Bézier Extraction

Bézier extraction is a process which allows us to use NURBS basis functions and geometries in conventional finite element codes. More precisely, Bézier extraction is a process by which we obtain a local element-wise representation of NURBS. Bézier extraction is executed by repeated knot insertion to yield a set of Bézier elements (i.e., a composite Bézier) from a given NURBS.

To fix our discussion, let us consider a B-spline curve:

$$\vec{C}(\xi) = \sum_{i=1}^{n} \vec{P}_i \, N_{i,p}(\xi)$$

for associated polynomial degree p and knot vector $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$. To obtain the composite Bézier form of the curve, we repeat all interior knots until they have multiplicity equal to p. This yields the refined curve:

$$\vec{C}(\xi) = \sum_{j=1}^{m} \vec{Q}_j \, \bar{N}_{j,p}(\xi)$$

where the basis $\{\bar{N}_{j,p}\}_{j=1}^{m}$ is associated with the extended knot vector:

$$\bar{\Xi} = \{\bar{\xi}_1, \ldots, \bar{\xi}_{m+p+1}\} \qquad m = n + \text{no. of added knots}$$

and new control points are computed using the dual relationships:

$$\underline{N} = \underline{T}^T \underline{\bar{N}} \quad \text{and} \quad \underline{Q}_A = \underline{T} \, \underline{P}_A \qquad A = 1, \ldots, d$$

$\underline{T}$: Refinement Matrix

Matrix $\underline{T}$ may be obtained using the Oslo algorithm or repeated applications of Boehm's algorithm.

A brief note on notation. Note that:

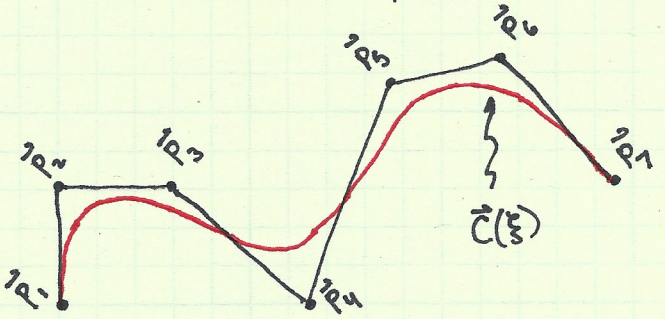
$$\{\underline{N}\}_i = N_{i,p}$$

However, we may also write $\underline{N}$ as:

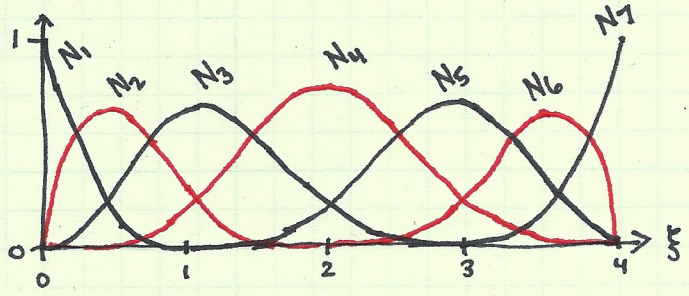
$$\underline{N} = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_n \end{bmatrix}$$

Thus, we will often omit p when referring to $N_i = N_{i,p}$.

To ground our discussion with a firm example, consider the following:



Control Polygon & B-spline Curve

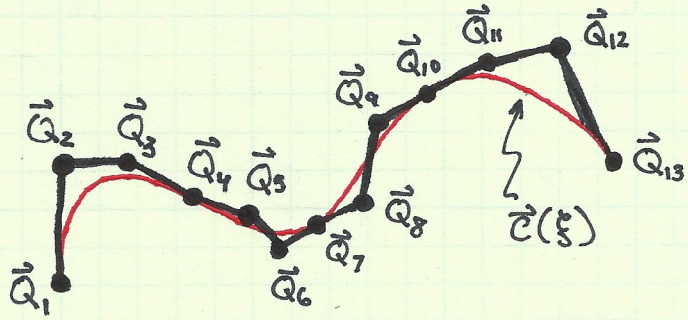


B-spline Basis

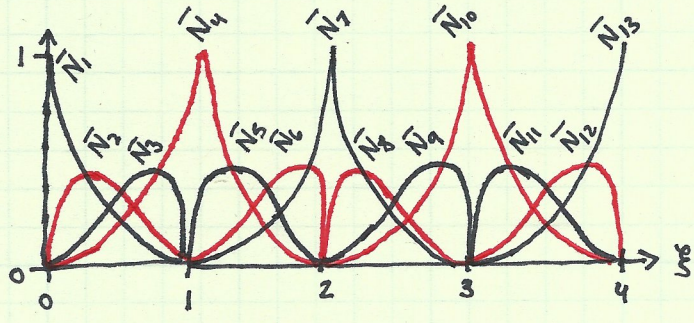
Parameters: $p = 3$

$\Xi = \{0,0,0,0,1,2,3,4,4,4,4\}$

After repeating all interior knots until they have multiplicity equal to p, we obtain the following:



