Electronic Supplementary Materials

For https://doi.org/10.1631/jzus.A2000380

A deep neural network-based algorithm for solving structural optimization

Dung Nguyen KIEN, Xiaoying ZHUANG

Algorithm 1 Range-reduction technique

Input: $\epsilon, \alpha_i, \beta_i$

Output: The minimum $\boldsymbol{x}*$ and then find $\hat{g}_0^*$

Initialize Δx_i, range of x_i
First run Algorithm 1 to find $\boldsymbol{x}^*$ and $\hat{g}_0^*$
if $|\hat{g}_0^r - \hat{g}_0^*| > \epsilon$ **then**
  Reduce the range of x_i and interval size Δx_i:
    $x_i^{\min} = x_i^{\min}(\text{old}) - \text{ff}_\text{i}$
    $x_i^{\max} = x_i^{\min}(\text{old}) + \text{ff}_\text{i}$
    $\Delta x_i = \Delta x_i(\text{old}) - \text{fi}_\text{i}$
  Second run Algorithm 1 to find $\boldsymbol{x}^*$ and $\hat{g}_0^*$
end if
(where $\hat{g}_0^r$ is the minimum value of the objective function in exact solution or reference, α_i, and β_i are reduced constants)

Algorithm 2 Deep Lagrange method for solving shape optimization problem

Input: Sheet dimension, sheet material properties, constraint, deep Lagrange method initialization

Output: Shape optimization of the sheet

for $(\alpha_{kl},\ k \leftarrow 1, 2;\ l \leftarrow 1, 2)$ **do**
  Generate $\boldsymbol{X}$ via 2D B-spline surface,
  for $(\lambda_j,\ j = 1, 2, \ldots, n_\lambda)$ **do**
    Calculate $g_0 = \boldsymbol{F}^{\mathrm{T}}\boldsymbol{u}$
    $g_1 = \int_{\Omega} \boldsymbol{N}|\boldsymbol{J}|t\mathrm{d}\Omega - W_{\max}$
    Calculate target $\boldsymbol{t} = \mathcal{L}^3(\boldsymbol{x}, \boldsymbol{\lambda}) = g_0 + \lambda_{j1} g_1$
    Apply DLM to solve the optimization problem
  end for
end for

Algorithm 3 Deep Lagrange method

Input: Define input $\boldsymbol{X} = [\boldsymbol{x}]$, range of $\boldsymbol{\lambda}$, weight $\boldsymbol{w}$ initialization, neural network hyper-parameters configuration

Output: The minimum $\boldsymbol{x}^*$ and then find $\hat{g}_0^*$

1. Find the minimum of deep neural network output

for $i \leftarrow 1, I$ **do**
  Get λ_i
  Obtain target $\boldsymbol{t} = \mathcal{L}(\boldsymbol{x}, \lambda_i) = \hat{g}_0(\boldsymbol{x}) + \sum_{j=1}^{m} \lambda_{ji}\hat{g}_j(\boldsymbol{x})$, where $\hat{g}_0(\boldsymbol{x}) \le 0$
  1.1. Deep neural network calculation.
  for $k \leftarrow 1, K$ **do**
    Feedforward:
    Compute $\hat{\boldsymbol{z}}^k = f_k(\boldsymbol{w}^k\hat{\boldsymbol{z}}^{k-1} + \boldsymbol{b}^k)$
    At $k = 1 : \hat{\boldsymbol{z}}^1 = f_1(\boldsymbol{w}^1\boldsymbol{X} + \boldsymbol{b}^1)$
    Set $f_1 = f_2 = \ldots = f_K = f$
    Define loss function $L(\boldsymbol{\theta})$
    Backpropagation:
    Gradient of L computation in the output layer: $\frac{\partial L}{\partial \boldsymbol{\theta}^K} = \left[\frac{\partial L}{\partial \boldsymbol{w}^K}; \frac{\partial L}{\partial \boldsymbol{b}^K}\right]^{\mathrm{T}}$
    First, $\frac{\partial L(\hat{\boldsymbol{z}}^K(\boldsymbol{\theta}))}{\partial \boldsymbol{z}^K} = \frac{\partial L(\hat{\boldsymbol{z}}^K)}{\partial \hat{\boldsymbol{z}}^K}\frac{\partial \hat{\boldsymbol{z}}^K}{\partial \boldsymbol{z}^K} = L'(f_K)' = \boldsymbol{\delta}^K$, then, $\frac{\partial L}{\partial \boldsymbol{w}^K} = \boldsymbol{\delta}^K\hat{\boldsymbol{z}}^k$ and $\frac{\partial L}{\partial \boldsymbol{b}^K} = \delta^K$
    Gradient L computation in the kth layer: $\frac{\partial L}{\partial \boldsymbol{\theta}^k} = \left[\frac{\partial L}{\partial \boldsymbol{w}^k}; \frac{\partial L}{\partial \boldsymbol{b}^k}\right]^{\mathrm{T}}$
    Similar to the previous, $\frac{\partial L}{\partial \boldsymbol{\theta}^k} = \left(\boldsymbol{\delta}^K\boldsymbol{w}^K(f_k)'\right)\hat{\boldsymbol{z}}^{k-1} = \boldsymbol{\delta}^k\hat{\boldsymbol{z}}^{k-1}$ and $\frac{\partial L}{\partial \boldsymbol{b}^k} = \boldsymbol{\delta}^k$
    Weight update:
    Update weights and biases by a gradient-based optimization algorithm in every layer, for example, stochastic gradient descent:
    for $k \leftarrow K, 1$ **do**
      $\boldsymbol{w}^k_{\text{new}} = \boldsymbol{w}^k + \gamma\frac{\partial L}{\partial \boldsymbol{w}^k}$
      $\boldsymbol{b}^k_{\text{new}} = \boldsymbol{b}^k + \gamma\frac{\partial L}{\partial \boldsymbol{b}^k}$
      (where γ is the learning rate)
    end for
  end for
  1.2. Obtain the NN output from the input data set
  1.3. Find the minimum of the NN output $\phi(\lambda_i) = \min \hat{\boldsymbol{z}}^K(\boldsymbol{\theta})$
end for
2. Find $\phi_{\max} = \max_{\boldsymbol{\lambda}} \phi(\boldsymbol{\lambda})$
3. Find the minimum $\boldsymbol{x}^*$ at $\phi_{\max}$
4. Find the optimum $\hat{g}_0^*$

