ABSTRACT

Structural engineers face several code-restricted design decisions. Codes impose many conditions and requirements to the designs of structural frames, such as columns and beams. However, it is difficult to intuitively find optimized solutions, while satisfying all code requirements simultaneously. Engineers commonly make design decisions based on empirical observations. Optimization techniques can be employed to make more rational engineering decisions, which result in designs that can meet various code restrictions simultaneously. Lagrange optimization techniques with constraints, not based on explicit parameterization, are implemented to make rational engineering decisions and find minimized or maximized design values by solving nonlinear optimization problems under strict constraints imposed by design codes. It is difficult to express objective functions analytically directly in terms of design variables to use derivative methods, such as Lagrange multipliers. This study proposes the use of neural network to approximate well-behaved objective functions and other output parameters into one universal function that can also give a generalizable solution for operating Jacobian and Hessian matrices to solve the Lagrangian function. The proposed method was applied successfully in optimizing a cost of a reinforced concrete column under various design requirements. An efficacy of optimal results was also proven by 5 million datasets.

1. Introduction

Several studies, including (Aghaee, Yazdi, and Tsavdaridis 2014), (Fanaie, Aghajani, and Dizaj 2016), (Madadi et al. 2018), (Nasrollahi et al. 2018), and (Paknahad, Bazzaz, and Khorami 2018), have been conducted to optimize reinforced concrete (RC) structures. These studies mainly focused on minimizing the manufacturing and construction costs; only a few considered structural capabilities against external forces, which are influenced by design codes. Studies of the optimization of RC beams have been reported by (Shariati et al. 2010), (Fanaie, Aghajani, and Shamloo 2012), (Toghroli et al. 2014), (Awal, Shehu, and Ismail 2015), (Kaveh and Shokohi 2015), (Safa et al. 2016), (Shah et al. 2016), (Korouzhddeh, Eskandari-Naddaf, and Gharouni-Nik 2017), and (Heydari and Shariati 2018).

Barros, Martins, and Barros (2005) presented stress–strain diagrams using the conventional Lagrangian multiplier method (LMM) to develop nominal moment strengths based on the optimal area of the upper and lower sections of steel for four classes of concrete. (Shariat et al. 2018) obtained analytical objective functions for the cost of frames as a function of the design parameters for structural systems in a limited...
circumstance; however, it is generally very difficult to derive analytical objective functions that represent the entire behavior of structural components such as columns and beams. (Villarrubia et al. 2018) employed artificial neural networks (ANNs) to approximate objective functions to optimized analytical objective functions. They approximated objective functions using nonlinear regression that can optimize problems and employing a multilayer perceptron when the use of linear programming or Lagrange multipliers was not feasible.

2. Significance of the study

There are numerous available computer-aided engineering tools, including CAD packages, FEM software, self-write calculation codes, that are used to study the performance of structures. Objective functions, however, could constitute a mixture of numerical simulations, analytical calculations, and catalog selections, which makes it difficult to apply differentiation to derivative optimization methods, such as the Lagrange multiplier. Some nonderivative optimization methods such as Genetic algorithms were applied in structural design problems (Rajeev and Krishnamoorthy 1998)(Camp, Pezeshk, and Hansson 2003) as they do not require any derivatives to find an optimal solution. However, computing times of nonderivative methods heavily rely on the computational cost of engineering tools because each trial requires one run of software. In this study, the use of artificial neural networks was adopted to universally approximate objective functions obtained from any computer-aided engineering tools. New objective functions, hence, not only enhance the computational speed compared to conventional software but also can be differentiated and implemented in Lagrange optimization.

Optimization and sensitivity analyses using computational LMM can be based on ANNs. The optimized results were verified using rectangular RC columns. The analysis was conducted to obtain the minimum design cost for reinforced concrete columns, as specified by the American Concrete Institute (ACI) regulations (2014) (ACI-318 2014). Moreover, a sensitivity analysis was performed on the cost with respect to the effective parameters, including the rebar ratios and failure criteria. Accordingly, various failure criteria were developed to be used in designing RC columns. Numerical examples were also presented to better illustrate the design steps. Complex but inaccurate analytical objective functions, such as describing the cost of structural frame and emissions of CO2, were replaced by ANNs-based objective functions. The sensitivity analysis of the LMM revealed that the best optimal values based on constraints can be identified for specific situations. Optimization using artificial intelligence (AI)-based objective functions based on large datasets without the need for primary optimization knowledge can effectively aid the selection of design parameters for best practices.

The goodness of proposed method is that its performance would be less dependent on problem types such as column, beam, frame, seismic design, etc. but relies on characteristics of big datasets of considered problem. Once big data is good enough to generate approximation objective function as well as other parameters using ANN, an optimization solution is then generalizable by using AI-based Lagrange method. Therefore, applications of proposed method would not restrict to optimizing RC columns only but also can expand to other problems, such as optimizing beam-column connection (Ye et al. 2021), GFRP RC Columns (Sun et al. 2020), RC shear wall (Zhang et al. 2019)(Yazdani, Fakhimi, and Alitalesh 2018) subjected to lateral impact loading (Zhang, Gholipour, and Mousavi 2021), or even severe impulsive loading (Abedini and Zhang 2021), etc.

3. Lagrange procedures based on ANNs

Joseph–Louis LMM optimizes objective functions with constraints, identifying the saddle point of the Lagrange function, as mentioned by (Walsh 1975) and (Kalman 2009), which can be identified among local stationary points based on the Hessian matrix (which are differentiated twice) (Silberberg and Wing. 2001). To find the stationary (saddle) points of a Lagrangian function, the function must be formed as a function of the constraining input variables and the Lagrange multiplier λ ((Protter and Morrey 1985)). This can be achieved by solving systems of nonlinear differential equations that lead to the identification of the maximum or minimum of the Lagrange function subjected to inequality and equality constraints (Hoffmann and Bradley 2004).

