Shape optimization: software and equations

Tozoni, Davi Colli
`davi.tozoni@nyu.edu`

January 25, 2018

1 Software

1.1 Pipeline

1. Read `.wire` file and populate graph structures
2. At each optimization step:
   (a) Inflate mesh with current truss parameters
   (b) Compute the shape velocity of the boundary vertices (in `PostProcess` + `Inflators`)
   (c) Solve PDE of general linear elasticity problem, finding displacement (done in similar class to `in PatternOptimizationIterate`)
   (d) Evaluate objective function (in similar class to `WCSObjectiveTerm`)):
      - Compute function $s(\epsilon)$ corresponding to stress (e.g. $\sigma : \sigma$)
      - Compute cost function for each point x: $j(s) = s^{p/2}$
      - Integrate $j(s)$ through entire function to obtain $J = \int_{\Omega} j(s(\epsilon))\, d\Omega$
   (e) Evaluate shape derivative (in class similar to `WorstCaseStress`):
      - Solve adjoint cell PDE problem to find ρ (used to compute $dJ[v]$)
      - Build discrete volume differential form $dJ[v]$
   (f) Extend boundary shape velocities into intern nodes, in order to use $dJ[v]$ (done in `ObjectiveTerm`)
   (g) Compute gradient of J w.r.t. shape parameters
3. Output best solution found

1.2 Tasks based on WCS Optimization analysis

1. Create class `NonPeriodicCellOps` similar to baseCellOps to solve non periodic general PDEs
2. Create `MicroscopicStress` class, similar to `WorstCaseStress` for computing dJ and other auxiliar functions (to compute stress in our new scenario)
3. Create `MicroscopicStressObjectiveTerm`, in a similar way to `WCSObjectiveTerm`
4. Modify `PatternOptimizationIterate` or create similar class where general PDE is solved at each iteration, (by calling `NonPeriodicCellOps`)

The interaction works as following: class `MicroscopicStressObjectiveTerm` uses `NonPeriodicCellOps` of `PatternOptimizationIterate` (after solving PDE) to evaluate objective term and compute differential. The iterate class (4) then should be used by the `IterateManager` class inside the solvers.

2 Our problem

In this project, the objective is to solve a general shape optimization problem, given fixed Dirichlet and Neumann boundary conditions, as represented in the picture below:

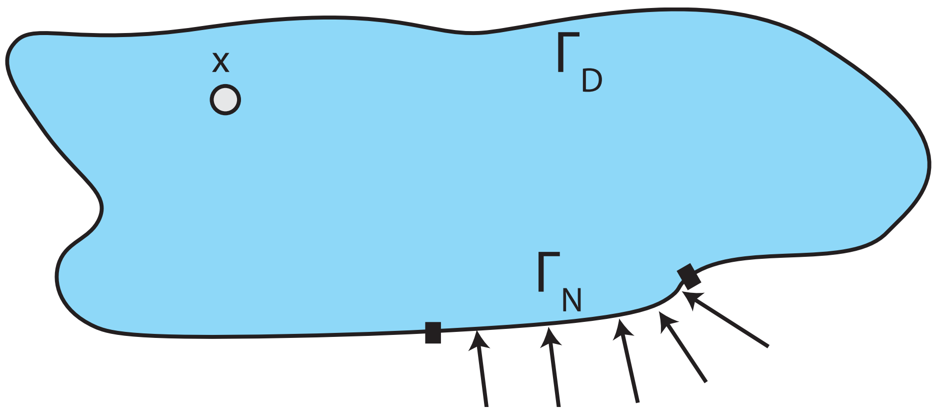

Figure 1: Our problem.

The corresponding PDE then is the following (strong form):

$$\begin{aligned} -\nabla \cdot \sigma = 0 \text{ (external force)} \\ \text{such that:} \\ u = \hat{u} \quad \text{on } \Gamma_D \\ \sigma n = \hat{T} \quad \text{on } \Gamma_N \end{aligned}$$

, where u is the unknown displacement, $\epsilon = 1/2(\nabla u + (\nabla u)^T)$ is the strain and $\sigma = C\epsilon$ is the stress.

The corresponding weak form is the following:

$$\int_\Omega \epsilon(\phi) : C : \epsilon(u)\, dw = \int_{\Gamma_N} \phi \cdot \hat{T}\, d\Gamma_N \tag{1}$$

, for all ϕ where $\phi(y) = 0 \quad \forall\, y \in \Gamma_D$. The corresponding weak form for the periodic case can be found in equation (A1) in [1].

