}}) = J(\Gamma) + \frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\theta}}(\boldsymbol{\theta}) + o(\boldsymbol{\theta}), \quad \text{where} \quad \frac{|o(\boldsymbol{\theta})|}{||\boldsymbol{\theta}||_{W^{1,\infty}(\Omega,\mathbb{R}^d)}} \xrightarrow{\boldsymbol{\theta}\to 0} 0,$$

Stakes

For industrial applications, we seek to solve

$$\begin{aligned} \min_{\Gamma} \quad & J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \\ \text{s.t.} \quad & g_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) = 0,\ 1 \leq i \leq p\cdot \\ & h_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \leq 0,\ 1 \leq i \leq q \end{aligned}$$

For reliability:

1. For compatibility with industrial solvers, the physics should not be altered.

Stakes

For industrial applications, we seek to solve

$$\begin{aligned} \min_{\Gamma} \quad & J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \\ \text{s.t.} \quad & g_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) = 0, \ 1 \le i \le p \cdot \\ & h_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \le 0, \ 1 \le i \le q \end{aligned}$$

For reliability:

1. For compatibility with industrial solvers, the physics should not be altered.
2. User should provide only minimal information: J, g_i, h_i and sensitivities

$$\frac{\partial J}{\partial \Gamma}, \dots, \frac{\partial J}{\partial \boldsymbol{u}}, \frac{\partial g_i}{\partial \Gamma}, \dots, \frac{\partial g_i}{\partial \boldsymbol{u}}, \frac{\partial h_i}{\partial \Gamma}, \dots, \frac{\partial h_i}{\partial \boldsymbol{u}} \text{ and so on.}$$

Stakes

For industrial applications, we seek to solve

$$\begin{aligned} \min_{\Gamma} \quad & J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \\ \text{s.t.} \quad & g_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) = 0,\ 1 \leq i \leq p \cdot \\ & h_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \leq 0,\ 1 \leq i \leq q \end{aligned}$$

For reliability:

1. For compatibility with industrial solvers, the physics should not be altered.
2. User should provide only minimal information: J, g_i, h_i and sensitivities

$$\frac{\partial J}{\partial \Gamma}, \dots, \frac{\partial J}{\partial \boldsymbol{u}}, \frac{\partial g_i}{\partial \Gamma}, \dots, \frac{\partial g_i}{\partial \boldsymbol{u}}, \frac{\partial h_i}{\partial \Gamma}, \dots, \frac{\partial h_i}{\partial \boldsymbol{u}} \text{ and so on.}$$

3. Optimization should handle unfeasible initializations Γ

Stakes

For industrial applications, we seek to solve

$$\begin{aligned} \min_{\Gamma} \quad & J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \\ \text{s.t.} \quad & g_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) = 0,\ 1 \leq i \leq p\cdot \\ & h_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \leq 0,\ 1 \leq i \leq q \end{aligned}$$

For reliability:

1. For compatibility with industrial solvers, the physics should not be altered.
2. User should provide only minimal information: J, g_i, h_i and sensitivities

$$\frac{\partial J}{\partial \Gamma}, \dots, \frac{\partial J}{\partial \boldsymbol{u}}, \frac{\partial g_i}{\partial \Gamma}, \dots, \frac{\partial g_i}{\partial \boldsymbol{u}}, \frac{\partial h_i}{\partial \Gamma}, \dots, \frac{\partial h_i}{\partial \boldsymbol{u}} \text{ and so on.}$$

3. Optimization should handle unfeasible initializations Γ
4. No fine tuning of optimization algorithm parameters should be required

Stakes

For industrial applications, we seek to solve

$$\begin{aligned} \min_{\Gamma} \quad & J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \\ \text{s.t.} \quad & g_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) = 0, \; 1 \leq i \leq p \cdot \\ & h_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \leq 0, \; 1 \leq i \leq q \end{aligned}$$

Our method:

1. The interface Γ is remeshed at every iteration for solving original state equations on each subdomain

Stakes

For industrial applications, we seek to solve

$$\begin{aligned} \min_{\Gamma} \quad & J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \\ \text{s.t.} \quad & g_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) = 0,\ 1 \leq i \leq p \cdot \\ & h_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \leq 0,\ 1 \leq i \leq q \end{aligned}$$

Our method:

1. The interface Γ is remeshed at every iteration for solving original state equations on each subdomain
2. We implement a *single* analytical formula for

$$\frac{\mathrm{d}}{\mathrm{d}\Gamma}\left[J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)\right]$$

for an arbitrary J based on $\dfrac{\partial J}{\partial \Gamma}, \ldots \dfrac{\partial J}{\partial \boldsymbol{u}}$.