3.1. Optimization using LMM and Newton–Raphson method

LMM finds stationary points (saddle points; maximum or minimum of a Lagrange function) when a Lagrange function, \( L \), is considered as a function of the variables \( x = [x_1, x_2, ..., x_n]^T \) and the Lagrange multiplier for both equality and inequality constraints, \( \lambda_e = [\lambda_{e1}, \lambda_{e2}, ..., \lambda_{en}]^T \) and \( \lambda_i = [\lambda_{i1}, \lambda_{i2}, ..., \lambda_{in}]^T \), respectively, as shown in Eq. (1):

\[
L(x, \lambda_e, \lambda_i) = f(x) - \lambda_e^T c(x) - \lambda_i^T v(x)
\]  

(1)
where \( f(x) \) is a multivariate objective function subject to equality and inequality constraints, \( c(x) = [c_1(x), c_2(x), ..., c_m(x)]^T = 0 \) and \( v(x) = [v_1(x), v_2(x), ..., v_n(x)]^T \geq 0 \), respectively.
The diagonal matrix of the inequality, $S$ (Eq. (2)), activates the inequality to equality if the condition of inequality is not satisfied or deactivate it if considered parameters are within the range defined by inequality constraints.

$$S = \begin{bmatrix} s_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & s_l \end{bmatrix}$$ (2)

where $s_i$ is the status of the inequality constraint $v_i$; $s_i = 1$ when $v_i$ is active and $s_i = 0$ when $v_i$ is inactive. The stationary points of Lagrange functions (Eq. (1)) can then be identified by solving the partial differential equations with respect to $x, \lambda_c$, and $\lambda_v$ (Eq. (3)), finding slopes of the parallel tangential lines of the objective functions, $f(x)$, and constraints, $c(x)$ and $v(x)$.

$$\nabla L(x, \lambda_c, \lambda_v) = \begin{bmatrix} \nabla f(x) - J_v(x)^T \lambda_v - J_c(x)^T S \lambda_v \\ -c(x) \\ -Sv(x) \end{bmatrix}$$ (3)

where

$$J_c(x) = \begin{bmatrix} \nabla c_1(x) \\ \vdots \\ \nabla c_m(x) \end{bmatrix}$$

$$J_v(x) = \begin{bmatrix} \nabla v_1(x) \\ \vdots \\ \nabla v_l(x) \end{bmatrix}$$ (4)

are the Jacobian of the constraint vectors $c$ and $v$, respectively, at $x$. The main advantage of Lagrange multipliers is that they are added to constrained optimization problems to convert the problems into unconstrained optimization problems. Lagrangian functions are formulated based on the relationships between the gradients of objective functions and those of the constraints of the original problems (Beavis and Dobbs, 1990) such that the derivative test for unconstrained problems can still be applied. Lagrange algorithms linearize restrictions and objective functions at a specific space point by employing derivatives and partial derivatives that are solved based on equality constraints, as shown in Eq. (3). The Newton–Raphson method is employed in solving a set of partial differential equations representing a tangential line of Lagrange functions (Eqs. (3) and (5)), $\nabla L(x, \lambda_c, \lambda_v)$, which needs to be differentiated one more time to linearize the partial differential Lagrange functions with respect to $x, \lambda_c$, and $\lambda_v$ (Eq. (5)), leading to finding the stationary points of the Lagrange function, $L(x, \lambda_c, \lambda_v)$. Linear approximation of tangential line of Lagrange functions can be predicted as $x^{(0)} + \Delta x$, which is very close to $x^{(0)}$, as shown in Eq. (5), where partial differential equations, $\nabla L(x, \lambda_c, \lambda_v)$, is differentiable at $x^{(0)}$. The Newton–Raphson method is based on first-order approximation, which works for any system of equations whose functions are differentiable in the considered region.

$$\nabla L(x^{(0)} + \Delta x, \lambda^{(0)}_c, \lambda^{(0)}_v + \Delta \lambda_c, \lambda^{(0)}_v + \Delta \lambda_v)$$

$$\approx \nabla L(x^{(0)}, \lambda^{(0)}_c, \lambda^{(0)}_v) + [H_L(x^{(0)}, \lambda^{(0)}_c, \lambda^{(0)}_v)] \begin{bmatrix} \Delta x \\ \Delta \lambda_c \\ \Delta \lambda_v \end{bmatrix}$$ (5)

where $[H_L(x^{(0)}, \lambda^{(0)}_c, \lambda^{(0)}_v)]$ is the Hessian matrix of Lagrangian $L$ at $x^{(0)}, \lambda^{(0)}_c, \lambda^{(0)}_v$. Since seeking $\nabla L(x^{(0)} + \Delta x, \lambda^{(0)}_c + \Delta \lambda_c, \lambda^{(0)}_v + \Delta \lambda_v) = 0$, $[\Delta x \Delta \lambda_c \Delta \lambda_v]^T$ can be computed from Eq. (6). Generally, the variable $x$ and Lagrange multipliers $\lambda$ can be updated after every iteration as Eq. (7).

$$\begin{bmatrix} \Delta x \\ \Delta \lambda_c \\ \Delta \lambda_v \end{bmatrix} = -[H_L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v)]^{-1} \nabla L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v)$$ (6)

$$\begin{bmatrix} x^{(k+1)} \\ \lambda^{(k+1)}_c \\ \lambda^{(k+1)}_v \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ \lambda^{(k)}_c \\ \lambda^{(k)}_v \end{bmatrix} - [H_L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v)]^{-1} \nabla L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v)$$ (7)

The first derivative, $\nabla L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v)$, can be obtained from Eq. (3), while the Hessian matrix $[H_L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v)]$, of second-order partial derivative of Lagrange function, $L(x, \lambda_c, \lambda_v) = f(x) - \lambda^T c(x) - \lambda_v^T Sv(x)$ (Eq. (1)), is derived as followed:

$$\begin{bmatrix} H_L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v) \\ -J_c(x^{(k)})^T \\ -Sv(x^{(k)}) \end{bmatrix} = \begin{bmatrix} H_L(x^{(k)}, \lambda^{(k)}_c, \lambda^{(k)}_v) \\ -J_c(x^{(k)})^T \\ -Sv(x^{(k)}) \end{bmatrix}$$ (8)

$$H_L(x) = H_f(x) - \sum_{i=1}^{m} \lambda_i H_c(x) - \sum_{i=1}^{f} s_i H_v(x)$$ (9)

where $H_f(x), H_c(x)$, and $H_v(x)$ are Hessian matrices of the objective function ($f(x)$), equality constraint ($c_i$), and inequality constraint ($v_i$), respectively, with respect to the variable vector ($x$). The procedure for Newton–Raphson approximation is repeated until convergence is achieved.

### 3.2. Generalization of objective functions and their derivatives using neural network

#### 3.2.1. Formulation of universal approximation function using neural network

One problem in Lagrange optimization is that the objective function, $f(x)$, and/or the output parameters that appear in constraints, $c(x)$ and $v(x)$, are sometimes complex or impossible to derive into twice-differentiable functions in order to employ the Lagrange optimization method. Even when it is possible, deriving the Jacobian and
3.2.2. Neural network-based universal approximation function

\[ f(x) = g^{(0)} \]

\[ f_{lin}(i) \left( W^{(l-1)} f^{(l-1)}(x) + b^{(l)} \right) \]

where \( x \) is the input (vector of features); \( L \) the number of layers, including hidden and output layers; \( W^{(l)} \) the weight matrix between layer \( l-1 \) and layer \( l \); \( b^{(l)} \) the bias matrix of layer \( l \); and \( g^{(N)} \) and \( g^{(0)} \) the normalization and denormalization functions, respectively. Min-max normalization function as shown in Equation (10b) is conducted in this study.