Now, consider we are minimizing a cost function $\int_\Omega j(s(u))\, dx$, where s corresponds to some measure of stress, and $j(s) = s^{p/2}$. In summary, we are optimizing the L^p norm of the stress in the whole object.

In the remaining of this text, I suppose $s(u) = \|\sigma\|_F^2 = \sigma^T : \sigma = \sigma : \sigma$. Notice that operation $A : B$ equals $A_{ij}B_{ij}$ in index notation (for A and B being 2-order tensors).

The computation of the cost itself is not difficult and can be done directly through a quadrature rule after solving our original elasticity PDE. However, when optimizing the shape, we need to compute how cost changes when deforming the Ω. This means computing the ***shape derivative*** of J.

So here are the steps:

1. Find equation of $dJ[v]$
2. Compute τ, which corresponds to the derivative of j with respect to strain ϵ
3. Compute derivative of our PDE weak form
4. Find adjoint PDE for computing continuous version of $dJ[v]$
5. Compute discrete differential form for $dJ[v]$ (the one that is actually implemented)

2.1 Find Equation for $dJ[v]$

Consider the mapping $f_t(X) = X + tv(X)$ represented in the Figure 2, where v is the velocity. (The inverse of f_t is then $f_t^{-1}(x) = x - tv$).

The jacobian of the map is then ∇f_t and equals $F_t = I + t\nabla v$. (As we know, the jacobian of f_t^{-1} corresponds to F_t^{-1}).

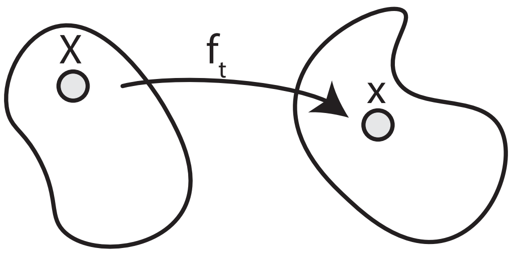

Figure 2: Mapping between initial and deformed object after time t.

We can compute the displacement u of a point x (after time t) as $u_t(x)$, which equals function $\hat{u}(X)$ with domain Ω_0:

$$u_t(x) = \hat{u}(X) = \hat{u}(x - tv)$$

As a consequence, we also have that:

$$\epsilon(u_t(x)) = \epsilon(\hat{u}(x - tv)) = \epsilon(\hat{u}(f_t^{-1}(x))) = \epsilon(\hat{u}(X))$$

Applying this to our cost J, we can obtain a formula entirely on Ω_0, facilitating derivatives w.r.t. time:

$$J = \int_{\Omega(t)} j(s(\epsilon(u_t(x))))dx = \int_{\Omega_0} j(s(\epsilon(\hat{u}(X)))) \det(\nabla f_t) dX$$

Deriving J w.r.t t:

$$\begin{aligned} \frac{dJ}{dt} &= \int_{\Omega_0} \frac{d}{dt} \left(j(s(\epsilon(\hat{u}(X)))) \det(\nabla f_t) \right) dX \\ &= \int_{\Omega_0} \tau : D[\epsilon] \det(F_t) + j \det(F_t) \operatorname{Tr}(F_t^{-1} \nabla v) dX \end{aligned}$$

, where $D[\cdot]$ is the material derivative of $\cdot$ and $\tau = \frac{\partial j}{\partial s}\frac{\partial s}{\partial \epsilon}$.

We can then compute the shape derivative by evaluating the derivative at $t = 0$. Notice that $F_0 = I$ and that $\mathrm{Tr}(\nabla v) = \nabla \cdot v$.

$$dJ[v] = \frac{dJ}{dt}\Big|_{t=0} = \int_{\Omega_0} \tau : D[\epsilon] + j\nabla \cdot v \, dX$$

But $D[\epsilon(u)]$ can be written as $\epsilon(D[u]) - \mathrm{sym}(\nabla u \nabla v)$, where $\mathrm{sym}[\cdot]$ means the symmetric part of $\cdot$. Plugging this, we obtain the equation below:

$$\boxed{dJ[v] = \int_{\Omega_0} \underbrace{\tau : \epsilon(D[u])}_{III} - \tau : \mathrm{sym}(\nabla u \nabla v) + j\nabla \cdot v \, dX} \tag{2}$$

, which corresponds to formula (A17) in the supplementary material of [1].

