Problem 10 Stress constraints

Tianyang Sun[*]

Problem 10: Stress constraints

Implement stress constraints for a problem of your choice. Use the general formulation:

$$\begin{aligned}
\min_{\boldsymbol{\rho}} &: \Phi(\boldsymbol{\rho}) \\
\text{s.t.} &: \mathbf{K}(\boldsymbol{\rho})\mathbf{u} = \mathbf{f} \\
&: \sum_{e=1}^{N} v_e \rho_e = \mathbf{v}^T \boldsymbol{\rho} \leqslant V^* \\
&: ||\boldsymbol{\sigma}_{VM}(\boldsymbol{\rho})||_P \leqslant \sigma_y^* \\
&: \mathbf{0} < \boldsymbol{\rho}_{\mathbf{min}} \leqslant \boldsymbol{\rho} \leqslant \mathbf{1},
\end{aligned} \tag{10}$$

where the global Von-Mises stress measure is defined as

$$||\boldsymbol{\sigma}_{VM}(\boldsymbol{\rho})||_P = \left(\sum_{e=1}^{N} v_e \sigma_{VM,e}^P \right)^{\frac{1}{P}} \tag{11}$$

An implementation for a mechanism design problem (including sensitivity analysis and appropriate density interpolation schemes) can be found in Leon et al. (2015).

Introduction

Traditional topology optimization methods often face issues such as insufficient structural safety and short fatigue life, which can lead to failures or performance degradation in practical applications. To address these problems, we naturally introduced stress-constrained topology optimization. This approach effectively prevents excessive local stresses under loading conditions, ensuring the rational use of materials and overall reliability. Additionally, we have found in practice that stress-constrained topology optimization successfully resolves the one-node-connected hinge problem.

Refer to reference 1, we used a normalized version of the commonly used p-norm of the effective von Mises stresses to approximate the maximum stress. And we reference this literature when conducting the sensitivity analysis.

Stress constraint

Following the reference [1], we use the von Mises effective stress $\sigma_{vm,i}$ for each element to characterize the magnitude of stress. Mathematically, it is expressed as:

$$\sigma_{vm,i} = \left(\boldsymbol{\sigma}_i^T \mathbf{V} \boldsymbol{\sigma}_i\right)^{\frac{1}{2}} = \left(\mathbf{u}_i^T \mathbf{B}^T \mathbf{C}_0^T \mathbf{V} \mathbf{C}_0 \mathbf{B} \mathbf{u}_i\right)^{\frac{1}{2}} =: \left(\mathbf{u}_i^T \mathbf{Q} \mathbf{u}_i\right)^{\frac{1}{2}}.$$

Here, $\boldsymbol{\sigma}_i$ is the element stress vector for the i-th element, $\mathbf{u}_i$ is the displacement vector for the i-th element, $\mathbf{V}$ is the auxiliary matrix, $\mathbf{C}_0$ is the elasticity tensor, and $\mathbf{B}$ is the strain-displacement matrix.

$$\mathbf{V} = \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad \mathbf{C}_0 = \frac{E}{1-\mu^2} \begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \frac{1-\mu}{2} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

[*]School of Mathematical Sciences USTC, `tysun@mail.ustc.edu.cn`

What we want to constrain is actually $\max_i \sigma_{vm,i}$, but this function is non-differentiable, and it is impractical to impose separate constraints on each $\sigma_{vm,i}$ due to the excessive number of constraints required. Therefore, we use the p-norm, which first introduced by [2] and expanded by [3],

$$\left(\sum_i \sigma_{vm,i}^p\right)^{\frac{1}{p}}$$

to approximate the ∞-norm. To balance the approximation accuracy and the nonlinearity of the p-norm, we follow the recommendation in [1] and choose $p = 12$. To eliminate the impact of regions that are expected to have inherently high stress, such as the two positions where springs are placed in a compliant inverter, we introduce N_σ, which represents the set of elements that need to be considered. Additionally, to address the issue of degenerate regions mentioned in [3], we repalace the von Mises effective stress by the relaxed stress $\hat{\sigma}_{vm,i} = x_{phys,i}^q \sigma_{vm,i}$, where q is a parameter that needs to be adjusted during the optimization process. In this way, the quantity we need to consider becomes

$$\sigma_{PN} = \left(\sum_{i\in N_\sigma} \hat{\sigma}_{vm,i}^p\right)^{\frac{1}{p}} = \left(\sum_{i\in N_\sigma} \left(x_{phys,i}^q \left(\mathbf{u}_i^T \mathbf{Q} \mathbf{u}_i\right)^{\frac{1}{2}}\right)^p\right)^{\frac{1}{p}} \tag{1}$$

