A Gentle Introduction to Isogeometric Analysis  
Part 3: Elastostatic Analysis for Euler-Bernoulli Beam

Keiji Yanase**

Based on the previous studies, in the present study, we embark on the numerical simulation for structural mechanics. In specific, the deflection behavior of Euler-Bernoulli beam is simulated with B-spline basis functions. Starting from the fundamental theory, we make use of the theorem of minimum potential energy to formulate the system equation. By comparing to a classical approach with Hermite polynomials, unique characteristics rendered by B-spline are highlighted and discussed.

Key Words: B-spline basis function, Finite element method, Hermite polynomial, MATLAB

1. Simply-supported Beam on Elastic Foundation

Euler-Bernoulli beam theory is a simplification of the linear theory of elasticity and it provides a means of calculating the load-carrying and deflection characteristics of beams. It covers the case for small deflection of a beam that is subjected to lateral loads.

Here, let us recall a brief history of Euler-Bernoulli beam theory. During the last quarter of 17th and the beginning of 18th centuries, a rapid development of the infinitesimal calculus took place. Started by Leibnitz (1646-1716), it progressed principally by the work of Jacob and John Bernoulli. In trying to expand the field of application of this new mathematical tool, they discussed several examples from mechanics and physics. One such example treated by Jacob Bernoulli (1654-1707, Fig. 1) concerned the shape of the deflection curve of an elastic bar, and in this way he began an important chapter in the mechanics of elastic bodies. The general form formulated by Jacob Bernoulli was used later on by other mathematicians (principally by Euler (1707-1783, Fig. 2)) in their investigations of elastic curves [1].

Fig. 1. Jacob Bernoulli (1654-1707).

Fig. 2. Leonhard Euler (1707-1783).

Fig. 3. Deflection of beam.
Euler-Bernoulli beam theory stipulates that when a beam deflects, the plane sections, which are originally perpendicular to the axis of beam, remain plane and perpendicular to the deformed axis, as shown by Fig. 1. Based on Fig. 1, the change in length of $dx$ at distance $y$ from the neutral plane is calculated as:

$$dl(x) = \frac{dw(x)}{dx} \times y - \left\{ \frac{dw(x)}{dx} + \frac{d}{dx} \left( \frac{dw(x)}{dx} \right) \right\} \times dx \times y$$

$$= -\frac{d}{dx} \left( \frac{dw(x)}{dx} \right) \times dx \times y$$

where $w$ is the deflection of beam. Thus, the engineering strain is calculated as:

$$\varepsilon(x) = \frac{dl(x)}{dx} = -\frac{d^2w(x)}{dx^2} \times y$$

The corresponding stress is:

$$\sigma(x) = E \varepsilon(x) = -\frac{E d^2w(x)}{dx^2} \times y$$

where $E$ is Young’s modulus. Finally, the internal moment is calculated as follows:

$$M(x) = \int (\sigma(x) \times y) dA = -E I \frac{d^3w(x)}{dx^3}$$

where the moment of inertia, $I$, is rendered as:

$$I = \int y^2 dA$$

Here, $A$ signifies the cross sectional area of beam.

Let us consider a beam on elastic foundation \[2\], as shown in Fig. 4. The stored energy in the beam and the foundation, $U$, is rendered as:

$$U = \int_{x_1}^{x_2} \frac{M^2(x)}{2EI} dx + \frac{1}{2} \int_{x_1}^{x_2} F(x)w(x) dx$$

where, $M(x)$ is the internal moment in beam given by Eq. (4) and $F(x)$ is the internal force in spring given as:

$$F(x) = k w(x)$$

where, $k$ is the spring constant. By substituting Eqs. (4) and (7) into Eq. (6), we obtain the following equation:

$$U = \int_{x_1}^{x_2} \frac{EI}{2} \left( \frac{d^4w(x)}{dx^4} \right) dx + \frac{1}{2} \int_{x_1}^{x_2} kw(x) dx$$

On the other hand, the external virtual work is rendered as:

$$W = \int_{x_1}^{x_2} q(x)w(x) dx$$

Based on Eqs. (8) and (9), the potential energy, $\Pi$, is rendered as:

$$\Pi = U - W$$

$$= \frac{EI}{2} \left( \frac{d^4w(x)}{dx^4} \right) dx + \frac{1}{2} \int_{x_1}^{x_2} kw(x) dx - \int_{x_1}^{x_2} q(x)w(x) dx$$