\[ \bar{x} = g^{(N)}(x) = a_x (x - x_{\text{min}}) + x_{\text{min}} \]

\[ a_x = \frac{x_{\text{max}} - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \]

where \( \bar{x} \) is a normalization of \( x \) between the minimum and maximum value of \( x_{\text{min}} = -1 \) and \( x_{\text{max}} = 1 \), respectively. A coefficient \( a_x \) is the ratio of normalization data range \( (x_{\text{max}} - x_{\text{min}}) \) to original data range \( (x_{\text{max}} - x_{\text{min}}) \) as shown in Eq. (10c). Similarly, a denormalization function is expressed in Eq. (10d).

\[ x = g^{(0)}(\bar{x}) = \frac{1}{a_x} (x - x_{\text{min}}) + x_{\text{min}} \quad (10d) \]

The activation functions (\textit{tanh}, \textit{tanh}) shown in Figure 1, \( f_l \) at layer \( l \), were implemented to formulate nonlinear relationships between the networks. As mentioned by Goodfellow (Bengio, Goodfellow, and Courville 2017), the hyperbolic tangent activation function \( \text{tanh} \) generally performs better than a \textit{sigmoid} activation function. The function takes any real value as input and outputs values in the range \(-1 \) to \( 1 \). The bigger the input, the closer the output value to \( 1 \), whereas the smaller the input the closer the output to \(-1 \). Figure 1 also illustrates first and second derivatives of \textit{tanh} activation function, which are needed for Jacobian and Hessian calculations. A linear activation function, \( f_{lin}^{(l)} \), was selected for the output layer because the output values are unbounded. For example, an output of safety factor (\( SF \)) varies from 0.5 (normalized as \(-1 \)) to 2 (normalized as 1) in training datasets, however, \( SF \) could be either greater than 2 (normalized as 1) or small than 0.5 (normalized as \(-1 \)) depending on design values. \textit{sigmoid} or \textit{ReLU} activation functions for an output layer are even worse for this case because their lower bounds are 0 for any normalized safety factor smaller than 0 (denormalized as 1.25), which means they may have vanishing problems in these ranges. Linear activation at the output layer, on the other hand, predicts output values which is not influenced by activation functions.

![Figure 1. Tansig activation function and its derivatives.](image-url)
3.2.3. Formulation of Jacobian matrix for universal approximation function

In neural networks, universal approximation functions can also be expressed as a series of composite mathematical operations, as shown in Eq. (11a):

\[ y = f(x) = z^{(D)}(x) = z^{(l)}(x) = z^{(l-1)}(x) \ldots = z^{(N)}(x) \] (11a)

where \( z^{(l)} \) is the output vector at layer \( l \), which can be calculated using Eq. (11b):

\[ z^{(l)} = f^{(l)}(W^{(l)}z^{(l-1)} + b^{(l)}) \]

\[ z^{(l-1)} = f^{(l-1)}(W^{(l-1)}z^{(l-2)} + b^{(l-1)}) \]

where \( z^{(N)} = g^{(N)}(x) = \alpha_x \odot (x - \mathbf{x}_{\text{min}}) + \mathbf{x}_{\text{min}} \)

\[ y = z^{(D)} = g^{(D)}(y) = \frac{1}{\alpha_y} (z^{(l)} - y_{\text{min}}) + y_{\text{min}} \] (11b)

where \( \odot \) denotes the Hadamard (element-wise) product operation, \( \alpha_y, y_{\text{min}}, \) and \( y_{\text{min}} \) are normalization parameters of output \( y \); whereas \( \alpha_x = [\alpha_{x_1}, \alpha_{x_2}, \ldots, \alpha_{x_n}]^T \), \( \mathbf{x}_{\text{min}} = [x_{1,\text{min}}, x_{2,\text{min}}, \ldots, x_{n,\text{min}}]^T \) are vectors contained normalization parameters of input vectors \( x \).

In calculus, a chain rule is employed to calculate derivatives of such composite functions, as shown in Eq. (11). Formally, the Jacobian matrix of \( z^{(l)} \) with respect to \( x \) can be derived as the Jacobian matrix of \( z^{(l)} \) with respect to \( z^{(l-1)} \) multiplied by that of \( z^{(l-1)} \) with respect to \( x \) (Eq. (12))

\[ J^{(l)} = \frac{\partial z^{(l)}}{\partial x} = \frac{\partial z^{(l)}}{\partial z^{(l-1)}} \frac{\partial z^{(l-1)}}{\partial x} = \frac{\partial z^{(l)}}{\partial z^{(l-1)}} J^{(l-1)} \] (12)

The Jacobian matrix for universal approximation, \( J^{(l)} \), is then computed by forward propagation as follows:

\[ J^{(l)} = \frac{\partial z^{(N)}}{\partial x} = \frac{\partial (\alpha_x \odot (x - \mathbf{x}_{\text{min}}) + \mathbf{x}_{\text{min}})}{\partial x} = l_n \odot \alpha_x \]

![Figure 2](image-url)  
**Figure 2.** Size configuration of output vector, Jacobian, and Hessian of hidden layer \( l \).
Conventional design
(E.g. Reinforced concrete design following ACI-318 codes)
Inputs: \([b, h, \rho_{as}, f_c', f_y, P_u, M_u]\)
Output: \([\phi P_u, \phi M_u, SF, b/h, \epsilon_b, CIC, CO_2, W_c, \alpha_{ef/E_b}]\)

Big data generation based on conventional design

Neural Network Training

\[
\begin{align*}
J^1 & = \left( 1 - (z^{(1)})^2 \right) \odot W^{(1)} J^{(0)} \\
& \quad \vdots \\
J^{(l-1)} & = \left( 1 - (z^{(l-1)})^2 \right) \odot W^{(l-1)} J^{(l-2)} \\
J^{(l)} & = W^{(l)} J^{(l-1)} \\
J^{(D)} & = \frac{1}{\alpha_y} J^{(l)}
\end{align*}
\] (13)

3.2.4. Formulation of Hessian matrix for universal approximation function

This section summarizes derivations of AI based Hessian matrix which was developed by MathWorks Technical Support Department ((MATLAB 2020b). The Jacobian matrix of the hidden layer \(l\) and \(m^{(l)}\) neurons with respect to \(n\) input variables \(x = [x_1, x_2, \ldots, x_n]^T\) is
Figure 4. RC column.