Stakes

For industrial applications, we seek to solve

$$\begin{aligned} \min_{\Gamma} \quad & J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \\ \text{s.t.} \quad & g_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) = 0,\, 1 \leq i \leq p\cdot \\ & h_i(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)) \leq 0,\, 1 \leq i \leq q \end{aligned}$$

Our method:

1. The interface Γ is remeshed at every iteration for solving original state equations on each subdomain
2. We implement a *single* analytical formula for

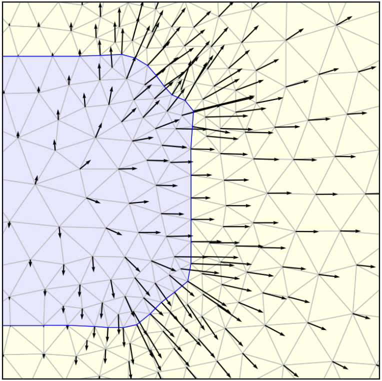
$$\frac{\mathrm{d}}{\mathrm{d}\Gamma}\left[J(\Gamma, \boldsymbol{v}(\Gamma), p(\Gamma), T(\Gamma), \boldsymbol{u}(\Gamma)\right]$$

   for an arbitrary J based on $\dfrac{\partial J}{\partial \Gamma}, \dots \dfrac{\partial J}{\partial \boldsymbol{u}}$.
3. We designed our own constrained optimization algorithm

1. Level set based mesh evolution method

We consider the algorithm proposed by Allaire, Dapogny, Frey (2013):

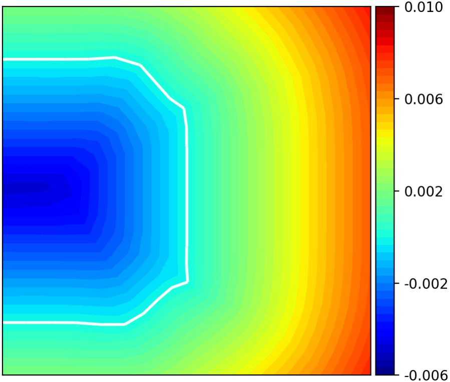
1. Given a mesh and a moving vector field



1. Level set based mesh evolution method

We consider the algorithm proposed by Allaire, Dapogny, Frey (2013):

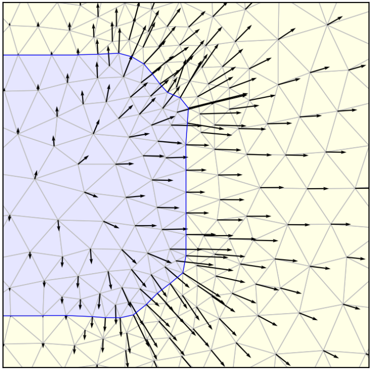
2. A level-set function ϕ associated to $\Omega = \Omega_s \cup \Omega_f$ is computed on the mesh.



1. Level set based mesh evolution method

We consider the algorithm proposed by Allaire, Dapogny, Frey (2013):

3. The level-set function is avected on the computational domain which is then adaptively remeshed:



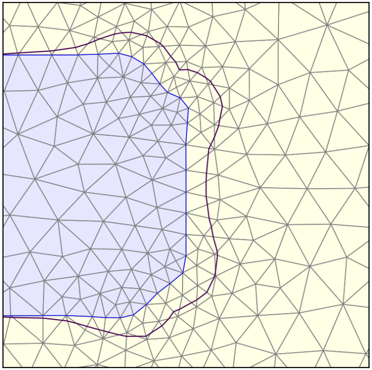
1. Level set based mesh evolution method

We consider the algorithm proposed by Allaire, Dapogny, Frey (2013):

3. The level-set function is avected on the computational domain which is then adaptively remeshed:

Advection of a level set for Ω on the computational mesh.

$$\partial_t \phi + \boldsymbol{\theta} \cdot \nabla \phi = 0$$

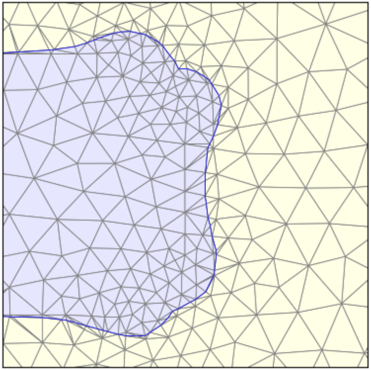


1. Level set based mesh evolution method

We consider the algorithm proposed by Allaire, Dapogny, Frey (2013):

3. The level-set function is avected on the computational domain which is then adaptively remeshed:

Breaking the zero isoline of the level set.