2. Approximation with Hermite Polynomials

Here, let us consider a classical approach with finite elements. By using the cubic Hermite polynomials [2], the deflection of an element, $w'$, can be approximated as follows:

$$w' = H_1(u) \delta'_1 + H_2(u) \delta'_2 + H_3(u) \delta'_3 + H_4(u) \delta'_4$$

where:

$$H_1(u) = 2u^3 - 3u^2 + 1 \quad ; \quad H_2(u) = L'(u^3 - 2u^2 + u)$$

$$H_3(u) = -2u^3 + 3u^2 \quad ; \quad H_4(u) = L'(u^3 - u^2)$$

Here, $L'$ is the length of an element and $u$ is the parametric point $(0 \leq u \leq 1)$. Further, $\delta'$ and $\theta'$ are the end-deflections and the end-slopes of an element, as shown by Fig. 5. For simplicity, we rewrite Eq. (11) as:

$$w' = \sum_{i=1}^{4} H_i(u) w'_i$$

Thus:
By substituting Eqs. (13) and (14) into Eq. (10), we obtain the following equation:

\[
\Pi = \sum_{j=1}^{\infty} \int_{e} \frac{1}{2} E I \left( \frac{d^2 H_j(u)}{dx^2} \right) \left( \frac{d^2 H_j(u)}{dx^2} \right) w_j' dx + \sum_{j=1}^{\infty} \int_{e} k \left( \frac{d H_j(u)}{dx} \right) \left( \frac{d H_j(u)}{dx} \right) w_j' dx - \sum_{j=1}^{\infty} \int_{e} q(x) \left( \frac{d H_j(u)}{dx} \right) w_j' dx
\]

where \( NE \) signifies the number of elements.

In practice, the theorem of minimum potential energy renders the solution for elastostatic problem. Then, the minimization of Eq. (15) with the parameter \( w_j' \) renders the following equation:

\[
\frac{\partial \Pi}{\partial w_j'} = 0
\]

\[
= \sum_{j=1}^{\infty} \int_{e} \frac{1}{2} E I \left( \frac{d^2 H_j(u)}{dx^2} \right) \left( \frac{d^2 H_j(u)}{dx^2} \right) w_j' dx + \sum_{j=1}^{\infty} \int_{e} k \left( \frac{d H_j(u)}{dx} \right) \left( \frac{d H_j(u)}{dx} \right) w_j' dx - \sum_{j=1}^{\infty} \int_{e} q(x) H_j(u) w_j' dx
\]

By considering the connectivity of nodes, Eq. (16) can be expressed as:

\[
\sum_{j=1}^{\infty} K_{ij} w_j = F_i
\]

(17)

where:

\[
K_{ij} = \sum_{j=1}^{\infty} \int_{e} \frac{1}{2} E I \left( \frac{d^2 H_j(u)}{dx^2} \right) \left( \frac{d^2 H_j(u)}{dx^2} \right) + k \frac{d H_j(u)}{dx} \frac{d H_j(u)}{dx} dx
\]

\[
F_i = \sum_{j=1}^{\infty} \int_{e} q(x) H_j(u) dx
\]

(18)

Here, \( NP \) is the number of nodal points.

To evaluate the derivatives with respect to \( x \), we make use of the chain rule:

\[
\frac{d H_j(u)}{dx} = \frac{d H_j(u)}{du} \frac{du}{dx}
\]

(20)

Finally, by taking advantage of Gauss integration, we write:

\[
\sum_{\xi}^{} \int_{e} \frac{1}{2} E I \left( \frac{d^2 H_j(u)}{dx^2} \right) \left( \frac{d^2 H_j(u)}{dx^2} \right) + k \frac{d H_j(u)}{dx} \frac{d H_j(u)}{dx} dx \]

\[
= \sum_{\xi}^{} K_{ij} \left( \int_{e} q(x) H_j(u) dx \right)
\]

(21)

Here, let us consider the following geometrical description for elements (cf. Fig. 5):

\[
x(\xi) = \left( 1 - \frac{\xi}{2} \right) \xi' + \left( 1 + \frac{\xi}{2} \right) \xi'
\]