As expected, term III is the difficult one to compute, since u is the output of a PDE.

2.2 Compute τ

Let's compute τ:

$$\tau = j' \frac{\partial s}{\partial \epsilon}$$

It is easier to compute for each entry in τ the following way:

$$\begin{aligned}\tau_{ij} &= j' \frac{\partial s}{\partial \epsilon_{ij}} = j' \frac{\partial s}{\partial \epsilon_{ij}}(\sigma : \sigma) = j' \left(\frac{\partial \sigma}{\partial \epsilon_{ij}} : \sigma + \sigma : \frac{\partial \sigma}{\partial \epsilon_{ij}} \right) \\ &= 2j' \frac{\partial \sigma}{\partial \epsilon_{ij}} : \sigma = 2j' C^{\text{base}} : \frac{\partial \epsilon}{\partial \epsilon_{ij}} : \sigma\end{aligned}$$

But $\frac{\partial \epsilon_{kl}}{\partial \epsilon_{ij}} = \delta_{ki}\delta_{lj}$ and then:

$$\frac{\partial (C_{abcd}\epsilon_{cd})}{\partial \epsilon_{ij}} = C_{abij}$$

Consequently, $\tau_{ij} = 2j'(\sigma_{ab}C_{abij})$ and

$$\boxed{\tau = 2j'\sigma : C} \tag{3}$$

, which corresponds to equation (A10) in [1].

2.3 Compute Derivative of PDE

In order to compute term III, we need to compute the derivative of the weak form in Equation 1, as shown here.

We start with the weak form:

$$\int_{\Omega} \epsilon(\phi) : C : \epsilon(u) \, dX = \int_{\Gamma_N} \phi \cdot \hat{T} \, d\Gamma_N$$

Differentiating on both sides (and obtaining the value at $t = 0$):

$$\underbrace{\frac{d}{dt}\Big|_{t=0} \left(\int_{\Omega} \epsilon(\phi) : C : \epsilon(u) \, dX \right)}_{\text{lhs}} = \underbrace{\frac{d}{dt}\Big|_{t=0} \left(\int_{\Gamma_N} \phi \cdot \hat{T} \, d\Gamma_N \right)}_{\text{rhs}}$$

2.3.1 Computing left hand side:

Similar math to derivation of dJ:

$$u(x) = \hat{u}(x - tv) \text{ and } \phi(x) = \hat{\phi}(x - tv)$$

$$\text{lhs} = \int_{\Omega_0} \frac{d}{dt}\Big|_{t=0} \left(\epsilon(\hat{\phi}) : C : \epsilon(\hat{u})\right) \det(F_0) + \epsilon(\hat{\phi}) : C : \epsilon(\hat{u}) \frac{d}{dt}\Big|_{t=0} (\det(F_t))\, dX$$

Applying product rule on first term and using that $\frac{dA_t}{dt} = det(A_t)\,\mathrm{Tr}(A_t^{-1}\frac{\partial A_t}{\partial t})$, we obtain:

$$\text{lhs} = \int_{\Omega_0} \left(D[\epsilon(\hat{\phi})] : C : \epsilon(\hat{u}) + \epsilon(\hat{\phi}) : C : D[\epsilon(\hat{u})] + \epsilon(\hat{\phi}) : C : \epsilon(\hat{u}) \nabla \cdot v\right) dX$$

Because $D[\epsilon(w)] = \epsilon(D[w]) - \mathrm{sym}(\nabla w \nabla v)$, we have:

$$\begin{aligned}\text{lhs} &= \int_{\Omega_0} \left((\epsilon(D[\phi]) - \mathrm{sym}(\nabla\hat{\phi}\nabla v)) : \underbrace{C : \epsilon(\hat{u})}_{\sigma} + \epsilon(\hat{\phi}) : C : (\epsilon(D[u]) - \mathrm{sym}(\nabla\hat{u}\nabla v)) + \epsilon(\hat{\phi}) : \underbrace{C : \epsilon(\hat{u})}_{\sigma} \nabla \cdot v \right) dX \\ &= \int_{\Omega_0} \left((\epsilon(D[\phi]) - \mathrm{sym}(\nabla\hat{\phi}\nabla v)) : \sigma + \epsilon(\hat{\phi}) : C : (\epsilon(D[u]) - \mathrm{sym}(\nabla\hat{u}\nabla v)) + \epsilon(\hat{\phi}) : \sigma \nabla \cdot v \right) dX\end{aligned}$$

In our case, since ϕ are functions of the barycentric coordinate functions in our FEM, $D[\phi] = 0$. In other words, $\phi(x) = \sum_i \phi_i(x)$ depends only on x and not on t. But x depends on t, doesn't it? The material point x doesn't't! Actually, ϕ depends on how x is positioned related to its neighbors, which does not change after time t, since x continue being a weight average of its face vertices (with same weights always). Would any of this change if we were using a non linear basis function for the FEM?.