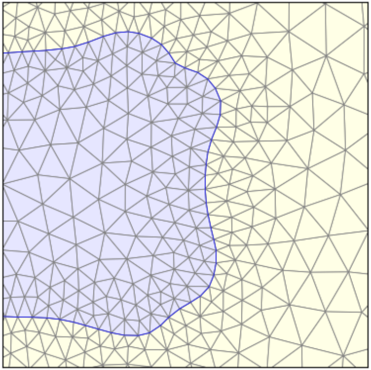


1. Level set based mesh evolution method

We consider the algorithm proposed by Allaire, Dapogny, Frey (2013):

3. The level-set function is avected on the computational domain which is then adaptively remeshed:

Remeshing adaptively the computational mesh.



2. User provides minimal information

For instance, for drag minimization

$$J(\Gamma, \boldsymbol{v}(\Gamma)) = \int_{\Omega_f} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v}) \mathrm{d}x$$

with $e(\boldsymbol{v}) = (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T)/2$.

2. User provides minimal information

For instance, for drag minimization

$$J(\Gamma, \boldsymbol{v}(\Gamma)) = \int_{\Omega_f} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v}) \mathrm{d}x$$

with $e(\boldsymbol{v}) = (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T)/2$.
User will provide only

$$\frac{\partial J}{\partial \Gamma} \cdot \boldsymbol{\theta} = \int_{\Gamma} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v})\, \boldsymbol{\theta} \cdot \boldsymbol{n} \mathrm{d}s$$

2. User provides minimal information

For instance, for drag minimization

$$J(\Gamma, \boldsymbol{v}(\Gamma)) = \int_{\Omega_f} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v}) \mathrm{d}x$$

with $e(\boldsymbol{v}) = (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T)/2$.
User will provide only

$$\frac{\partial J}{\partial \Gamma} \cdot \boldsymbol{\theta} = \int_{\Gamma} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v})\, \boldsymbol{\theta} \cdot \boldsymbol{n} \mathrm{d}s$$

$$\frac{\partial J}{\partial \boldsymbol{v}} \cdot \boldsymbol{w} = \int_{\Omega_f} 4\nu e(\boldsymbol{v}) : e(\boldsymbol{w}) \mathrm{d}x$$

2. User provides minimal information

For instance, for drag minimization

$$J(\Gamma, \boldsymbol{v}(\Gamma)) = \int_{\Omega_f} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v})\mathrm{d}x$$

with $e(\boldsymbol{v}) = (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T)/2$.
User will provide only

$$\frac{\partial J}{\partial \Gamma} \cdot \boldsymbol{\theta} = \int_{\Gamma} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v})\, \boldsymbol{\theta} \cdot \boldsymbol{n} \mathrm{d}s$$

$$\frac{\partial J}{\partial \boldsymbol{v}} \cdot \boldsymbol{w} = \int_{\Omega_f} 4\nu e(\boldsymbol{v}) : e(\boldsymbol{w})\mathrm{d}x$$

Then the value of $\frac{\mathrm{d}}{\mathrm{d}\Gamma}[J(\Gamma, \boldsymbol{v}(\Gamma))]$ is computed analytically and automatically by solving adjoint states.