(22)

where \(-1 \leq \xi \leq 1\). Further, we link \( u (0 \leq u \leq 1) \) to \( \xi (-1 \leq \xi \leq 1) \) as follows:

\[
u(\xi) = \frac{1}{2} \xi + 1
\]

(23)

Thus:

\[
\frac{du}{dx} = \frac{d u}{d \xi} \frac{d \xi}{dx} = \frac{1}{L'}
\]

(24)

Accordingly for each element, Eqs. (18) and (19) can be evaluated with \( \xi \) coordinate as follows:

\[
K_i = \int_{\xi} \left( \frac{d^2 \xi(u)}{d \xi^2} \right) \left( \frac{d^2 \xi(u)}{d \xi^2} \right) - k \xi(u) \xi(u) \left( \frac{dx}{d \xi} \right) d \xi
\]

(25)

\[
F_i = \int_{\xi} \left( q(\xi) \xi(\xi) \right) \left( \frac{dx}{d \xi} \right) d \xi
\]

(26)

Finally, by taking advantage of Gauss integration, we write:

\[
K_i' \equiv \sum_{\xi} \int_{e} \frac{1}{2} E I \left( \frac{d^2 H_j(\xi)}{dx^2} \right) \left( \frac{d^2 H_j(\xi)}{dx^2} \right) + k H_j(\xi) H_j(\xi) \left( \frac{dx}{d \xi} \right) \left( \frac{dx}{d \xi} \right) W_s
\]

(27)

\[
F_i' = \sum_{\xi} \int_{e} q(\xi) H_j(\xi) \left( \frac{dx}{d \xi} \right) \left( \frac{dx}{d \xi} \right) W_s
\]

(28)

where \( \xi \) is the Gauss point (sampling point), \( W_s \) is the corresponding weight and \( NG \) is the number of Gauss points.

2.1. MATLAB Code with Hermite Polynomials

The following program is created to compute the deflection and the internal moment for Fig. 4. It is noted that for simplification, we set:

\[
EI = 1 \; ; \; \; L = 2 \; ; \; \; k = 3
\]

(29)

Further, the following form is considered for the distributed force:
\[ q(x) = \left( \frac{x}{L} \right)^p \left( 1 - \frac{x}{L} \right) \]  

(30)

---

```matlab
% Simulate the beam problem with Hermite polynomial
% Gaussian point and weight
Gpt = [1, 0, 0, 0, 0, 0; 0.577350, 0.577350, 0, 0, 0; 0.774597, 0.774597, 0, 0; 0.866136, -0.339981, 0.339981, 0.866136]; % Gauss point
Gw = [2.000000, 0.000000, 0.000000, 0.000000; 1.000000, 1.000000, 0.000000, 0.000000; 0.555556, 0.888889, 0.555556, 0.000000; 0.347855, 0.652145, 0.652145, 0.347855]; % Weight

% Parameters
EI = 1; % Rigidity of beam
L = 2; % Length of beam
ks = 3; % Spring constant
Ng = 4; % Number of Gauss points
Ne = 32; % Number of elements
Fi = zeros(2*Ng+2,1); % Force vector
Kij = zeros(Ne,Ng+2); % Stiffness matrix
xcoor = linspace(0,L,Ne+1); % x-coordinate
Le = xcoor(2) - xcoor(1); % Element length

for l = 1:Ne % Element loop
    dKij = zeros(4,4);
    dFl = zeros(4,1);

    for K = 1:Ng % Gauss integration loop
        xi = Gpt(Ng,K); % Gauss point in xi
        u = 1/2*(xcoor(l)+xcoor(l+1)); % Transformation from x to u
        x = (1-xi)/(2*xcoor(l-1)+1)*(1-xi)/(2*xcoor(l+1)+1); % x
        q = x*Le*L*(1-xe/l); % Distributed force

    % Hermite polynomial
    He = [2*u^3-3*u^2+1,Le*(u^3-2*u^2+1),2*u^3-3*u^2,Le*u^3-2*u^2];
    dHe2x = [12*u^3, Le*(6*u^2), -12*u^2, Le*(6*u-2)];

    % Calculate the deflection and moment
    eps = 10^-10;
    u = linspace(eps,1-eps,11); % Sampling point within an element
    Def = zeros(1,length(u)*Ne); % Deflection
    Mom = zeros(1,length(u)*Ne); % Moment
    xp = zeros(1,length(u)*Ne); % Sampling point

    K = 1; % Initial index
    for l = 1:Ne % Element loop
        % Sampling point loop
        v = u(l);
        % Hermite polynomial
        He = [2*v^3-3*v^2+1,Le*(v^3-2*v^2+1),2*v^3-3*v^2,Le*v^3-2*v^2];
        dHe2x = [12*v^3, Le*(6*v^2), -12*v^2, Le*(6*v-2)];

        % Field variables
        xpK = (1-u(l))*xcoor(l) + u(l)*xcoor(l+1); % Sampling point
        DefK = He'*wso(l)*xcoor(l+1); % Deflection
        MomK = dHe*2*xcoor(l)*xcoor(l+1); % Moment

        K = K + 1; % Update for index
    end; % End of "Sampling-point" loop
end; % End of "Element" loop

% Plot the results
FN = 'Fontname'; FS = 'Fontsize'; FW = 'Fontweight';
plot(xp,Def); % Plot for deflection
set(gca,'YDir','reverse'); % Change the direction of y
set(gca,'Fontsize',18,'Xminortick','on', 'Yminortick','on');
xlabel('Coordinate', x, 'FN', 'Callibri', FS, 28, FW, 'bold')
ylabel('Deflection, w(x)', FN, 'Callibri', FS, 28, FW, 'bold')
axis square;

figure; % Plot for moment
plot(xp,Mom); % Plot for deflection
set(gca,'YDir','reverse'); % Change the direction of y
set(gca,'Fontsize',18,'Xminortick','on', 'Yminortick','on');
xlabel('Coordinate', x, 'FN', 'Callibri', FS, 28, FW, 'bold')
ylabel('Moment, M(x)', FN, 'Callibri', FS, 28, FW, 'bold')
axis square;
```