Finally,

$$\text{lhs} = \int_{\Omega_0} \left(-\mathrm{sym}(\nabla\hat{\phi}\nabla v)) : \sigma + \epsilon(\hat{\phi}) : C : (\epsilon(D[\hat{u}]) - \mathrm{sym}(\nabla\hat{u}\nabla v)) + \epsilon(\hat{\phi}) : \sigma \nabla \cdot v\right) dX$$

2.3.2 Computing right hand side:

We know:

$$\hat{T}_t(x) = \hat{T}(x - tv) \text{ and } \phi(x) = \hat{\phi}(x - tv)$$

And we aim to compute the following integral

$$\text{rhs} = \frac{d}{dt}\Big|_{t=0} \left(\int_{\Gamma_N} \phi \cdot \hat{T}_t \, d\Gamma_N\right)$$

Then

$$\begin{aligned}\text{rhs} &= \int_{\Gamma_{N_0}} \frac{d}{dt}\Big|_{t=0} \left(\hat{\phi} \cdot \hat{T} \det(F_t)\right) d\Gamma_{N_0} \\ &= \int_{\Gamma_{N_0}} (\underbrace{D[\phi]}_{\mathbf{0}\text{ (FEM)}} \cdot \hat{T} + \phi \cdot \underbrace{D[\hat{T}]}_{\mathbf{0}\text{ (b.c.)}}) \det(F_0) + \hat{\phi} \cdot \hat{T} \frac{d}{dt}\Big|_{t=0} (\det(F_t))\, d\Gamma \\ &= \int_{\Gamma_{N_0}} \hat{\phi} \cdot \hat{T} \quad \nabla \cdot v \, d\Gamma_{N_0}\end{aligned}$$

2.3.3 Combining results

At $t = 0$, $\hat{u} = u$ and $\hat{\phi} = \phi$. Then, combining the left and right hand sides:

$$\int_{\Omega_0} (-\operatorname{sym}(\nabla\phi\nabla v)) : \sigma + \epsilon(\phi) : C : (\epsilon(D[u]) - \operatorname{sym}(\nabla u \nabla v)) + \epsilon(\phi) : \sigma \nabla \cdot v)\, dX = \int_{\Gamma_{N_0}} \phi \cdot \hat{T} \quad \nabla \cdot v \, d\Gamma_{N_0}$$

And

$$\boxed{\int_{\Omega} \epsilon(\phi) : C : \epsilon(D[u]) dX = \int_{\Omega} \operatorname{sym}(\nabla\phi\nabla v) : \sigma + \epsilon(\phi) : C : \operatorname{sym}(\nabla u \nabla v) - \epsilon(\phi) : \sigma \nabla \cdot v dX + \int_{\Gamma_N} \phi \cdot \hat{T} \quad \nabla \cdot v \, d\Gamma_N} \tag{4}$$

You can think this equation a weak form of another PDE on $D[u]$. However, we would have to solve a different PDE for each v. This corresponds to formula A(19) in [1].

2.4 Adjoint PDE Problem

We can then try to find the value of equation III in another way, with an adjoint version of the original PDE.

Initially, consider we have the following PDE with unknown ρ:

$$\int_{\Omega} \tau : \epsilon(\psi)\, dx = \int_{\Omega} \epsilon(\rho) : C : \epsilon(\psi)\, dx \tag{5}$$

, where ρ is from the same function space as ϕ (meaning it equals 0 on Dirichlet boundary) and ψ is from the same space as $D[u]$. Since ψ is used as test function here, it should be 0 at Γ_D, but how can we ensure it happens? Since values at Dirichlet boundary does not change, the material derivative is then 0, agreeing with what should happen to the test functions at Γ_D.