Refined Control Polygon & B-spline Curve



Refined B-spline Basis

Parameters: $p = 3$

$\bar{\Xi} = \{0,0,0,0,1,1,1,2,2,2,3,3,3,4,4,4,4\}$

Note that each basis function in the set $\{\bar{N}_{j,p}\}_{j=1}^{m}$ is a scaled and translated Bernstein basis function of degree p. Thus, the control points $\{\vec{Q}_j\}_{j=1}^{m}$ correspond to a composite Bézier curve. For this reason, we will use the notation:

$$\left.\begin{aligned} B_j(\xi) &\equiv \overline{N}_{j,p}(\xi) \\ \vec{P}^b_j &\equiv \vec{Q}_j \end{aligned}\right\} \quad \text{for } j = 1, \ldots, m$$

We will also use the notation $\underline{\underline{C}} = \underline{\underline{T}}^T$, and we refer to $\underline{\underline{C}}$ as the Bézier extraction operator. We have:

$$\underline{N} = \underline{\underline{C}}\,\underline{B} \qquad \text{and} \qquad \underline{P}^b_A = \underline{\underline{C}}^T \underline{P}_A \qquad A = 1, \ldots, d$$

Consequently, the extraction operator transforms the refined basis into the original B-spline basis, and its transpose extracts the composite Bézier control polygon from the original control polygon.

For our earlier example, we have:

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{7}{12} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{7}{12} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \\ B_8 \\ B_9 \\ B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{bmatrix}$$

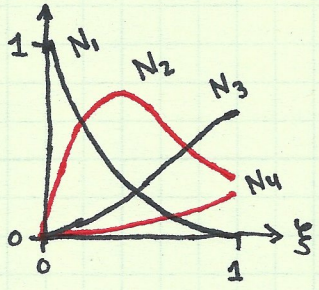
Basis function vector $\underline{N}$

Bézier extraction operator $\underline{\underline{T}}^T = \underline{\underline{C}}$

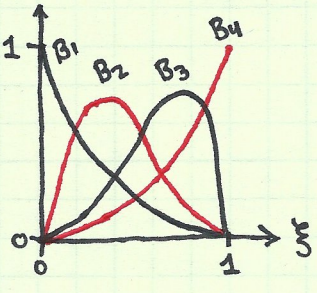
Refined basis function vector $\underline{B}$

Bézier decomposition results in one Bézier element for each non-empty knot interval in the original knot vector. As a consequence, over each non-empty knot interval, the original B-spline basis can be represented as a linear combination of the Bernstein basis functions corresponding to that knot interval.

For the first non-empty knot interval $[0, 1)$ in our earlier example, we have:



B-spline Basis



Bernstein Basis

Algebraically:

$$\begin{bmatrix} N_1(\xi) \\ N_2(\xi) \\ N_3(\xi) \\ N_4(\xi) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 7/12 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} B_1(\xi) \\ B_2(\xi) \\ B_3(\xi) \\ B_4(\xi) \end{bmatrix} \qquad \text{for } \xi \in [0,1)$$

As the knot interval $[0,1)$ is the first nonempty knot interval in Ξ, it is referred to as the first Bézier element, denoted as $\hat{\Omega}^1 = [0,1)$. We use the index e to refer to element e and quantities associated with element e.

For the other three Bézier elements, we have:

Element 2: $\hat{\Omega}^2 = [1,2)$

$$\begin{bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{bmatrix} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 7/12 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{bmatrix}$$

2/3

Element 3: $\hat{\Omega}^3 = [2,3)$

$$\begin{bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/4 \end{bmatrix} \begin{bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{bmatrix}$$

Element 4: $\hat{\Omega}^4 = [3,4)$

$$\begin{bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 7/12 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{bmatrix}$$

To localize the extraction operator, we first localize the basis functions to each element:

$$\begin{bmatrix} N_1^1 \\ N_2^1 \\ N_3^1 \\ N_4^1 \end{bmatrix} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix}, \quad \begin{bmatrix} N_1^2 \\ N_2^2 \\ N_3^2 \\ N_4^2 \end{bmatrix} = \begin{bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{bmatrix}, \quad \begin{bmatrix} N_1^3 \\ N_2^3 \\ N_3^3 \\ N_4^3 \end{bmatrix} = \begin{bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix}, \quad \begin{bmatrix} N_1^4 \\ N_2^4 \\ N_3^4 \\ N_4^4 \end{bmatrix} = \begin{bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{bmatrix}$$

$$\begin{bmatrix} B_1^1 \\ B_2^1 \\ B_3^1 \\ B_4^1 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix}, \quad \begin{bmatrix} B_1^2 \\ B_2^2 \\ B_3^2 \\ B_4^2 \end{bmatrix} = \begin{bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{bmatrix}, \quad \begin{bmatrix} B_1^3 \\ B_2^3 \\ B_3^3 \\ B_4^3 \end{bmatrix} = \begin{bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{bmatrix}, \quad \begin{bmatrix} B_1^4 \\ B_2^4 \\ B_3^4 \\ B_4^4 \end{bmatrix} = \begin{bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{bmatrix}$$