Table 1. Summary of RC column parameters.

| RC COLUMN PARAMETERS | Description |
|----------------------|-------------|
| b                    | Column section width |
| h                    | Column section height |
| $\rho_s$             | Geometric rebar ratio |
| $f_c$                | Concrete strength |
| $f_y$                | Rebar yield strength |
| $M_i$                | Factored axial force |
| $M_i'$               | Factored bending moment |
| $\phi P_n$           | Design axial force |
| $\phi M_n$           | Design moment |
| SF                   | Safety factor ($\phi P_n / M_n$) |
| $b/h$                | Aspect ratio of column section |
| $C_l$                | Cost index of column per height |
| $C_{\text{CO}_2}$    | $\text{CO}_2$ emission per column height |
| $W_i$                | Weight of column per column |
| $h_i$                | Height |
| $\alpha_{i,h}$       | Angular eccentricity of applied load |

a matrix of size $m^{(l)} \times n$. Then, the corresponding Hessian is a third-order tensor as illustrated in Figure 2, which could be an expensive operation. A convenient method is to calculate explicitly the slices of the Hessian of the second derivative and then obtain the full Hessian by accurately reshaping the slices (MATLAB 2020b). A slice of the Hessian, $H_{i}^{(l)}$, is a derivative of Jacobian $J^{(l)}$ with respect to one of the input elements, $x_i$ (Eq. (14)).

$$H_{i}^{(l)} = \frac{\partial^2 z^{(l)}}{\partial x_i \partial z^{(l-1)}} J^{(l-1)} + \frac{\partial^2 z^{(l)}}{\partial z_{i}^{(l-1)}} H_{i}^{(l-1)}$$  (14)

where $\frac{\partial^2 z^{(l)}}{\partial x_i \partial z^{(l-1)}}$ can be obtained by applying the chain rule as:

$$\frac{\partial^2 z^{(l)}}{\partial x_i \partial z^{(l-1)}} = \frac{\partial^2 z^{(l)}}{\partial x_i \partial z^{(l-1)}} \odot \frac{\partial z^{(l-1)}}{\partial x_i}$$  (15)

The expression $\frac{\partial z^{(l-1)}}{\partial x_i}$ can be written as $i^{(l-1)}$, which is the $i$-th column of Jacobian $J^{(l-1)}$. Substituting Eq. (15) to Eq. (14), we obtain

$$H_{i}^{(l)} = \frac{\partial^2 z^{(l)}}{\partial z^{(l-1)}} H_{i}^{(l-1)} + \frac{\partial^2 z^{(l)}}{\partial z^{(l-1)}} \odot i^{(l-1)} \odot J^{(l-1)}$$  (16)

Applying forward propagation in Eq. (16), the slice of Hessian at the final layer, $L$, can be obtained as:

$$H_{i}^{(N)} = \frac{\partial^2 z^{(N)}}{\partial x_i \partial x} = 0$$

$$H_{i}^{(1)} = -2z^{(1)} \odot \left(1 - \left(z^{(1)}\right)^2 \right) \odot i^{(N)} \odot \left(W^{(1)}\right)^2 J^{(N)}$$

$$\ldots$$

$$H_{i}^{(L-1)} = \left(1 - \left(z^{(L-1)}\right)^2 \right) \odot W^{(L-1)} H_{i}^{(L-2)} - 2z^{(L-1)} \odot \left(1 - \left(z^{(L-1)}\right)^2 \right) \odot i^{(L-2)} \odot \left(W^{(L-1)}\right)^2 J^{(L-2)}$$

$$H_{i}^{(L)} = W^{(L)} H_{i}^{(L-1)}$$

$$H_{i}^{(D)} = \frac{1}{\alpha} H_{i}^{(L)}$$  (17)
Finally, a Hessian matrix at the output layer, $L$, is obtained by reshaping $n$ Hessian slices of $H^{(Q)}$, which are row vectors of size $1 \times n$, into an $n \times n$ matrix, as expressed in Eq. (18):

$$
H^{(Q)} = \begin{bmatrix}
H^{(Q)}_{1,1} & \cdots & H^{(Q)}_{1,n} \\
\vdots & \ddots & \vdots \\
H^{(Q)}_{n,1} & \cdots & H^{(Q)}_{n,n}
\end{bmatrix} \quad (18)
$$

The Lagrange optimization problem presented in Section 3.1 can then be solved in a generalizable way using a neural network to generate AI-based objective functions, Jacobian, and Hessian matrices, as shown in Eqs. (10), (13), and (18), respectively.

The concept of AI-based Lagrange Optimization method is explained following three steps as shown in Figure 3.

- Step 1: Structural big datasets are generated from conventional design software, such as AutoCol. A proper number of big datasets needed for training should be selected carefully based on level of complexity of considered problem.

- Step 2: AI-based objective functions are achieved based on neural networks trained on big datasets obtained from Step 1. Accuracies of AI-based model are considerably affected by not only number of big datasets but also neural network parameters, such as number of hidden layers, neurons, and required epochs, etc. Therefore, a proper framework of neural network should be employed to get good training results.

- Step 3: Lagrange multiplier method is applied to optimize AI-based objective functions. The aim of AI-based objective functions is to approximate any well-

### Table 2. Training accuracies based on PTM.

| No. | Data In | Layers | Neurons | Best Epoch | MSE T. Perf | R at Best Epoch |
|-----|---------|--------|---------|------------|-------------|-----------------|
| 1a  | 100,000 | 1      | 80      | 53,763     | 2.9.E-5     | 0.998           |
| 1b  | 100,000 | 2      | 80      | 76,929     | 2.3.E-6     | 1.000           |
| 1c  | 100,000 | 5      | 80      | 57,688     | 7.7.E-7     | 1.000           |
| 2a  | 100,000 | 1      | 80      | 40,253     | 5.8.E-5     | 0.998           |
| 2b  | 100,000 | 2      | 80      | 38,900     | 4.6.E-6     | 1.000           |
| 2c  | 100,000 | 5      | 80      | 34,897     | 2.2.E-6     | 1.000           |
| 3a  | 100,000 | 1      | 80      | 73,206     | 4.1.E-3     | 0.975           |
| 3b  | 100,000 | 2      | 80      | 86,964     | 4.5.E-4     | 0.999           |
| 3c  | 100,000 | 5      | 80      | 117,296    | 1.3.E-4     | 0.999           |
| 4a  | 100,000 | 1      | 80      | 100,000    | 1.8.E-7     | 1.000           |
| 4b  | 100,000 | 2      | 80      | 52,526     | 4.8.E-9     | 1.000           |
| 4c  | 100,000 | 5      | 80      | 160,349    | 2.2.E-9     | 1.000           |
| 5a  | 100,000 | 1      | 80      | 47,339     | 2.0.E-4     | 0.999           |
| 5b  | 100,000 | 2      | 80      | 150,505    | 3.7.E-5     | 0.999           |
| 5c  | 100,000 | 5      | 80      | 45,505     | 2.4.E-5     | 0.999           |

### Table 3. Column design scenarios for optimization.

| Column configuration | Squared column: $b = h$ |
|----------------------|-------------------------|
| Material properties  | Concrete: $f_c = 40$ MPa |
| Factored load        | Axial force: $P_a = 1000$ kN |
| Safety factor        | Bending moment $M_b = 3000$ kN-m |
| Design target        | $SF = 1$ |
|                       | Minimized $C_l$ (KRW/m)  |