2. User provides minimal information

$$
\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\Big[J(\Gamma_{\boldsymbol{\theta}}, \boldsymbol{v}(\Gamma_{\boldsymbol{\theta}}), p(\Gamma_{\boldsymbol{\theta}}), T(\Gamma_{\boldsymbol{\theta}}), \boldsymbol{u}(\Gamma_{\boldsymbol{\theta}}))\Big](\boldsymbol{\theta}) \\
&\quad = \overline{\frac{\partial \mathfrak{J}}{\partial \boldsymbol{\theta}}}(\boldsymbol{\theta}) + \int_\Gamma (\boldsymbol{f}_f \cdot \boldsymbol{w} - \sigma_f(\boldsymbol{v}, p) : \nabla \boldsymbol{w} + \boldsymbol{n} \cdot \sigma_f(\boldsymbol{w}, q) \nabla \boldsymbol{v} \cdot \boldsymbol{n} + \boldsymbol{n} \cdot \sigma_f(\boldsymbol{v}, p) \nabla \boldsymbol{w} \cdot \boldsymbol{n})(\boldsymbol{\theta} \cdot \boldsymbol{n}) \mathrm{d}s \\
&\quad + \int_\Gamma \left(k_s \nabla T_s \cdot \nabla S_s - k_f \nabla T_f \cdot \nabla S_f + Q_f S_f - Q_s S_s - 2k_s \frac{\partial T_s}{\partial n} \frac{\partial S_s}{\partial n} + 2k_f \frac{\partial T_f}{\partial n} \frac{\partial S_f}{\partial n} \right) (\boldsymbol{\theta} \cdot \boldsymbol{n}) \mathrm{d}s \\
&\quad + \int_\Gamma (\sigma_s(\boldsymbol{u}, T_s) : \nabla \boldsymbol{r} - \boldsymbol{f}_s \cdot \boldsymbol{r} - \boldsymbol{n} \cdot Ae(\boldsymbol{r}) \nabla \boldsymbol{u} \cdot \boldsymbol{n} - \boldsymbol{n} \cdot \sigma_s(\boldsymbol{u}, T_s) \nabla \boldsymbol{r} \cdot \boldsymbol{n}) (\boldsymbol{\theta} \cdot \boldsymbol{n}) \mathrm{d}s
\end{aligned}
$$

2. User provides minimal information

$$\int_{\Omega_s} Ae(\boldsymbol{r}) : \nabla \boldsymbol{r}' \mathrm{d}\boldsymbol{x} = \frac{\partial \mathfrak{J}}{\partial \hat{\boldsymbol{u}}}(\boldsymbol{r}') \quad \forall \boldsymbol{r}' \in V_{\boldsymbol{u}}(\Gamma)\,.$$

2. User provides minimal information

$$\int_{\Omega_s} Ae(\boldsymbol{r}) : \nabla \boldsymbol{r}' \mathrm{d}\boldsymbol{x} = \frac{\partial \mathfrak{J}}{\partial \hat{\boldsymbol{u}}}(\boldsymbol{r}') \quad \forall \boldsymbol{r}' \in V_{\boldsymbol{u}}(\Gamma)\,.$$

$$\downarrow$$

$$\int_{\Omega_s} k_s \nabla S \cdot \nabla S' \mathrm{d}\boldsymbol{x} + \int_{\Omega_f} (k_f \nabla S \cdot \nabla S' + \rho c_p S \boldsymbol{v} \cdot \nabla S') \mathrm{d}\boldsymbol{x} = \int_{\Omega_s} \alpha \mathrm{div}(\boldsymbol{r}) S' \mathrm{d}\boldsymbol{x} + \frac{\partial \mathfrak{J}}{\partial \hat{T}}(S) \quad \forall S' \in V_T(\Gamma)\,.$$

2. User provides minimal information

$$\int_{\Omega_s} Ae(\boldsymbol{r}) : \nabla \boldsymbol{r}' \mathrm{d}\boldsymbol{x} = \frac{\partial \mathfrak{J}}{\partial \hat{\boldsymbol{u}}}(\boldsymbol{r}') \quad \forall \boldsymbol{r}' \in V_{\boldsymbol{u}}(\Gamma)\,.$$

$$\downarrow$$

$$\int_{\Omega_s} k_s \nabla S \cdot \nabla S' \mathrm{d}\boldsymbol{x} + \int_{\Omega_f} (k_f \nabla S \cdot \nabla S' + \rho c_p S \boldsymbol{v} \cdot \nabla S') \mathrm{d}\boldsymbol{x} = \int_{\Omega_s} \alpha \mathrm{div}(\boldsymbol{r}) S' \mathrm{d}\boldsymbol{x} + \frac{\partial \mathfrak{J}}{\partial \hat{T}}(S) \quad \forall S' \in V_T(\Gamma)\,.$$