---

(4)
3. Approximation with B-spline Basis Function

By using the B-spline basis functions, $N_{i,p}(u)$, we approximate the deflection of beam as follows [2][3][4]:

$$w(u) = \sum_{i=1}^{NC} N_{i,p}(u) w_i$$  \hspace{1cm} (31)

where $u$ is the parametric point ($0 \leq u < 1$) and $NC$ is the number of control points. A substitution of Eq. (31) into Eq. (10) yields the following equation:

$$\Pi = \int_0^1 \frac{EI}{k} \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) w_i \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) dx$$

$$+ \int_0^1 \frac{k}{2} \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) dx - \int_0^1 q(x) \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) dx$$ \hspace{1cm} (32)

Then, the minimization of Eq. (32) with respect to a parameter $w_i$ renders the following equation:

$$\frac{\partial \Pi}{\partial w_i} = 0 = \int_0^1 \frac{EI}{k} \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) w_i \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) dx$$

$$+ \int_0^1 \frac{k}{2} \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) dx - \int_0^1 q(x) \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) dx$$ \hspace{1cm} (33)

For simplicity, Eq. (33) can be rewritten as:

$$\sum_{i=1}^{NC} K_{i} w_i = F_i$$ \hspace{1cm} (34)

where:

$$K_i = \int_0^1 \frac{EI}{k} \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(u)}{\partial x^2} \right) dx$$

$$+ \int_0^1 \frac{k}{2} \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) dx - \int_0^1 q(x) \left( \sum_{i=1}^{NC} N_{i,p}(u) \right) dx$$ \hspace{1cm} (35)

$$F_i = \int_0^1 q(x)N_{i,p}(u)dx$$ \hspace{1cm} (36)

It is noted that in contrast to Eq. (17), the slope $\theta$ (Fig. 5) is not considered in the system equation.