Table 4. Summary of equality and inequality constraints.

| Equality  | Inequality         | Constraint(s) | Constraint(s) |
|-----------|--------------------|---------------|---------------|
| \( c_1(x) = b - h = 0 \) | \( v_1(x) = \rho_s - 0.01 \geq 0 \) | \[ b \times h \] | \[ b \times h \] |
| \( c_2(x) = f - 40 = 0 \) | \( v_2(x) = -\rho_s + 0.08 \geq 0 \) | \[ b \times h \] | \[ b \times h \] |
| \( c_3(x) = f_y = 500 = 0 \) | | \[ b \times h \] | \[ b \times h \] |
| \( c_4(x) = P_u = 1000 = 0 \) | | \[ b \times h \] | \[ b \times h \] |
| \( c_5(x) = M_u - 3000 = 0 \) | | \[ b \times h \] | \[ b \times h \] |
| \( c_6(x) = S_F - 1 = 0 \) | | \[ b \times h \] | \[ b \times h \] |

Behaved objective functions and generalize calculation procedure of Jacobian and Hessian matrices, which are needed to solve KKT conditions based on Newton–Raphson method.

4. Application of AI-based Lagrange method on optimizing RC columns

In this study, conventional structural software (AutoCol) is employed to generate big datasets for neural network training. Column configuration \((b \times h)\), rebar ratio \((\rho_s)\), and material properties are conducted to evaluate structural performance of RC column, such as design axial force \((\phi P_n)\), design bending moment \((\phi M_n)\), and rebar strain \((\epsilon_e)\), against factored load pair \((P_u,M_u)\). Figure 4 and Table 1 describe seven inputs \((b,h,\rho_s,f_c,f_y,P_u,M_u)\) needed to calculate nine corresponding output parameters \((\phi P_n,\phi M_n,S_F, b/h, \epsilon_e, C_L, CO_2, W_c, \alpha_c/h)\).

4.1. Formulation of objective function and other parameters based on ANNs

The nine output parameters, including the objective function \((C_L)\), are functions of seven variables that are complex and difficult to not only implement analytically but also find their Jacobian and Hessian matrices for solving the Lagrange function (Eq. (7)). AI-based neural networks are developed to universally approximate all the output parameters, as expressed in Eq. (10), and hence, a unique process of finding their Jacobian and Hessian matrices can be applied. For example, the objective function of column cost index \((C_L)\) is obtained using Eq. (19) based on the given input parameters \((b,h,\rho_s,f_c,f_y,P_u,M_u)\). The equation is forward-network-based weight-bias functions with \(L\) layers and 80 neurons, which are linked using weighted interconnections and bias through an activation function, thereby performing nonlinear numerical computations. An activation function \((\text{tansig}, \text{tanh})\), as shown in Figure 4, is used in Eq. (19).

The input, \(\mathbf{x} = [b, h, \rho_s, f_c, f_y, P_u, M_u]^T\), for networks is related to neurons of fully connected successive layers using weights at each neuron and bias in each hidden layer. The layers are then summed up for the outputs, such as \(C_L\) and \(CO_2\) emissions, and \(W_c\) ((Hong 2019)). The neural networks are formulated to be able to generalize trends (recognized as machine learning) between inputs and outputs to obtain objective functions rather than being based on analytical engineering mechanics or knowledge (Berrais 1999).

\[
\begin{align*}
C_L &= g_{C_L}^{(D)} (f_1^{(L-1)} (W_{C_L}^{(L-1)} (x) + b_1^{(L-1)} )) + b_1^{(L)} \tag{19}
\end{align*}
\]

Similarly, the rest of the outputs \((\phi P_n, \phi M_n, S_F, b/h, \epsilon_e, CO_2, W_c, \alpha_c/h)\) can also be formulated as functions of seven inputs based on forward ANNs, as shown in Table 2. Table 2 present training results of forward networks, in which three types of hidden layer (1, 2, and 5) with 80 neurons are implemented. Structural datasets of 100,000 are randomly divided into three small subsets: training set, validation set, and test set. According to Brian Ripley (Ripley 1996), a training set (70% of big datasets) is a set of examples used for learning, that is to fit parameters, whereas a validation set is used to tune a neural network to avoid overfitting. A test subset, on the other hand, is independent of the training dataset, which does not affect training procedure. It is, therefore, used only to access the performance of a fully specified classifier. Hence, in Table 2, a MSE of test set \((\text{MSE T. Perf})\) is suitable for evaluating the goodness of designs, indicating capability of training model against unseen datasets.

4.2. Optimization of RC column using AI-based Lagrange method

The neural network models of RC column presented in Table 2 are used to minimize cost index \((C_L)\) of a RC column under several design requirements as shown in Table 3. All design requirements can be expressed in term of equality constraints \(\mathbf{c}(\mathbf{x}) = [c_1(\mathbf{x}), \ldots, c_6(\mathbf{x})]^T\) as stated in Table 4. Besides, the rebar ratio, \(\rho_s\), should only be constrained following the ACI-318 code requirements \((\rho_{s,min} \leq \rho_s \leq \rho_{s,max})\), which are expressed in terms of two inequality constraints: \(v_1(x) = \rho_s - \rho_{s,min} \geq 0\) and \(v_2(x) = -\rho_s + \rho_{s,max} \geq 0\) (Table 4).

In Eq. (19), the \(C_L\) function for forward optimization is defined as \(C_L = f_{C_L}^{PW}(\mathbf{x})\), as a function of seven input parameters. According to Eqs. (1) and (3), the Lagrange
optimization function of cost index, $L_{\text{CI}}^{\text{FW}}$, and its KKT conditions are then expressed as functions of the input variables $x = [b, h, \rho_s, f_c, f_y, P_u, M_u]^T$ and Lagrange multiplier of equality and inequality constraints, $\lambda_c = [\lambda_{c1}, \lambda_{c2}, \ldots, \lambda_{cn}]^T$ and $\lambda_v = [\lambda_{v1}, \lambda_{v2}]^T$, respectively, as shown in Eqs. (20) and (21).

\[ L_{\text{CI}}^{\text{FW}}(x, \lambda_c, \lambda_v) = f_{\text{CI}}^{\text{FW}}(x) - \lambda_c^Tc(x) - \lambda_v^TSv(x) \]  \hfill (20)

KKT condition:

\[ \nabla L_{\text{CI}}^{\text{FW}}(x, \lambda_c, \lambda_v) = \begin{bmatrix} \nabla f_{\text{CI}}^{\text{FW}}(x) - J_c(x)^T\lambda_c - J_v(x)^T\lambda_v - c(x) \\ -Sv(x) \end{bmatrix} = 0 \]  \hfill (21)

It is well-known that the Newton–Raphson method relies heavily on a good initial vector assumed as $x^{(0)} = [b, h, \rho_s, f_c, f_y, P_u, M_u]^T$ to expedite the run process as well as enhance accuracy. A good initial vector is predetermined based on simple equality and active inequality; $c_3(x), c_4(x), c_5(x)$ and $v_1(x), v_2(x)$ (Table 4). Inequality constraint ($v_1$) is activated

Initial vector when $v_1$ is activated is obtained from

\[ x^{(0)} = [b, h, \rho_s, f_c, f_y, P_u, M_u]^T \]

where five input variables ($\rho_s, f_c, f_y, P_u, M_u$) are predetermined based on simple equality constraints $c_3(x), c_4(x), c_5(x)$ and active inequality $v_1(x)$. The initial vector $x^{(0)} = [b, h, 0.01, 40, 500, 1000, 3000]^T$ is used to find the saddle point of the Lagrange optimization function, as expressed in Eq. (20), based on the Newton–Raphson method. Unknown input parameters ($b, h$) for the initial vector are random in large datasets which are to be determined during optimization.