Then we have:

$$\underline{N}^e = \underline{\underline{C}}^e \underline{B}^e \qquad \text{for } e = 1,2,3,4$$

$\underline{\underline{C}}^e$: Localized Element Extraction Operator

where, for our example, the element extraction operators are:

$$\underline{\underline{C}}^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 7/12 \\ 0 & 0 & 0 & 1/6 \end{bmatrix}, \quad \underline{\underline{C}}^2 = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 7/12 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 2/3 \\ 0 & 0 & 0 & 1/6 \end{bmatrix}, \quad \underline{\underline{C}}^3 = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/4 \end{bmatrix}$$

$$\underline{\underline{C}}^4 = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 7/12 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that we have implicitly defined a mapping, referred to as the IEN array, from local basis function number a and element number e to corresponding global basis function i:

$$i = \text{IEN}(a,e)$$

such that:

$$N_a^e(\xi) \equiv N_{i,p}(\xi) \qquad \text{for } \xi \in \hat{\Omega}^e \text{ where } i = \text{IEN}(a,e)$$

As a convention, we will use i and j to denote global quantities and a and b to denote local quantities.

In the univariate case, we have a simple means of constructing the IEN array. Namely, for element $\hat{\Omega}^e = [\xi_l, \xi_{l+1})$:

$$\text{IEN}(a,e) = (l+a) - (p+1) \qquad a = 1,\dots,p+1$$

In addition to localizing basis functions, we may also localize control points. Namely, we define:

$$\vec{P}_a^e \equiv \vec{P}_i \qquad \text{where } i = \text{IEN}(a,e)$$

and element-wise Bézier control points analogously. Then we have:

$$\underline{P}_A^{b,e} = (\underline{\underline{C}}^e)^T \underline{P}_A^e \qquad A = 1,\dots,d$$

$\underline{P}_A^{b,e}$: Vector of Local Bézier Control Points

$\underline{P}_A^e$: Vector of Local B-Spline Control Points

Before proceeding, it is useful to discuss how to construct both the element extraction operators and the IEN array. In what follows, we present an algorithm for constructing the extraction operators based on repeated application of Boehm's algorithm. It was originally presented in:

M.J. Borden, M.A. Scott, J.A. Evans, and T.J.R. Hughes, "Isogeometric finite element data structures based on Bézier extraction of NURBS," IJNME, 87:15-47, 2011.

The algorithm works by inserting knots from left to right, and it is able to reduce the total computational cost by updating overlapping coefficients between neighboring elements. See Borden et al. for more details.

```
function Extraction1D (n, p, Ξ)
begin
    Set a = p+1                          } Set First Element to [ξ_a, ξ_b)
    Set b = a+1
    Set n_b = 1                          Initialize number of elements
    Set C^1 = I                          Initialize first extraction operator

    while b < n+p+1
        Set C^(nb+1) = I                 Initialize next operator
        Set i = b

        while b < n+p+1 and ξ_(b+1) = ξ_b
            update b += 1
        endloop

        set mult = b-i+1                 Compute multiplicity of ξ_b

        if mult < p
            set numer = ξ_b - ξ_a        Boehm's alphas

            for j = p, p-1, ..., mult+1
                set alphas(j-mult) = numer/(ξ_(a+j) - ξ_a)
            endloop

            set r = p-mult               Compute # of added knots

            for j = 1, 2, ..., r
                Set save = r-j+1         Region of overlap
                Set s = mult+j

                for k = p+1, p, ..., s+1
                    Set α = alphas(k-s)
                    Set C^nb(:,k) = α * C^nb(:,k)
                                  + (1-α) * C^nb(:,k-1)
                endfor

                if b < n+p+1
                    Set C^(nb+1)(save:j+save, save) =
                        C^nb(p-j+1:p+1, p+1)
                endif
            endfor
        endif
        if b < n+p+1
            Set a = b
            Set b = a+1                  } Set Next Element
```

Application of Boehm's Algorithm

Update overlapping coefficients

```
                    ← Set n_b = n_b + 1      Update number of elements
                  endif
            endif
      endwhile

      return n_b and { C^e }_{e=1}^{n_b}      Return # of elements and operators

end
```