To evaluate the derivatives of $w$ we make use of the chain rule:

$$\frac{\partial N_{i,p}(u)}{\partial x} = \frac{\partial N_{i,p}(u)}{\partial u} \frac{\partial u}{\partial x}$$

$$\frac{\partial^2 N_{i,p}(u)}{\partial x^2} = \frac{\partial}{\partial x} \left( \frac{\partial N_{i,p}(u)}{\partial u} \frac{\partial u}{\partial x} \right)$$

$$= \left( \frac{\partial^2 N_{i,p}(u)}{\partial u^2} \right) \frac{\partial u}{\partial x} + \left( \frac{\partial N_{i,p}(u)}{\partial u} \right) \frac{\partial^2 u}{\partial x^2}$$ \hspace{1cm} (37)

$$= \frac{\partial^2 N_{i,p}(u)}{\partial u^2} \left( \frac{\partial u}{\partial x} \right)^2 + \frac{\partial N_{i,p}(u)}{\partial u} \frac{\partial u}{\partial x} \frac{\partial^2 u}{\partial x^2}$$ \hspace{1cm} (38)

Here, let us consider the following description of geometry:

$$x(u) = \sum_{i=1}^{NC} N_{i,1}(u) x_i$$ \hspace{1cm} (39)

It is noted that the basis functions, $N_{i,p}(u)$, is independent of the ones used for the approximation of the unknown [2]. For simplicity, based on Eqs. (23) and (39) let us consider:

$$x = N_{i,1}(u)x_i + N_{i,2}(u)x_2 = (1-u)0 + (u)L = \frac{1}{2}(1+\xi)L$$ \hspace{1cm} (40)

Therefore, the transformation from $x$ to $u$ is rendered as:

$$u = x / L$$ \hspace{1cm} (41)

Then, the following relations are obtained:

$$\frac{du}{dx} = \frac{1}{L} ; \quad \frac{dx}{d\xi} = \frac{L}{2}$$ \hspace{1cm} (42)

Further, we adopt Eq. (23) and we can evaluate Eqs. (35) and (36) with $\xi$-coordinate as follows:

$$K_{\xi} = \int_0^1 \left( \frac{EI}{k} \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(\xi)}{\partial \xi^2} \right) \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(\xi)}{\partial \xi^2} \right) \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(\xi)}{\partial \xi^2} \right) \right) d\xi$$

$$F_{\xi} = \int_0^1 q(\xi)N_{i,p}(\xi) \left( \frac{dx}{d\xi} \right) d\xi$$ \hspace{1cm} (43)

$$F_{\xi} = \sum_{i=1}^{NC} \int_0^1 q(\xi)N_{i,p}(\xi) \left( \frac{dx}{d\xi} \right) d\xi$$ \hspace{1cm} (44)

By taking advantage of Gauss integration, we rewrite Eqs. (43) and (44) as follows:

$$K_{\xi} \equiv \sum_{i=1}^{NC} \left( \frac{EI}{k} \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(\xi)}{\partial \xi^2} \right) \left( \sum_{i=1}^{NC} \frac{\partial^2 N_{i,p}(\xi)}{\partial \xi^2} \right) \right) W_{\xi}$$

$$F_{\xi} = \sum_{i=1}^{NC} q(\xi)N_{i,p}(\xi) \left( \frac{dx}{d\xi} \right) W_{\xi}$$ \hspace{1cm} (45)

$$F_{\xi} = \sum_{i=1}^{NC} q(\xi)N_{i,p}(\xi) \left( \frac{dx}{d\xi} \right) W_{\xi}$$ \hspace{1cm} (46)

Compared to Eqs. (18) and (19), it is obvious that Eqs. (43) and (44) involve no discretizations (i.e., it is mesh-free).
3.1. Sub-regions

Since B-spline basis functions are zero for a certain range of \( u \), a consideration of sub-regions may be effective instead of use of a lot of Gauss points to evaluate Eqs. (45) and (46). For demonstration, let us consider the following knot vector:

\[
\Xi = [0, 0, 0, 0.5, 1, 1, 1]
\]

Then, the sub-regions are defined by considering the distinct knot values as follows:

\[
\begin{align*}
\xi_1^{(1)} &= 0.0 & u_1 &< \xi_1^{(1)} & \text{for Sub-region (1)} \\
\xi_2^{(2)} &= 0.5 & u_2 &< \xi_2^{(2)} & \text{for Sub-region (2)}
\end{align*}
\]

Corresponding to Eq. (48), we make use of the following equation to express \( u \)-coordinate in a sub-region (Fig. 6):

\[
\frac{u^{(i)}(\xi)}{\xi_1^{(i)} - \xi_2^{(i)}} = \left( \frac{1 - \xi}{2} \right) u_1^{(i)} + \left( \frac{1 + \xi}{2} \right) u_2^{(i)}
\]