The Newton–Raphson method is implemented to solve partially differentiated Eq. (21) using one initial Lagrange multiplier vector, $[\lambda_{c1}, \lambda_{c2}] = [0, 0]$, and 5^2 initial vectors of $x^{(0)} = [b, h, 0.01, 40, 500, 1000, 3000]^T$, in which $b$ and $h$ are randomly distributed within a training data range. The initial values of 0 for the Lagrange multipliers are used because they do not have boundaries; they can be any number while the Newton–Raphson model calculates the exact Lagrange multipliers. $[b, h] = [1070.1, 1070.1]$ is the best value.

### Table 5. Lowest cost index of columns ($C_l$) for given constraints based on forward networks

| Parameters | Case 1 | Case 2 | Case 3 |
|------------|--------|--------|--------|
| 1          | b (mm) | 1022.1 | 629.1  | 980.1  |
| 2          | h (mm) | 1022.1 | 629.1  | 980.1  |
| 3          | $\rho_s$ | 0.0100 | 0.0800 | 0.0118 |
| 4          | $f_c$ (MPa) | 40 | 40 | 40 |
| 5          | $f_y$ (MPa) | 500 | 500 | 500 |
| 6          | $P_u$ (kN) | 1000 | 1000 | 1000 |
| 7          | $M_u$ (kN·m) | 3000 | 3000 | 3000 |
| **Objective:** $C_l$ (KRW/m) | 185,999.2 | 299,994.8 | 184,684.6 |

(b) Based on forward training with 2 layers – 80 neurons

| Parameters | Case 1 | Case 2 | Case 3 |
|------------|--------|--------|--------|
| 1          | b (mm) | 1068.5 | 615.5  | 924.8  |
| 2          | h (mm) | 1068.5 | 615.5  | 924.8  |
| 3          | $\rho_s$ | 0.0100 | 0.0800 | 0.0162 |
| 4          | $f_c$ (MPa) | 40 | 40 | 40 |
| 5          | $f_y$ (MPa) | 500 | 500 | 500 |
| 6          | $P_u$ (kN) | 1000 | 1000 | 1000 |
| 7          | $M_u$ (kN·m) | 3000 | 3000 | 3000 |
| **Objective:** $C_l$ (KRW/m) | 201,942.3 | 286,645.1 | 195,507.9 |

(c) Based on forward training with 5 layers – 80 neurons

| Parameters | Case 1 | Case 2 | Case 3 |
|------------|--------|--------|--------|
| 1          | b (mm) | 1070.1 | 619.1  | 919.8  |
| 2          | h (mm) | 1070.1 | 619.1  | 919.8  |
| 3          | $\rho_s$ | 0.0100 | 0.0800 | 0.0166 |
| 4          | $f_c$ (MPa) | 40 | 40 | 40 |
| 5          | $f_y$ (MPa) | 500 | 500 | 500 |
| 6          | $P_u$ (kN) | 1000 | 1000 | 1000 |
| 7          | $M_u$ (kN·m) | 3000 | 3000 | 3000 |
| **Objective:** $C_l$ (KRW/m) | 202,275.4 | 289,955.2 | 195,671.1 |

**Note**

- Case 1 – Inequality $v_1(x)$ is active
- Case 2 – Inequality $v_2(x)$ is active
- Case 3 – None of inequalities are active
among 25 trials based on the network with five hidden layers and 80 neurons, producing an optimal value of 202,275.4 when inequality (\(v_1\)) is activated, as shown in Table 5(c). The optimized results based on one and two hidden layers and 80 neurons are also shown in Table 5(a) and (b) with optimal values of \(C_l = 185,069.2\), and 201,942.3, respectively. Similarly, the optimized results of case 2 (inequality constraint (\(v_2\)) is activated) and case 3 (none is activated) are obtained (Table 5).

The optimized design results obtained using the Lagrange multipliers based on forward networks trained with one, two, and five layers and 80 neurons are listed in Table 5(a), (b), and (c), respectively, and are compared with those obtained using a structural software (AutoCol) (Table 6(a), (b), and (c)). In Table 6(a), the largest error of 11.79% of \(SF\) was demonstrated with one layer and 80 neurons. With two and five layers and 80 neurons (Table 6(b) and (c)), reduced errors of
1.55% and 2.7% for the design moment ($\phi M_n$) and design axial force ($\phi P_n$) were obtained, respectively, by LMM based on forward networks.

4.3. Verification of $\text{Cl}_c$ by large datasets

The goodness of optimal designs presented in Table 5 is evaluated by large datasets as shown in Figure 5. The lowest $\text{Cl}_c$ for large datasets is 195,716, where five million datasets are filtered through $f' = 40 \text{MPa}$, $f_y = 500 \text{MPa}$, $SF = 1$, and $b/h = 1$. The accuracy is demonstrated with $\text{Cl}_c$ of 184,684.6 (−5.64%; one layer, Table 6(a)), 195,507.9 (−0.11%; two layers, Table 6(b)), and 195,671.1 (−0.02%; five layers, Table 6(c)) based on AI-based network compared with the optimal $\text{Cl}_c$ (195,716) of large datasets. The $\text{Cl}_c$ obtained by the AI-based Lagrange method with two and five layers and that obtained in large datasets are in good agreement.

4.4. P–M diagram

The optimal $\text{Cl}_c$-based P–M interaction diagrams for RC columns that satisfy various design criteria (Table 4.1(b) and Table 4.2) are plotted on the basis of forward networks, as shown in Figure 6. The P–M diagrams pass through $P_u$ and $M_u$, as indicated by a solid point. In Figure 6, the P–M diagrams indicated by Legends 1, 2, and 3 were constructed with the parameters shown in the dashed black box in Table 6(a), (b), and (c), respectively, which were obtained by AI-based Lagrange optimization. These were used to construct P–M diagrams since the
accuracies of the parameters obtained by ANN are acceptable. The $P$-$M$ diagrams were plotted using AutoCol with appropriate input parameters. The interaction diagrams shown in Figure 6 meet the minimum $C_l$, as listed in Table 6. $C_l$ was optimized by Lagrange multiplier-based forward neural networks with one, two, and five layers and 80 neurons. The dotted curve cannot pass through $P_{\text{co}}$ and $M_x$ (solid point), indicating that the training accuracy of a model with one layer is not sufficient. All three optimized $P$-$M$ diagrams (Legends 1, 2, and 3) shown in Figure 6 would converge, passing through one $P$-$M$ diagram when the training accuracies are sufficient.