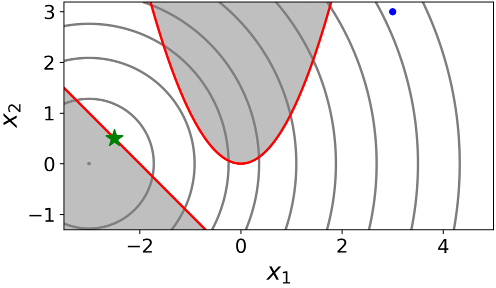
$$\downarrow$$

$$\begin{aligned} &\boldsymbol{w} = \boldsymbol{r} \text{ on } \Gamma \text{ and } \forall (\boldsymbol{w}', q') \in V_{\boldsymbol{v},p}(\Gamma) \\ &\quad \int_{\Omega_f} \Big(\sigma_f(\boldsymbol{w}, q) : \nabla \boldsymbol{w}' + \rho \boldsymbol{w} \cdot \nabla \boldsymbol{w}' \cdot \boldsymbol{v} + \rho \boldsymbol{w} \cdot \nabla \boldsymbol{v} \cdot \boldsymbol{w}' - q' \mathrm{div}(\boldsymbol{w}) \Big) \mathrm{d}\boldsymbol{x} = \\ &\qquad\qquad \int_{\Omega_f} -\rho c_p S \nabla T \cdot \boldsymbol{w}' \mathrm{d}\boldsymbol{x} + \frac{\partial \mathfrak{J}}{\partial (\boldsymbol{v}', p')}(\boldsymbol{w}', q'), \end{aligned}$$

3. A generic optimization algorithm

$$\min_{(x_1,x_2)\in\mathbb{R}^2} \quad J(x_1,x_2) = x_1^2 + (x_2+3)^2$$

$$\text{s.t.} \quad \begin{cases} h_1(x_1,x_2) = -x_1^2 + x_2 & \leq 0 \\ h_2(x_1,x_2) = -x_1 - x_2 - 2 & \leq 0 \end{cases}$$



3. A generic optimization algorithm

All in all, we solve an ODE of the form

$$\dot{x} = -\alpha_J \xi_J(x) - \alpha_C \xi_C(x)$$

- $-\xi_J(x)$ is the best descent direction $-\xi_J(x)$ tangent to the constraints:
  1. If no constraint are saturated, $\xi_J(x) = \nabla J(x)$

3. A generic optimization algorithm

All in all, we solve an ODE of the form

$$\dot{x} = -\alpha_J \xi_J(x) - \alpha_C \xi_C(x)$$

- $-\xi_J(x)$ is the best descent direction $-\xi_J(x)$ tangent to the constraints:
  1. If no constraint are saturated, $\xi_J(x) = \nabla J(x)$
  2. If not, $\xi_J(x) = -\Pi_{C_{I(x)}}(\nabla J(x))$ where Π_{C_I} is the linear tangent projection on

     $$\{\xi \in \mathbb{R}^n,\, \mathrm{D}h_i(x)\xi = 0 \text{ for } i \in I\}$$

     for I a relevant subset of the constraints.

3. A generic optimization algorithm

All in all, we solve an ODE of the form

$$\dot{x} = -\alpha_J \xi_J(x) - \alpha_C \xi_C(x)$$

- $-\xi_J(x)$ is the best descent direction $-\xi_J(x)$ tangent to the constraints:
    1. If no constraint are saturated, $\xi_J(x) = \nabla J(x)$
    2. If not, $\xi_J(x) = -\Pi_{C_{I(x)}}(\nabla J(x))$ where Π_{C_I} is the linear tangent projection on

       $$\{\xi \in \mathbb{R}^n,\, \mathrm{D}h_i(x)\xi = 0 \text{ for } i \in I\}$$

       for I a relevant subset of the constraints.
    3. The subset I can determined by solving some dual quadratic optimization subproblem.

3. A generic optimization algorithm

All in all, we solve an ODE of the form

$$\dot{x} = -\alpha_J \xi_J(x) - \alpha_C \xi_C(x)$$

- $-\xi_J(x)$ is the best descent direction $-\xi_J(x)$ tangent to the constraints:
  1. If no constraint are saturated, $\xi_J(x) = \nabla J(x)$
  2. If not, $\xi_J(x) = -\Pi_{C_{I(x)}}(\nabla J(x))$ where Π_{C_I} is the linear tangent projection on

     $$\{\xi \in \mathbb{R}^n,\, \mathrm{D}h_i(x)\xi = 0 \text{ for } i \in I\}$$

     for I a relevant subset of the constraints.
  3. The subset I can determined by solving some dual quadratic optimization subproblem.