Then, based on Eq. (41), we have:

\[
\frac{d}{d\xi} \frac{u^{(i)}(\xi)}{\xi_1^{(i)} - \xi_2^{(i)}} = L
\]

Accordingly:

\[
\frac{du^{(i)}}{d\xi} = \frac{1}{L} ; \quad \frac{d^2 u^{(i)}}{d\xi^2} = \frac{u_1^{(i)} - u_2^{(i)}}{2} L
\]

Finally, by considering the sub-regions, we can rewrite Eqs. (45) and (46) as follows:

\[
K_i = \frac{\sum_{\xi\in\Xi} \int_{\xi_1^{(i)}}^{\xi_2^{(i)}} \left( \frac{d^2 N_{i,p}^{(i)}(\xi)}{d\xi^2} \right) W_k \right) d\xi}{\sum_{\xi\in\Xi} q(\xi) N_{i,p}(\xi) \left( \frac{d^{(i)}}{d\xi} \right) W_k}
\]

\[
F_i = \frac{\sum_{\xi\in\Xi} q(\xi) N_{i,p}(\xi) \left( \frac{d^{(i)}}{d\xi} \right) W_k}{\sum_{\xi\in\Xi} q(\xi) N_{i,p}(\xi) \left( \frac{d^{(i)}}{d\xi} \right) W_k}
\]

where \( NS \) is the number of sub-regions.

3.2. Derivatives of B-spline Basis Functions

Let us denote the \( k \)th derivatives of B-spline basis functions, \( N_{i,p}^{(k)} \), as follows [5]:

\[
N_{i,p}^{(k)} = p \left( \frac{N_{i+1,p-1}^{(k-1)}(\xi)}{\xi_{i+p-1} - \xi_{i+p-1}} \right)
\]

where \( p \) represents the polynomial order. Accordingly, the 1\textsuperscript{st} derivatives (i.e., \( k = 1 \) in Eq. (54)) are given as:

\[
N_{i,p}^{(1)} = p \left( \frac{N_{i+1,p-1}^{(0)}(\xi)}{\xi_{i+p-1} - \xi_{i+p-1}} \right)
\]

Similarly:

\[
N_{i,p}^{(k-1)} = (p-1) \left( \frac{N_{i+1,p-1}^{(k-2)}(\xi)}{\xi_{i+p-1} - \xi_{i+p-1}} \right)
\]

Further, 2\textsuperscript{nd} derivatives (i.e., \( k = 2 \) in Eq. (54)) are given as:

\[
N_{i,p}^{(2)} = p \left( \frac{N_{i+1,p-1}^{(1)}(\xi)}{\xi_{i+p-1} - \xi_{i+p-1}} \right)
\]

By making use of Eqs. (56) and (57), Eq. (58) can be rewritten as:

\[
N_{i,p}^{(2)} = \left( p(p-1) \right) \left( \frac{N_{i+1,p-1}^{(0)}(\xi)}{\xi_{i+p-1} - \xi_{i+p-1}} \right)
\]

\[
\quad - (p-1) \left( \frac{N_{i+1,p-1}^{(1)}(\xi)}{\xi_{i+p-1} - \xi_{i+p-1}} \right)
\]

\[
\quad - c_1 N_{i+1,p-2} + c_2 N_{i+1,p-1} + c_3 N_{i+1,p-3}
\]

\[
\text{where:}
\]

\[
\begin{align*}
c_1 &= \frac{p(p-1)}{\xi_{i+p-1} - \xi_{i+p-1}} \\
c_2 &= \frac{p(p-1)}{\xi_{i+p-1} - \xi_{i+p-1}} + \frac{p(p-1)}{\xi_{i+p-1} - \xi_{i+p-1}} \\
c_3 &= \frac{p(p-1)}{\xi_{i+p-1} - \xi_{i+p-1}}
\end{align*}
\]