5. Conclusions
This study presents hybrid optimization techniques, with which objective functions are proposed based on ANNs. Lagrange optimization techniques with constraints were implemented to achieve rational engineering decisions and find minimized or maximized design values by solving nonlinear optimization problems under strict constraints, conditions, and requirements imposed by design codes for the design of structural frames such as columns and beams. This study helps engineers make the final design decisions, not on the engineers’ empirical observations but on more rational designs, to meet various design requirements, including code restrictions and/or architectural criteria while the objective parameters, such as cost index ($C_l$), CO$_2$ emissions, and structural weight ($W_c$), are optimized. The conclusions drawn from the study are as follows:

1. Constructing objective functions and their derivatives, which is a challenge in the Lagrange multiplier method, can now be generalized to any structural design optimization by using ANN to generate universal AI-based objective functions, Jacobian, and Hessian matrices.

2. Automatic designs of structural frames are proposed for realistic engineering applications to identify design solutions that optimize all design requirements simultaneously, thereby achieving design decisions not based on engineers’ intuition. The results of the sensitivity analysis of LMM show that the best optimal values based on constraints can be identified for specific situations such as official design codes.

3. Analytical objective functions were difficult to obtain. This study developed the objective functions for $C_l$, CO$_2$ emissions, and $W_c$ based on AI-based networks to be implemented for LMM to find the optimized solutions.

4. The Karush–Kuhn–Tucker conditions were considered to account for inequality constraints, leading to automatic designs of structural frames that meet various code restrictions simultaneously.

5. Optimization process for $C_l$ was performed, and negligible errors were obtained, which were verified using large structural datasets. Engineering calculations also validated the design accuracies when the optimized $C_l$ was implemented in the designs.

6. $P$–$M$ diagrams were uniquely designed to optimize columns. The proposed optimization will offer generic designs for many types of structures, including machinery and structural frames.

7. The AI-based objective functions developed in this study can be implemented in broad areas, including engineering, general science, and economics.

A generalizable optimization method proposed herein can be applied to any optimization problem once a sufficient number of data can be collected for establishing approximated objective functions and other parameters to formulate ANN. New objective functions not only enhance the computational speed compared to conventional software but also produce a generalizable calculation method for Jacobian and Hessian matrices for Lagrange optimization. In future work, comprehensive design optimization of a dynamic design of tall buildings would be performed based on AI-based Lagrange optimization. One concerning problem is that computational time of Lagrange optimization is heavily dependent on quantity of inequality constraints because a number of running case (active inequality) corresponds to combinations of inequality constraints. Likewise, a building design is composed of many design requirements considered as inequality constraints, such as lateral displacement, story drift, design strength of each component, and required total base shears, etc.

Acknowledgments
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT 2019R1A2C2004965).

Disclosure statement
No potential conflict of interest was reported by the author.

Funding
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government [MSIT 2019R1A2C2004965].

Notes on contributors
Dr. Won-Kee Hong is a Professor of Architectural Engineering at Kyung Hee University. Dr. Hong received his Masters and Ph.D. degrees from UCLA, and he worked for Englekirk and Hart, Inc. (USA), Nihon Sekkei (Japan) and Samsung Engineering and Construction Company (Korea) before joining Kyung Hee University (Korea). He also has professional
enrolled technologies. Camp, Barros, ACI-318.

10.1016/j.conbuildmat.2015.09.001
10.1007/
10.12989/scs.2018.29.2.243
10.12989/scs.2015.18.2.305

Engineering Master Manh Systems: of developing approaches engineering
Cylinder”.

129:1(105)
Design Sciences Engineering: Concrete Construction Regime Reinforced
2014 Subjected Assessment

Fuel engineering. Mn., and I. Goodfellow, and A. Courville. 2017. Deep Learning. Vol. I. Massachusetts, USA: MIT press.

Berrais, A. 1999. “Artificial Neural Networks in Structural Engineering: Concept and Applications.” Engineering Sciences 12: 1.

Camp, C. V., S. Pezeshk, and H. Hansson. 2003. “Flexural Design of Reinforced Concrete Frames Using a Genetic Algorithm.” Journal of Structural Engineering 129 (1): 105–115. doi:10.1061/(ASCE)0733-9445(2003)129:1(105).

Fanaie, N., S. Aghajani, and S. Shamloo. 2012. “Theoretical Assessment of Wire Rope Bracing System with Soft Central Cylinder”. Proceedings of the 15th World Conference on Earthquake Engineering, Advances in Structural Engineering

Fanaie, N., S. Aghajani, and E. A. Dizaj. 2016. “Theoretical Assessment of the Behaviour of Cable Bracing System with Central Steel Cylinder.” Advances in Structural Engineering 19 (3): 463–472. doi:10.1177/1369433216660052.

Heydari, A., and M. Shariati. 2018. “Buckling Analysis of Tapered BDFGM Nano-beam under Variable Axial Compression Resting on Elastic Medium.” Structural Engineering & Mechanics 66 (6): 737–748. doi:10.12989/sem.2018.66.6.737.

Hoffmann, L. D., and G. L. Bradley. 2004. Calculus for Business, Economics, and the Social and Life Sciences.8th. 575–588. ISBN 0-07-242432–X. New York: McGraw Hill Education

Hong, W. K. 2019. Hybrid Composite Precast Systems: Numerical Investigation to Construction, 427–478. Kidlington, OX5 1GB, United Kingdom: Woodhead Publishing, Elsevier

Kalman, D. 2009. “Leveling with Lagrange: An Alternate View of Constrained Optimization.” Mathematics Magazine 82 (3): 186–196. doi:10.1080/0025570X.2009.11953617.

Kaveh, A., and F. Shokohi. 2015. “Optimum Design of Laterally Supported Castellated Beams Using CBO Algorithm.” Steel & Composite Structures 18 (2): 305–324. doi:10.12989/scs.v2015.i2.305.

Korouzdeh, T., H. Eskandari-Naddaf, and M. Gharouni-Nik. 2017. “An Improved Ant Colony Model for Cost Optimization of Composite Beams.” Applied Artificial Intelligence 31 (1): 44–63. doi:10.1080/08839514.2017.1296681.