Sensitivity Analysis

$$\begin{aligned}
\frac{\partial \sigma_{PN}}{\partial x_{Phys,i}} &= \sum_{j\in N_\sigma} \frac{\partial \sigma_{PN}}{\partial \hat{\sigma}_{vm,j}} \frac{\partial \hat{\sigma}_{vm,j}}{\partial x_{Phys,i}} = \left(\sum_{k\in N_\sigma} \hat{\sigma}_{vm,k}^p\right)^{\frac{1}{p}-1} \sum_{j\in N_\sigma} \hat{\sigma}_{vm,j}^{p-1} \cdot \frac{\partial \hat{\sigma}_{vm,j}}{\partial x_{Phys,i}} \\
&= \left(\sum_{k\in N_\sigma} \hat{\sigma}_{vm,k}^p\right)^{\frac{1}{p}-1} \sum_{j\in N_\sigma} \hat{\sigma}_{vm,j}^{p-1} \cdot \left(q x_{Phys,j}^{q-1} \delta_{ij} \sigma_{vm,j} + x_{Phys,j}^q \frac{\partial \sigma_{vm,j}}{\partial x_{Phys,i}}\right) \\
&= \left(\sum_{k\in N_\sigma} \hat{\sigma}_{vm,k}^p\right)^{\frac{1}{p}-1} \left(\hat{\sigma}_{vm,i}^{p-1} \cdot q \cdot x_{Phys,i}^{q-1} \cdot \sigma_{vm,i} + \sum_{j\in N_\sigma} \hat{\sigma}_{vm,j}^{p-1} \cdot x_{Phys,j}^q \frac{\partial \sigma_{vm,j}}{\partial x_{Phys,i}}\right) \\
\frac{\partial \sigma_{vm,j}}{\partial x_{Phys,i}} &= \frac{\partial \sigma_{vm,j}}{\partial \mathbf{u}_j}^T \cdot \frac{\partial \mathbf{u_j}}{\partial x_{Phys,i}} = \frac{1}{\sigma_{vm,j}} \mathbf{u_j}^T \cdot Q \cdot \frac{\partial \mathbf{u_j}}{\partial x_{Phys,i}} = \frac{1}{\sigma_{vm,j}} \mathbf{U}^T \cdot L_j^T \cdot Q \cdot L_j \cdot \frac{\partial \mathbf{U}}{\partial x_{Phys,i}} \\
\frac{\partial \sigma_{PN}}{\partial x_{Phys,i}} &= \left(\sum_{k\in N_\sigma} \hat{\sigma}_{vm,k}^p\right)^{\frac{1}{p}-1} \left(q \cdot x_{Phys,i}^{pq-1} \cdot \sigma_{vm,i}^p + \mathbf{U}^T \cdot \left(\sum_{j\in N_\sigma} x_{Phys,j}^{pq} \cdot \sigma_{vm,j}^{p-2} \cdot L_j^T Q L_j\right) \cdot \frac{\partial \mathbf{U}}{\partial x_{Phys,i}}\right)
\end{aligned}$$

Code implementation

- $\mathbf{u}_i^T \mathbf{Q} \mathbf{u}_i$: Each iteration of the optimization process requires a complete calculation
  - After calculation, it needs to be multiplied element-wise with another matrix.
  - `sum((U(edofMat) * Q) .* U(edofMat), 2)`
  - Using element-wise multiplication followed by summation instead of matrix multiplication perfectly achieves our goal. If matrix multiplication is used, a larger matrix will be obtained (because we use a unified large matrix instead of performing operations one by one), and only the diagonal elements of this larger matrix are what we want.
- $L_i^T Q L_i$: Can be precomputed, but it needs to be multiplied by sth that changes in each iteration.
  - My idea is to precompute $L_i^T Q L_i$ one by one, and either store them as a large sparse matrix where the i-th 8 rows correspond to $L_i^T Q L_i$, or store them as a three-dimensional array with the first dimension representing i. Then, during the loop, multiply them with the i-th component of the matrix and accumulate over i.