For instance:

\[
c_i = \begin{cases} 
0 & \text{if } \xi_{i+p-1} - \xi_{i+p-1} = 0 \\
p(p-1) & \text{otherwise}
\end{cases}
\]
The function "Bsp_deriv" is prepared to obtain the derivatives of B-spline basis functions.

```matlab
function[Nip, dNip1, dNip2] = Bsp_deriv(knot, p, u)
% Compute up to the 2nd-order derivatives of B-spline basis function

%% Input ==
% knot ... knot vector (open knot vector)
% p ... polynomial order
% u ... parameter value

%% Output ==
% Nip ... B-spline basis functions
% (column number corresponds to the Cox-de Boor recursion order)
% dNip1 ... 1st derivative of B-spline basis functions
% dNip2 ... 2nd derivative of B-spline basis functions

n = length(knot)-p-1;  % Number of basis functions
if u == 1
    u = 1-10^-10;
end;

%% Compute B-spline basis functions at a point, u ==
Nip = zeros(length(knot), 1, p+1);
for l = 1:length(knot)-1
    % 0th-order calculation
    if u >= knot(l) && u < knot(l+1)
        Nip(l,1) = 1;  % Column number represents 0th-order
    end;
end;
for K = 2:p+1
    % Higher-order calculation
    for l = 1:length(knot) - K
        pod = K - 1;  % Polynomial order
        c1 = knot(l+pod) - knot(l);  % Coefficient(1)
        if c1 ~= 0
            c1 = (u - knot(l))/c1;
        end;
        c2 = knot(l+pod+1) - knot(l+1);  % Coefficient(2)
        if c2 ~= 0
            c2 = (knot(l+pod+1) - u)/c2;
        end;
        Nip(l,K) = c1*Nip(l,pod) + c2*Nip(l+1,pod);
    end;
end;

%% Compute 1st derivatives ==
dNip1 = zeros(n,1);
for l = 1:n
    c1 = knot(l+p) - knot(l);
    if c1 ~= 0
        c1 = p/c1;
    end;
    c2 = knot(l+p+1) - knot(l+1);
    if c2 ~= 0
        c2 = p/c2;
    end;
    dNip1(l) = c1*Nip(l,p) - c2*Nip(l+1,p);
end;
```

Let us consider the following knot vector with polynomial order of \( p = 2 \):

\[
\Xi = [0, 0, 0, 0.5, 1, 1, 1]
\]  

(62)

By using the MATLAB code, the variations of B-spline basis functions are plotted in Fig. 7. In particular, Fig. 7(b) clearly shows that the basis functions are \( C^{p-1} = C^1 \) continuous (i.e., up to the 1st derivative is continuous).
3.3. MATLAB Code with B-spline Basis Functions

The following program is created to compute the deflection and the internal moment.

```matlab
% Simulate the beam problem with B-spline
% Gaussian point and weight =
Gpt = [0, 0, 0, 0, 0, 0, 0, 0; -0.577350, 0.577350, 0, 0, 0; -0.774597, 0.774597, 0, 0; -0.861136, -0.339981, 0.339981, 0.861136];
Gwt = [2.000000, 0.000000, 0.000000, 0.000000; 1.000000, 0.000000, 0.000000, 0.000000; 0.555556, 0.888889, 0.555556, 0.000000; 0.347855, 0.652145, 0.652145, 0.347855];

% Parameters =
E1 = 1;
L = 2; % Length of beam
ks = 3; % Spring constant

knot = [0, 0, 0, 0, 1, 6/2, 6/3, 6/4, 6/5, 6/6, 1, 1, 1, 1, 1];
p = 4; % Polynomial order
n = length(knot)/p + 1; % Number of basis functions/controls points

% Set up for sub-regions =
usub = [0, knot(p+2)]; % Initial sub-region
K = 1;
for i = (p+3):length(knot)
    if knot(i) > usub(K)
        usub = [usub, usub(K), knot(i)];
        K = K + 1;
    end;
end;

Fi = zeros(n, 1); % Force vector
Kj = zeros(n, n); % Stiffness matrix

Ng = 4; % Number of Gauss points
Nsub = length(usub(end, :)); % Number of sub-regions

for Ns = 1:Nsub
    us1 = usub(Ns, 1);
    us2 = usub(Ns, 2);

    for K = 1:Ng % Gauss integration
        xsi = Gpt(Ng, K); % Gaussian point
        u = (1-xsi)*us1 + (1+xsi)*us2;
        [Nip, dNip, dNip2] = Bsp_deriv(knot, p, u); % Basis function
        ddxsi = (us2-us1)/2; % Derivative
        % Calculate the force vector
        x = u^L; % Transformation from u to x;
        q = (x/L).*(1-x/L); % Distributed force
        df = xsip.*Nip.*ddxsi.*Gwt(Ng, K);
        Fi = Fi + df; % Gauss integration
    end;
end;
```