The next algorithm computes the IEN array.

```
function  IEN 1D (n, p, Ξ)
begin
      set l = p+1
      set e = 1

      while  l < n+1
            for a = 1, 2, ..., p+1
                  IEN(a,e) = (l+a) - (p+1)
            endloop

            Set l = l+1

            while ξ_{l+1} = ξ_l  and  l < n+1
                  l = l+1
            end loop

            if l < n+1
                  Set e = e+1
            endif
      end loop

      return  IEN
end
```

Note that each "Bernstein" basis function is simply a scaled and translated version of a Bernstein basis function on the interval $[0,1]$. Notably, we have:

$$B_a^e(\xi^e(\hat{\xi})) = B_{a,p}(\hat{\xi}) \qquad \text{for } \hat{\xi} \in [0,1)$$

↑ Canonical Bernstein basis functions over $[0,1]$

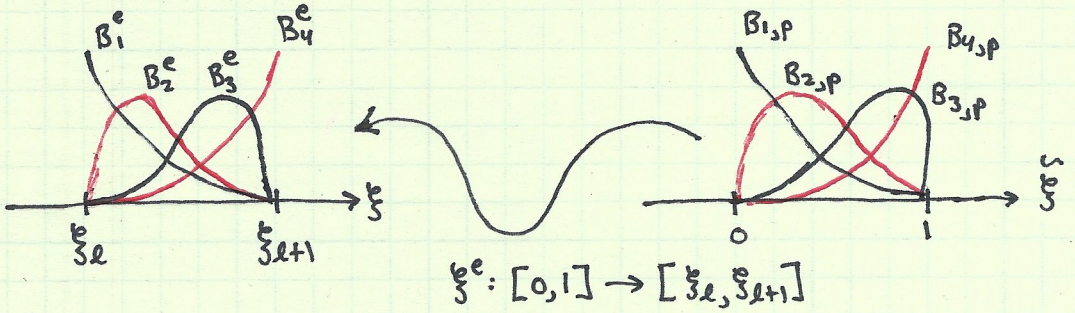
where:

$$B_{a,p}(\hat{\xi}) = \binom{p}{a-1} \hat{\xi}^{a-1} (1-\hat{\xi})^{p+1-a}$$

and $\xi^e : [0,1] \to \overline{\hat{\Omega}}^e$ is the affine map:

$$\xi^e(\tilde{\xi}) = (1-\tilde{\xi})\,\xi_\ell + \tilde{\xi}\,\xi_{\ell+1}$$

where $\hat{\Omega}^e = [\xi_\ell, \xi_{\ell+1})$. This allows us to pull all basis functions back to the canonical parent element $[0,1]$:



and hence we may reparametrize our curve as:

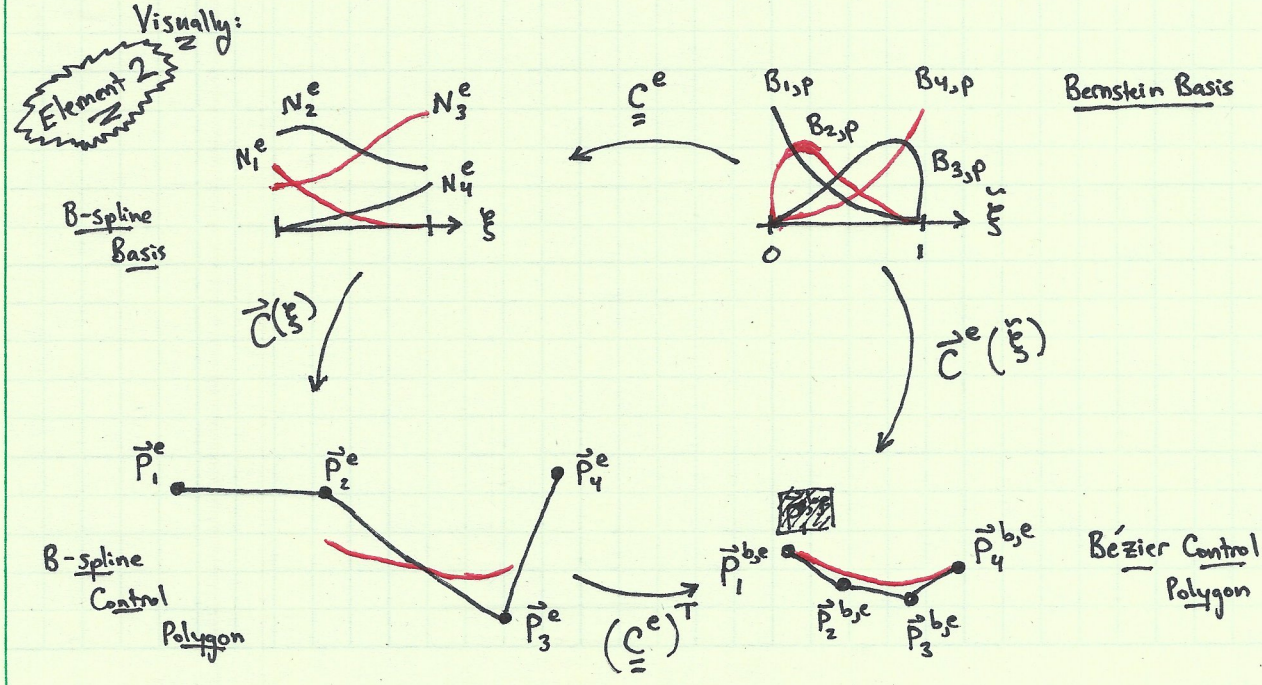
$$\vec{C}^e(\tilde{\xi}) = \sum_{a=1}^{p+1} \vec{P}_a^{b,e}\, B_{a,p}(\tilde{\xi}) \qquad \text{for } \tilde{\xi} \in [0,1)$$