Then, after finding ρ, we could compute III using the forward PDE Equation 4 and the following:

$$III = \int_{\Omega} \tau : \epsilon(D[u])\, dx = \int_{\Omega} \epsilon(\rho) : C : \epsilon(D[u])\, dx$$

Obtaining:

$$\boxed{III = \int_{\Omega} \operatorname{sym}(\nabla\rho\nabla v) : \sigma + \epsilon(\rho) : C : \operatorname{sym}(\nabla u \nabla v) - \epsilon(\rho) : \sigma \nabla \cdot v dX + \int_{\Gamma_N} \rho \cdot \hat{T} \quad \nabla \cdot v \, d\Gamma_N} \tag{6}$$

Combining this with Equation 2, and noticing we can drop the sym operation (because its output is always double contracted with a symmetric tensor):

$$\boxed{dJ[v] = \int_{\Omega} (j - \epsilon(\rho) : \sigma) \nabla \cdot v + (\nabla\rho\nabla v) : \sigma + (\epsilon(\rho) : C - \tau) : (\nabla u \nabla v)\, dX}$$
$$\boxed{+ \int_{\Gamma_N} \rho \cdot \hat{T} \quad \nabla \cdot v \, d\Gamma_N} \tag{7}$$

The corresponding formulas for the periodic case can be found in section 3.1.2 of the suplementarial material of [1].

2.5 Discrete Differential Form for $dJ_d[v]$

The input of the differential form is a velocity vector field $v(x)$. It is discretized considering velocities δq_m on the vertices of our input mesh $v(x) = \sum_m \lambda_m(x)\,\delta q_m$. Note that, despite being a global formula, only values of vertices close to x should affect $v(x)$ (which usually means $\lambda_m(x)$ is different than 0 only if m is in the face containing x).

At the same time, ρ and u are discretized in similar manner, but using the n FEM nodes created during the simulation. Summarizing, we have:

$$\begin{aligned} v(x) &= \sum_m \lambda_m(x)\,\delta q_m \\ u(x) &= \sum_n \varphi_n(x)\,u_n \\ \rho(x) &= \sum_n \varphi_n(x)\,\rho_n \end{aligned}$$

We can then imply some important properties:

$$\begin{aligned} \nabla v &= \sum_m \delta q_m \otimes \nabla\lambda_m \\ \nabla\rho &= \sum_n \rho_n \otimes \nabla\varphi_n \\ \nabla u &= \sum_n u_n \otimes \nabla\varphi_n \\ \nabla\cdot v &= \sum_m \nabla\lambda_m \cdot \delta q_m \\ \nabla\rho\nabla v &= \sum_{m,n}(\rho_n \otimes \nabla\varphi_n)(\delta q_m \otimes \nabla\lambda_m) \\ \tau : (\nabla\rho\nabla v) &= \sum_m \delta q_m \cdot (\sum_n [\nabla\lambda_m \cdot (\tau\rho_n)]\,\nabla\varphi_n) \\ (\nabla\rho\nabla v) : \sigma &= \sum_m \delta q_m \cdot (\sum_n [\nabla\lambda_m \cdot (\sigma\rho_n)]\,\nabla\varphi_n) \end{aligned}$$

Let's consider again $dJ[v]$:

$$\begin{aligned} dJ[v] = \int_\Omega (\nabla\rho\nabla v) : \sigma + \epsilon(\rho) : C : (\nabla u\nabla v) - \epsilon(\rho) : \sigma\nabla\cdot v - \tau : (\nabla u\nabla v) + j\nabla\cdot v\,dX \\ + \int_{\Gamma_N} \rho\cdot\hat{T} \quad \nabla\cdot v\,d\Gamma_N \end{aligned}$$

Now, let's apply our transformations for discrete v, u and ρ to obtain dJ_d.

$$\boxed{dJ_d[\lambda_m\delta q_m] = \underbrace{\left(\int_\Omega [j - \epsilon(\rho) : \sigma]\,\nabla\lambda_m + [\nabla\lambda_m \cdot (\sigma p_n + (\epsilon(p) : C - \tau)u_n)]\,\nabla\varphi_n dx + \int_{\Gamma_N} (\rho\cdot\hat{T})\nabla\lambda_m d_\Gamma\right)}_{dJ_m} \cdot\delta q_m} \tag{8}$$

, which corresponds to final formula of Section 3.1.3 of Supplementary Material of [1].

With this formula, all you have to do in order to compute dJ_d is to sum the dot products of dJ_m with the velocity m (δq_m) for all mesh vertices.

References

[1] J. Panetta, A. Rahimian, and D. Zorin. Worst-case stress relief for microstructures. *ACM Transactions on Graphics*, 36(4), 2017.