Madadi, A., H. Eskandari-Naddaf, R. Shadnia, and L. Zhang. 2018. “Characterization of Ferrocement Slab Panels Containing Lightweight Expanded Clay Aggregate Using Digital Image Correlation Technique.” Construction and Building Materials 180: 464–476. doi:10.1016/j.conbuildmat.2018.06.024.

MATLAB. 2020b. Version 9.9.0 (R2020b). Natick, Massachusetts: MathWorks.

Nasrollahi, S., S. Maleki, M. Shariati, A. Marto, and M. Khormai. 2018. “Investigation of Pipe Shear Connectors Using Push Out Test.” Steel & Composite Structures 27 (5): 537–543. doi:10.12989/scs.2018.27.5.537.

Paknahad, M., M. Bazzaz, and M. Khorami. 2018. “Shear Capacity Equation for Channel Shear Connectors in Steel-concrete Composite Beams.” Steel & Composite Structures 28 (4): 483–494. doi:10.12989/scs.2018.28.4.483.

Petkovic, M., and D. V. Morrey Jr. 1985. Intermediate Calculus. 2nd. 267. New York: Springer.ISBN 0-387-96058-9.

Rajeve, S., and C. S. Krishnamoorthy. 1998. “Genetic Algorithm–based Methodology for Design Optimization of Reinforced Concrete Frames.” Computer-Aided Civil and Infrastructure Engineering 13 (1): 63–74. doi:10.1111/0885-9507.00086.

Ripley, B. D. “Pattern Classification and Neural Networks.” 1996.

Safa, M., M. Shariati, Z. Ibrahim, A. Toğrul, S. B. Baharom, N. M. Nor, and D. Petkovic. 2016. “Potential of Adaptive Neuro Fuzzy Inference System for Evaluating the Factors Affecting Steel-concrete Composite Beam’s Shear Strength.” Steel & Composite Structures 21 (3): 679–688. doi:10.12989/scs.2016.21.3.679.

Shah, S. N. R., N. R. Sulong, R. Khan, M. Z. Jumaat, and M. Shariati. 2016. “Behavior of Industrial Steel Rack Connections.” Mechanical Systems and Signal Processing 70-71: 25–740. doi:10.12989/scs.2016.21.3.679.

Shariati, M., M. Shariati, A. Mscjadi, and K. Waki1. 2018. “Computational Lagrangian Multiplier Method by Using for Optimization and Sensitivity Analysis of Rectangular Reinforced Concrete Beams.” Steel & Composite Structures 29 (2): 243–256. doi:10.12989/scs.2018.29.2.243.

References

Abedini, M., and C. Zhang. 2021. “Dynamic Vulnerability Assessment and Damage Prediction of RC Columns Subjected to Severe Impulsive Loading.” Structural Engineering and Mechanics 77 (4): 441–461. doi:10.12989/sem.2021.77.4.441.

ACI-318. “Building Code Requirements for Reinforced.” 2014.

Aghaee, K., M. A. Yazdi, and K. D. Tsavdaridis. 2014. “Mechanical Properties of Structural Lightweight Concrete Reinforced with Waste Steel Wires.” Magazine of Concrete Research 66 (1): 1–9. doi:10.1680/macr.14.00232.

Awal, A. A., I. A. Shehu, and M. Ismail. 2015. “Effect of Cooling Regime on the Residual Performance of High-volume Palm Oil Fuel Ash Concrete Exposed to High Temperatures.” Construction and Building Materials 98: 875–883. doi:10.1016/j.conbuildmat.2015.09.001.

Barros, M. H. F. M., R. A. F. Martins, and A. F. M. Barros. 2005. “Cost Optimization of Singly and Doubly Reinforced Concrete Beams with EC2-2001.” Structural and Multidisciplinary Optimization 30 (3): 236–242. doi:10.1007/s00158-005-0156-2.

Beavis, B., and I. Dobbs. 1990. Optimisation and stability theory for economic analysis .Cambridge University press.

Bengio, Y., I. Goodfellow, and A. Courville. 2017. Deep Learning. Vol. I. Massachusetts, USA: MIT press.

Berrais, A. 1999. “Artificial Neural Networks in Structural Engineering: Concept and Applications.” Engineering Sciences 12: 1.
Shariati, M., S. N. H. Ramli, S. Maleki, and K. M. M. Arabnejad. “Experimental and Analytical Study on Channel Shear Connectors in Light Weight Aggregate Concrete.” Proceedings of the 4th International Conference on Steel & Composite Structures. Sydney, Australia. 2010; 21–23. 10.3 850/978-981-08-6218-3 CC-fr031.

Silberberg, E., and S. Wing. 2001. The Structure of Economics: A Mathematical Analysis. Third. 134–141. Boston: Irwin McGraw-Hill.ISBN 0-07-234352-4

Sun, L., Z. Yang, Q. Jin, and W. Yan. 2020. “Effect of Axial Compression Ratio on Seismic Behavior of GFRP Reinforced Concrete Columns.” International Journal of Structural Stability and Dynamics 20 (6): 2040004. doi:10.1142/S0219455420400004.

Toghroli, A., M. Mohammad hassani, M. Suhatril, M. Shariati, and Z. Ibrahim. 2014. “Prediction of Shear Capacity of Channel Shear Connectors Using the ANFIS Model.” Steel & Composite Structures 17 (5): 623–639. doi:10.12989/scs.2014.17.5.623.

Villarrubia, G., J. F. De Paz, P. Chamoso, and F. De la Prieta. 2018. “Artificial Neural Networks Used in Optimization Problems.” Neurocomputing 272: 10–16. doi:10.1016/j.neucom.2017.04.075.

Walsh, G. R. 1975. Saddle-point Property of Lagrangian Function. Methods of Optimization. 39–44. New York:John Wiley & Sons. ISBN 0-471-91922-5.

Zazdani, M., A. Fakhimi, and M. Alitalesh (2018). “Numerical Analysis of Effective Parameters in Direct Shear Test by Hybrid Discrete–finite Element Method.”

Ye, M., J. Jiang, H. M. Chen, H. Y. Zhou, and D. D. Song. 2021. “Seismic Behavior of an Innovative Hybrid Beam-column Connection for Precast Concrete Structures.” Engineering Structures 227: 111436. doi:10.1016/j.engstruct.2020.111436.

Zhang, C., G. Gholipour, and A. A. Mousavi. 2021. “State-of-the-Art Review on Responses of RC Structures Subjected to Lateral Impact Loads.” Arch Computat Methods Eng 28: 2477–2507. doi:10.1007/s11831-020-09467-5.

Zhang, C., Z. Alam, L. Sun, Z. Su, and B. Samali. 2019. “Fibre Bragg Grating Sensor-based Damage Response Monitoring of an Asymmetric Reinforced Concrete Shear Wall Structure Subjected to Progressive Seismic Loads.” Structural Control & Health Monitoring 26 (3): e2307. doi:10.1002/stc.2307.