This is the local, element-wise description of our B-spline curve. Moreover, we may represent our B-spline basis functions as:

$$N_a^e(\xi^e(\tilde{\xi})) = \sum_{b=1}^{p+1} C_{ab}^e\, B_{b,p}(\tilde{\xi}) \qquad \text{for } \tilde{\xi} \in [0,1)$$

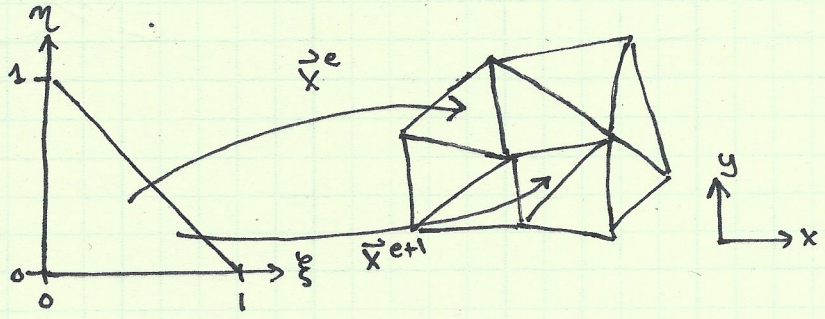
This is the local, element-wise description of our B-spline basis.

Visually:



Above, the top row represents the basis function point of view while the bottom represents the geometry point of view. The curves $\vec{C}(\xi)$ and $\vec{C}^e(\hat{\xi})$ link these points of view. Similarly, the left column represents the global description while the right represents the local description. These points of view are linked via the element extraction operator. Finally, we map local function numbers and element numbers to global function numbers using the IEN array.

In finite element analysis, we often consider building a basis on a canonical parent element and mapping it to the physical space using a specified set of nodal coordinates. We repeat this process for each element, as illustrated below:



This dramatically simplifies implementation. With Bézier extraction, we have obtained similar mappings from the canonical Bézier element to physical space. Moreover, we can reconstruct B-Spline basis functions simply using the extraction operator. As we shall later see, this will make possible the implementation of NURBS in a standard finite element code.

For NURBS curves, we perform Bézier extraction on the corresponding projective B-Spline curve. Recall that for a NURBS curve of the form:

$$\vec{C}(\xi) = \sum_{i=1}^{n} \vec{P}_i R_{i,p}(\xi)$$

the corresponding projective B-Spline curve is:

$$\vec{C}^w(\xi) = \sum_{i=1}^{n} \vec{P}_i^w N_{i,p}(\xi)$$

where:

$$\vec{P}_i^w = \begin{bmatrix} w_i \vec{P}_i \\ w_i \end{bmatrix}$$

Performing extraction, we find:

$$\vec{C}^{w,e}(\hat{\xi}) = \sum_{a=1}^{p+1} \vec{P}_a^{w,b,e} B_{a,p}(\hat{\xi}) \qquad \text{for } \hat{\xi} \in [0,1)$$

for every Bézier element. Then, the corresponding rational Bézier curve is:

$$\vec{C}^e(\tilde{\xi}) = \sum_{a=1}^{p+1} \frac{\vec{P}_a^{b,e} \; w_a^{b,e} \; B_{a,p}(\tilde{\xi})}{w^e(\tilde{\xi})} \qquad \text{for } \tilde{\xi} \in [0,1)$$

where:

$$\left(\vec{P}_a^{b,e}\right)_A = \left(\vec{P}_a^{w,b,e}\right)_A \Big/ w_a^{b,e} \qquad A = 1, \ldots, d$$