Algorithm 4 Numerical methods for solving shape optimization problem

Input: Sheet dimension, sheet material properties, and constraint

Output: Shape optimization of the sheet

for (each design variables $\alpha_{kl},\ k \leftarrow 1,2;\ l \leftarrow 1,2$) **do**

  Generate nodal coordinates matrix of finite element $\boldsymbol{X}'$, via B-spline surface,

  Calculate $g_0 = \boldsymbol{F}^{\mathrm{T}}\boldsymbol{u}$

  Calculate $g_1 = \int_{\Omega} \boldsymbol{N}|\boldsymbol{J}|\mathrm{d}\Omega - W_{\max}$,

  for (each design element (DE) of which the shape is affected by α_{kl}) **do**

    Calculate $\dfrac{\partial \boldsymbol{X}'}{\partial \alpha_{kl}}$ for all nodes in DE,

    for (each finite element e in DE) **do**

      Evaluate $\dfrac{\partial \boldsymbol{G}}{\partial \alpha_{kl}} = -\boldsymbol{G}\dfrac{\partial \boldsymbol{X}'}{\partial \alpha_{kl}}\boldsymbol{G}$,

      $\dfrac{\partial |\boldsymbol{J}|}{\partial \alpha_{kl}} = |\boldsymbol{G}|\mathrm{tr}\Big(\boldsymbol{G}\dfrac{\partial \boldsymbol{X}'}{\partial \alpha_{kl}}\Big)$,

      Evaluate the sensitivity of element stiffness and applied force $\dfrac{\partial \boldsymbol{k}_e}{\partial \alpha_{kl}} = \int_{\Omega_e} \Big(\dfrac{\partial \boldsymbol{B}^{\mathrm{T}}}{\partial \alpha_{kl}}\boldsymbol{D}\boldsymbol{B}|\boldsymbol{J}| + \boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\dfrac{\partial \boldsymbol{G}}{\partial \alpha_{kl}}|\boldsymbol{J}| + \boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\dfrac{\partial |\boldsymbol{J}|}{\partial \alpha_{kl}}\Big)\mathrm{d}\Omega_e$,

      $\dfrac{\partial \boldsymbol{f}_e}{\partial \alpha_{kl}} = \int_{l_e} \boldsymbol{N}\bar{p}t\Big(\sqrt{J_{11}^2 + J_{12}^2}\Big)^{-1/2}\big(J_{11}J'_{11} + J_{12}J'_{12}\big)\mathrm{d}\xi$

      Calculation of pseudo-load by assembling $\dfrac{\partial \boldsymbol{F}_e}{\partial \alpha_{kl}} - \dfrac{\partial \boldsymbol{K}_e}{\partial \alpha_{kl}}\boldsymbol{u} = \underset{e=1}{\overset{n}{\boldsymbol{A}}}\left[\dfrac{\partial \boldsymbol{f}_e}{\partial \alpha_{kl}} - \dfrac{\partial \boldsymbol{k}_e}{\partial \alpha_{kl}}\boldsymbol{u}_e\right]$,

    end for

  end for

  Solve for $\dfrac{\partial \boldsymbol{u}_e}{\partial \alpha_{kl}}$: $\boldsymbol{K}\dfrac{\partial \boldsymbol{u}_e}{\partial \alpha_{kl}} = \dfrac{\partial \boldsymbol{F}_e}{\partial \alpha_{kl}} - \dfrac{\partial \boldsymbol{K}_e}{\partial \alpha_{kl}}\boldsymbol{u}$,

  Calculate the sensitivities of objective function and constraint functions $\dfrac{\partial \hat{g}_i}{\partial \alpha_{kl}} = \dfrac{\partial g_i}{\partial \alpha_{kl}} + \dfrac{\partial g_i}{\partial \boldsymbol{u}}\dfrac{\partial \boldsymbol{u}}{\partial \alpha_{kl}}$

  So the sensitivities are written as $\dfrac{\partial \hat{g}_0}{\partial \alpha_{kl}} = 2\boldsymbol{u}^{\mathrm{T}}\dfrac{\partial \boldsymbol{F}_e}{\partial \alpha_{kl}} - \boldsymbol{u}^{\mathrm{T}}\dfrac{\partial \boldsymbol{K}_e}{\partial \alpha_{kl}}\boldsymbol{u}$,

  $\dfrac{\partial \hat{g}_1}{\partial \alpha_{kl}} = \int_{\Omega}\Big(\boldsymbol{N}\dfrac{\partial |\boldsymbol{J}|}{\partial \alpha_{kl}}\Big)\mathrm{d}\Omega$

  Apply MMA/SLP/MPEA method to solve the optimization problem.

end for