Fig. 7. B-spline basis functions.
4. Results and Discussion

Fig. 8 (cf. Fig. 4 and Eqs. (29) and (30)) shows the results with Hermite polynomials. As the number of degree of freedom, DOF, is increased, the convergence of solution is observed. This refinement is called $h$-refinement in which the same class of elements continues to be used but changed in size [6]. One of the most interesting aspects of B-splines is the myriad of ways in which the basis may be enriched while leaving the underlying geometry and its parametrization intact [7].

The first mechanism by which one can enrich the basis is knot insertion. Knots may be inserted without changing a curve geometrically or parametrically. In practice, when one adds one more knot, one add more basis function than in the unrefined case. This process may be repeated to enrich the solution space by adding more basis functions of the same order. Fig. 9(a) shows a process of $h$-refinement with polynomial order, $p = 2$.

The second mechanism by which one can enrich the basis is order elevation (sometimes also called degree elevation). As its name implies, the process involves raising the polynomial order of the basis functions. During order elevation, the multiplicity of each knot value is increased by one, but no new knot values are added. Fig. 9(b) shows a process of order elevation and the polynomial order is increased from $p = 2$ to $p = 4$.

Finally, $k$-refinement is unique to B-spline. This refinement involves the elevation of order before inserting the knots. Fig. 9(c) shows a process of $k$-refinement and the polynomial order is increased from $p = 2$ to $p = 4$. 

\begin{figure}
\centering
\includegraphics[width=\textwidth]{simulations_hermite}
\caption{Simulations with Hermite polynomials.}
\end{figure}
Fig. 9. Simulations with B-spline basis functions.
Fig. 10. Derivatives of B-spline basis functions.
Fig. 11. Three element, higher-order meshes for $p$- and $k$-refinement.
(Element is associated with the distinct knot values)
As mentioned, we can “insert” new knot values with multiplicity equal one to define the new elements across whose boundaries the basis functions will be $C^{p-1}$ (cf. Fig. 9(a)). Here, the elements are associated with the distinct values of knots. We can also “repeat” the existing knot values to lower the continuity of the basis functions across existing element boundaries. These insertion and repetition of knot values provide us with a more flexible process than simple $h$-refinement. In practice, we have a more flexible higher-order refinement as well [7], as demonstrated by Fig. 9.

For instance, if a unique knot value of $\xi^*$ is inserted between two distinct knot values in a curve of order $p$, the number of continuous derivatives of the basis functions at $u^*$ is $p-1$ (Fig. 10(a)). Subsequently, if we elevate the order from $p$ to $q$, the multiplicity of each knot value (including the newly inserted knot, $u^*$) is increased. Then, the basis still has $p-1$ continuous derivatives at $u^*$ in spite of the order elevation from $p$ to $q$ (Fig. 10(b)). If instead, we elevate the original coarse curve from $p$ to $q$ and then insert the unit knot value $u^*$, the basis would have $q-1$ continuous derivatives at $u^*$ (Fig. 10(c)). We refer to this procedure as $k$-refinement. We know no analogous practice in standard finite element analysis. The concept of $k$-refinement is potentially a superior approach for high-performance analysis in comparison with $p$-refinement.

In traditional $p$-refinement, there is a very inhomogeneous structure to arrays due to the different basis functions associated with surface, edge, and interior nodes. In addition, there is a proliferation in the number of control variables (i.e., basis functions and control points) because $C^0$-continuity is maintained in the refinement process, as shown in Fig. 11(a). On the other hand, in $k$-refinement, there is a homogeneous structure within patches and the growth in the number of control variables is limited, as shown by Fig. 11(b) [7].

5. Summary
In this study, an application of B-spline basis functions for structural mechanics is studied. By taking advantage of its simplicity, the deflection behavior of Euler-Bernoulli beam is simulated. By conduction a series of analysis with MATLAB programs, the unique characteristics rendered by B-spline is demonstrated.

References