$$w_a^{b,e} = \left(\vec{P}_a^{w,b,e}\right)_{d+1}$$

$$w^e(\tilde{\xi}) = \sum_{a=1}^{p+1} w_a^{b,e} \, B_{a,p}(\tilde{\xi})$$

In analogous fashion, we can construct local representations of NURBS basis functions. Defining:

$$R_a^e(\xi) \equiv R_{i,p}(\xi) \qquad \text{for } \xi \in \hat{\Omega}^e$$

where $i = \text{IEN}(a,e)$, we have:

$$R_a^e\left(\xi^e(\tilde{\xi})\right) = \frac{w_a^e \, N_a^e\left(\xi^e(\tilde{\xi})\right)}{w^e(\tilde{\xi})} \qquad \text{for } \tilde{\xi} \in [0,1)$$

where $w_a^e \equiv w_i$ with $i = \text{IEN}(a,e)$. Thus, using the same extraction machinery as with B-splines, we obtain local, element-wise descriptions of our NURBS curve and local, element-wise descriptions of the NURBS basis.

In the multi-dimensional setting, we exploit the tensor product nature of higher dimension B-splines and NURBS. In the following, we restrict our discussion to the bivariate setting. Here, we have:

$$N_{\underline{i},\underline{p}}(\xi_1, \xi_2) = N^{(1)}_{i_1,p_1}(\xi_1) \, N^{(2)}_{i_2,p_2}(\xi_2)$$

($\underline{i}$, $\underline{p}$: multi-indices)

We may perform Bézier extraction in each direction, yielding:

$$N^{(1)}_{i_1,p_1}(\xi_1) = \sum_{j_1=1}^{m_1} C^{(1)}_{i_1 j_1} \, B^{(1)}_{j_1,p_1}(\xi_1)$$

$$N^{(2)}_{i_2,p_2}(\xi_2) = \sum_{j_2=1}^{m_2} C^{(2)}_{i_2 j_2} \, B^{(2)}_{j_2,p_2}(\xi_2)$$

(Refined Basis in directions ξ_1 and ξ_2)

where $\underline{\underline{C}}^{(1)}$ and $\underline{\underline{C}}^{(2)}$ are the univariate Bézier extraction operators in directions ξ_1 and ξ_2. Defining:

$$B_{j,p}(\xi_1,\xi_2) = B^{(1)}_{j_1,p_1}(\xi_1)\, B^{(2)}_{j_2,p_2}(\xi_2) \qquad j=(j_1,j_2)$$

we have:

$$N_{1,p}(\xi_1,\xi_2) = \sum_{j\in J} C^{(1)}_{i_1,j_1}\, C^{(2)}_{i_2,j_2}\, B_{j,p}$$

where:

$$J \equiv \{(j_1,j_2) : 1 \le j_1 \le m_1,\ 1 \le j_2 \le m_2\}$$

In matrix form:

$$\underline{N}(\xi_1,\xi_2) = \underline{\underline{C}}\ \underline{B}(\xi_1,\xi_2)$$

↑ Multi-variate Extraction Operator

where:

$$\underline{N}(\xi_1,\xi_2) = \underline{N}^{(1)}(\xi_1) \otimes \underline{N}^{(2)}(\xi_2)$$

$$\underline{B}(\xi_1,\xi_2) = \underline{B}^{(1)}(\xi_1) \otimes \underline{B}^{(2)}(\xi_2)$$

$$\underline{\underline{C}} = \underline{\underline{C}}^{(1)} \otimes \underline{\underline{C}}^{(2)}$$

Moreover, we find the (global) Bézier coefficients are defined by:

$$\underline{P}^b_A = \underline{\underline{C}}^T \underline{P}_A \qquad A = 1,\dots,d$$

For the purposes of computation, it is useful to associate a single index with each basis function and associated control point. To determine an indexing scheme, note that:

$$\underline{N}(\xi_1,\xi_2) = \begin{bmatrix} N_1(\xi_1,\xi_2) \\ N_2(\xi_1,\xi_2) \\ N_3(\xi_1,\xi_2) \\ \vdots \\ N_n(\xi_1,\xi_2) \end{bmatrix} = \begin{bmatrix} N^{(1)}_{1,p_1}(\xi_1)\, N^{(2)}_{1,p_2}(\xi_2) \\ N^{(1)}_{2,p_1}(\xi_1)\, N^{(2)}_{2,p_2}(\xi_2) \\ N^{(1)}_{1,p_1}(\xi_1)\, N^{(2)}_{3,p_2}(\xi_2) \\ \vdots \\ N^{(1)}_{n_1,p_1}(\xi_1)\, N^{(2)}_{n_2,p_2}(\xi_2) \end{bmatrix}$$

where $n = n_1 * n_2$. Then, it makes sense to define:

$$N_i(\underline{\xi}_1, \underline{\xi}_2) = N^{(1)}_{i_1,p_1}(\xi_1)\, N^{(2)}_{i_2,p_2}(\xi_2)$$

where:

$$i = (i_1 - 1)\, n_2 + i_2$$

This indexing scheme dramatically simplifies computer implementation. As in the univariate case, we may also localize extraction. To do so, we first localize extraction in each parametric direction.