3. A generic optimization algorithm

All in all, we solve an ODE of the form

$$\dot{x} = -\alpha_J \xi_J(x) - \alpha_C \xi_C(x)$$

- $-\xi_J(x)$ is the best descent direction $-\xi_J(x)$ tangent to the constraints:
    1. If no constraint are saturated, $\xi_J(x) = \nabla J(x)$
    2. If not, $\xi_J(x) = -\Pi_{C_{I(x)}}(\nabla J(x))$ where Π_{C_I} is the linear tangent projection on

       $$\{\xi \in \mathbb{R}^n, \, \mathrm{D}h_i(x)\xi = 0 \text{ for } i \in I\}$$

       for I a relevant subset of the constraints.
    3. The subset I can determined by solving some dual quadratic optimization subproblem.
- $-\xi_C(x)$ is a Gauss-Newton direction moving the trajectory back onto the feasible set:

$$\mathrm{D}h_i(-\xi_C(x)) = -h_i(x).$$

Practical implementation

- State and adjoint PDE equations are solved with FreeFem++

Practical implementation

- State and adjoint PDE equations are solved with FreeFem++
- The optimization loop runs in python

Practical implementation

- State and adjoint PDE equations are solved with FreeFem++
- The optimization loop runs in python
- Remeshing is performed by mmg2d (`-ls` option)

Demonstrations on shape optimization test cases

Volume minimization subject to rigidity constraint

$$\begin{aligned} &\min_{\Omega} \int_{\Omega_s} \mathrm{d}x \\ &s.t. \int_{\Omega_s} Ae(\boldsymbol{u}) : e(\boldsymbol{u}) \mathrm{d}x \leq C \end{aligned}$$

Demonstrations on shape optimization test cases

Volume minimization subject to multiple load rigidity constraints

$$\min_{\Omega} \quad \int_{\Omega_s} \mathrm{d}x$$
$$s.t. \int_{\Omega_s} Ae(\boldsymbol{u}_i) : e(\boldsymbol{u}_i)\mathrm{d}x \leq C_i, \quad \forall i = 1 \ldots 9$$

Demonstrations on shape optimization test cases

Lift maximisation subject to drag, volume and center of mass constraint:

$$\max_{\Omega} \quad \int_{\partial\Omega_f} \boldsymbol{e}_y \cdot \sigma_f(\boldsymbol{v}, p) \cdot \boldsymbol{n} \mathrm{d}x$$

$$s.t. \begin{cases} \displaystyle\int_{\Omega_f} 2\nu e(\boldsymbol{v}) : e(\boldsymbol{v}) \mathrm{d}x \leq C_{drag}, \\ \displaystyle\int_{\Omega_f} \mathrm{d}x = C_{vol}, \\ \displaystyle\int_{\Omega_f} \boldsymbol{x} \mathrm{d}x = 0 \end{cases}$$

Demonstrations on shape optimization test cases

Heat transfer subject to maximal pressure drop, volume and center of mass constraint:

$$\max_{\Omega} \quad \int_{\partial\Omega_{f,out}} \rho c_p T \boldsymbol{v} \cdot \boldsymbol{n} Ds - \int_{\partial\Omega_{f,in}} \rho c_p T \boldsymbol{v} \cdot \boldsymbol{n} Ds$$

$$s.t. \int_{\partial\Omega_{f,out}} p\mathrm{d}s - \int_{\partial\Omega_{f,in}} p\mathrm{d}s \leq \mathrm{DP}_0$$

Demonstrations on shape optimization test cases

Heat exchange subject to maximal pressure drop and non penetration constraint:

$$
\begin{aligned}
\max_{\Omega} \quad & \int_{\Omega_{f,cold}} \rho c_p \boldsymbol{v} \cdot \nabla T \mathrm{d}x - \int_{\Omega_{f,hot}} \rho c_p \boldsymbol{v} \cdot \nabla T \mathrm{d}x \\
s.t. \quad & \int_{\partial\Omega_{f,out}} p \mathrm{d}s - \int_{\partial\Omega_{f,in}} p \mathrm{d}s \leq \mathrm{DP}_0, \\
& d(\Omega_{f,hot}, \Omega_{f,cold}) \geqslant d_{\min}
\end{aligned}
$$

Demonstrations on shape optimization test cases

...

References

Feppon, F., Allaire, G., Bordeu, F., Cortial, J., and Dapogny, C. Shape optimization of a coupled thermal fluid-structure problem in a level set mesh evolution framework.
HAL preprint hal-01686770 (2018).

Feppon, F., Allaire, G., and Dapogny, C. Null space gradient flows for constrained optimization with applications to shape optimization.
In preparation (2018).