- GPT provided me with a completely new idea: because L_i is special, with each row having only a single non-zero element, and the position of the non-zero element in the j-th row is $edofMat(i,j)$, $L_i^T Q L_i$ must also be a sparse matrix. Moreover, we can predict which positions will be non-zero and determine which position in Q each of these non-zero elements comes from.

$$(L_i^T Q L_i)_{mn} = \sum_{k,j} l^T_{i,mk} q_{kj} l_{i,jn} \neq 0 \Longrightarrow \exists k,j \ s.t. m = edofMat(i,k), n = edfoMat(i,j)$$

  Thus, $L_i^T Q L_i$ can be constructed as follows: iterate over indices k,j from 1 to 8, and accumulate the element at position (k,j) in Q into the position $(edofMat(i,k), edofMat(i,j))$ of $L_i^T Q L_i$. Thus, we can store the information of L_i^T and L_i as row indices and column indices to record their positions. Then, flatten $V_i Q$ into a column vector to store the values. Carefully align these three vectors (row indices, column indices, and values), and finally use the sparse syntax to generate the entire matrix.

```matlab
[I_elem, J_elem] = meshgrid(1:8, 1:8);
I_elem = I_elem(:)'; % 1 x 64
J_elem = J_elem(:)'; % 1 x 64

edofMat_expanded_I = edofMat(:, I_elem); % nele x 64
edofMat_expanded_J = edofMat(:, J_elem); % nele x 64

I = edofMat_expanded_I(:); % (nele * 64) x 1
J = edofMat_expanded_J(:); % (nele * 64) x 1

Q_vector = Q(:); % 64 x 1

G_elems = V * Q_vector'; % nele x 64
G = G_elems(:); % (nele * 64) x 1

S = sparse(I, J, G, ndof, ndof);
```

- $\mathbf{U}^T S \dfrac{\partial \mathbf{U}}{\partial x_{Phys,i}}$
  - If it were up to me, I would solve each column vector based on the system of linear equations

$$K \frac{\partial \mathbf{U}}{\partial x_{Phys,i}} = -\frac{\partial K}{\partial x_{Phys,i}} \mathbf{U}$$

    one by one, which would require solving $nele$ linear systems. This would result in a significant computational cost.
  - GPT suggests solving the adjoint equation

$$K\lambda = -SU \Longrightarrow \mathbf{U}^T S \frac{\partial \mathbf{U}}{\partial x_{Phys,i}} = \lambda^T \frac{\partial K}{\partial x_{Phys,i}} \mathbf{U},$$

    which would only require solving a single linear system.

Results and Analysis

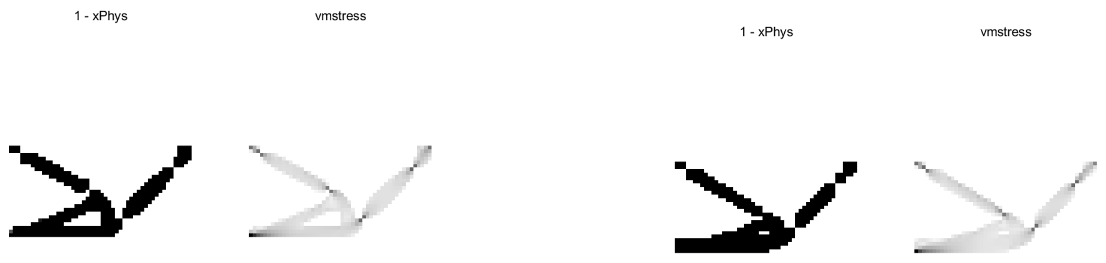

Figure 1: With (left) and without (right) stress constraints

- Without Stress Constraints: The objective function value is -4.7137, with a maximum stress of 1.2617.
- With Stress Constraints: The objective function value is -4.7654, with a maximum stress of 1.1500.

This assignment only completed a sensitivity analysis, and the experimental results under various parameters were not satisfactory. The reason is that the understanding of the optimization process is still insufficient. It would be better to analyze the impact of parameters on optimization effects using simpler models.

References

[1] Daniel M De Leon, Joe Alexandersen, Jun S O. Fonseca, and Ole Sigmund. Stress-constrained topology optimization for compliant mechanism design. *Structural and Multidisciplinary Optimization*, 52:929–943, 2015.

[2] Pierre Duysinx and Ole Sigmund. New developments in handling stress constraints in optimal material distribution. In *7th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization*, page 4906.

[3] Chau Le, Julian Norato, Tyler Bruns, Christopher Ha, and Daniel Tortorelli. Stress-based topology optimization for continua. *Structural and Multidisciplinary Optimization*, 41:605–620, 2010.