In direction $\underline{\xi}_1$, extraction yields a set of one-dimensional elements $\hat{\Omega}^{e_{1,1}}$ and extraction operators $\{\underline{\underline{C}}^{e_{1,1}}\}_{e_{1,1}=1}^{n_{e_{1,1}}}$. We can then localize our one-dimensional B-spline basis functions, obtaining the relationship:

$$\underline{N}^{e_{1,1}} = \underline{\underline{C}}^{e_{1,1}}\, \underline{B}^{e_{1,1}} \qquad \text{(Extraction in Direction 1)}$$

and if we pull back the Bernstein basis functions to the parent element $[0,1]$, we have:

$$\underline{N}^{e_{1,1}}(\underline{\xi}_1^{e_1}(\tilde{\xi}_1)) = \underline{\underline{C}}^{e_{1,1}}\, \underline{B}^{1}(\tilde{\xi}_1) \qquad \text{where } \tilde{\xi}_1 \in [0,1)$$

with $\underline{\xi}_1^{e_1} : [0,1] \to \overline{\hat{\Omega}^{e_{1,1}}}$ defined by:

$$\underline{\xi}_1^{e_1}(\tilde{\xi}_1) = (1 - \tilde{\xi}_1)\, \underline{\xi}^{(1)}_{\ell_1} + \tilde{\xi}_1\, \underline{\xi}^{(2)}_{\ell_1+1}$$

where $\hat{\Omega}^{e_{1,1}} = [\underline{\xi}^{(1)}_{\ell_1}, \underline{\xi}^{(1)}_{\ell_1+1})$.

In direction $\underline{\xi}_2$, we repeat the above process. Extraction yields a set of one-dimensional elements $\hat{\Omega}^{e_{2,2}}$ and extraction operators $\{\underline{\underline{C}}^{e_{2,2}}\}_{e_2=1}^{n_{e_{1,2}}}$. Localizing yields:

$$\underline{N}^{e_{2,2}} = \underline{\underline{C}}^{e_{2,2}}\, \underline{B}^{e_{2,2}} \qquad \text{(Extraction in Direction 2)}$$

and if we pull back the Bernstein basis functions to the parent element $[0,1]$, we have:

$$\underline{N}^{e_{2,2}}(\underline{\xi}_2^{e_2}(\tilde{\xi}_2)) = \underline{\underline{C}}^{e_{2,2}}\, \underline{B}^{2}(\tilde{\xi}_2) \qquad \text{where } \tilde{\xi}_2 \in [0,1)$$

with $\underline{\xi}_2^{e_2} : [0,1] \to \overline{\hat{\Omega}^{e_{2,2}}}$ defined by:

$$\underline{\xi}_2^{e_2}(\tilde{\xi}_2) = (1 - \tilde{\xi}_2)\, \underline{\xi}^{(2)}_{\ell_2} + \tilde{\xi}_2\, \underline{\xi}^{(2)}_{\ell_2+1}$$

where $\hat{\Omega}^{e_{2,2}} = [\underline{\xi}^{(2)}_{\ell_2}, \underline{\xi}^{(2)}_{\ell_2+1})$.

We define corresponding multi-dimensional B-spline basis functions through tensor products:

$$\underline{N}^{e}(\underline{\xi}_1, \underline{\xi}_2) = \underline{N}^{e_{1,1}}(\xi_1) \otimes \underline{N}^{e_{2,2}}(\xi_2)$$

Thus, we have:

Remark: Here we have implicitly defined element e:

$$\begin{cases} \hat{\Omega}^{e} := \hat{\Omega}^{e_{1,1}} \overset{\times}{\otimes} \hat{\Omega}^{e_{2,2}} \\ e := (e_{1,1} - 1)\, n_{e_{1,2}} + e_{2,2} \end{cases}$$

$$\begin{bmatrix} N_1^e(\xi_1,\xi_2) \\ N_2^e(\xi_1,\xi_2) \\ \vdots \\ N_{n_{loc}}^e(\xi_1,\xi_2) \end{bmatrix} = \begin{bmatrix} N_1^{e_1,1}(\xi_1) N_1^{e_2,2}(\xi_2) \\ N_1^{e_1,1}(\xi_1) N_2^{e_2,2}(\xi_2) \\ \vdots \\ N_{p_1+1}^{e_1,1}(\xi_1) N_{p_2+1}^{e_2,2}(\xi_2) \end{bmatrix}$$

where $n_{loc} = (p_1+1)(p_2+1)$, and hence:

$$N_a^e(\xi_1,\xi_2) = N_{a_1}^{e_1,1}(\xi_1) N_{a_2}^{e_2,2}(\xi_2)$$

where:

$$a = (a_1-1)(p_2+1) + a_2$$

Note that if we define:

$$\underline{B}^e(\xi_1,\xi_2) = \underline{B}^{e_1,1}(\xi_1) \otimes \underline{B}^{e_2,2}(\xi_2)$$

$$\underline{\underline{C}}^e = \underline{\underline{C}}^{e_1,1} \otimes \underline{\underline{C}}^{e_2,2}$$

then:

$$\underline{N}^e(\xi_1,\xi_2) = \underline{\underline{C}}^e \, \underline{B}^e(\xi_1,\xi_2)$$

Two-Dimensional Extraction

By pulling back to the parent domain $[0,1]\times[0,1]$, we obtain a fully local, element-wise description of the basis:

$$\underline{N}^e\left(\xi_1^{e_1}(\tilde{\xi}_1), \xi_2^{e_2}(\tilde{\xi}_2)\right) = \underline{\underline{C}}^e \, \underline{B}(\tilde{\xi}_1,\tilde{\xi}_2) \qquad \text{where } \tilde{\xi}_1,\tilde{\xi}_2 \in [0,1)$$

Above, $\underline{B}(\tilde{\xi}_1,\tilde{\xi}_2) = \underline{B}^1(\tilde{\xi}_1) \otimes \underline{B}^2(\tilde{\xi}_2)$ is the tensor-product Bernstein basis over the parent element $[0,1]\times[0,1]$:

$$B_a(\tilde{\xi}_1,\tilde{\xi}_2) = B_{a_1,p_1}(\tilde{\xi}_1) B_{a_2,p_2}(\tilde{\xi}_2)$$

where $a = (a_1-1)(p_2+1)+a_2$. Throughout, we will use the abuse of notation:

$$\underline{N}^e(\xi_1,\xi_2) = \underline{\underline{C}}^e \, \underline{B}(\tilde{\xi}_1,\tilde{\xi}_2)$$

where the pullback to the parent element is implied.

Note that:

$$N_{a_1}^{e_1,1}(\xi_1) \equiv N_{i_1,p_1}^{(1)}(\xi_1) \qquad \text{for } i_1 = (l_1+a_1)-(p_1+1), \quad a_1 = 1,\dots,p_1+1$$

$$N_{a_2}^{e_2,2}(\xi_2) \equiv N_{i_2,p_2}^{(2)}(\xi_2) \qquad \text{for } i_2 = (l_2+a_2)-(p_2+1), \quad a_2 = 1,\dots,p_2+1$$

for one-dimensional Bézier elements $\hat{\Omega}_1^{e_1,1} = [\xi_{l_1}^{(1)}, \xi_{l_1+1}^{(1)})$ and $\hat{\Omega}_1^{e_2,1} = [\xi_{l_2}^{(2)}, \xi_{l_2+1}^{(2)})$.
Thus, for element $\hat{\Omega}_1^e = \hat{\Omega}_1^{e_1,1} \times \hat{\Omega}_1^{e_2,2}$:

$$N_a^e(\xi_1,\xi_2) \equiv N_i(\xi_1,\xi_2)$$

$$N_{a_1}^{e_1,1}(\xi_1) N_{a_2}^{e_2,2}(\xi_2) \qquad = \qquad N_{i_1,p_1}^{(1)}(\xi_1) N_{i_2,p_2}^{(2)}(\xi_2)$$

where:

$$i = (i_1 - 1) n_2 + i_2$$
$$i_1 = (l_1 + a_1) - (p_1 + 1)$$
$$i_2 = (l_2 + a_2) - (p_2 + 1)$$

Thus, we have defined a multi-dimensional IEN array such that:

$$N_a^e(\xi_1,\xi_2) \equiv N_i(\xi_1,\xi_2) \qquad \text{for } (\xi_1,\xi_2) \in \hat{\Omega}_1^e \text{ where } i = \text{IEN}(a,e)$$

As in the univariate case, we may also localize control points. We define:

$$\vec{P}_a^e \equiv \vec{P}_i \qquad \text{where } i = \text{IEN}(a,e)$$

and element-wise Bézier control points analogously. Then:

$$\underline{P}_A^{b,e} = (\underline{\underline{C}}^e)^T \underline{P}_A^e \qquad A = 1,\dots,d$$

$\underline{P}_A^{b,e}$: Vector of Local Bézier Control Points

$\underline{P}_A^e$: Vector of Local B-spline Control Points

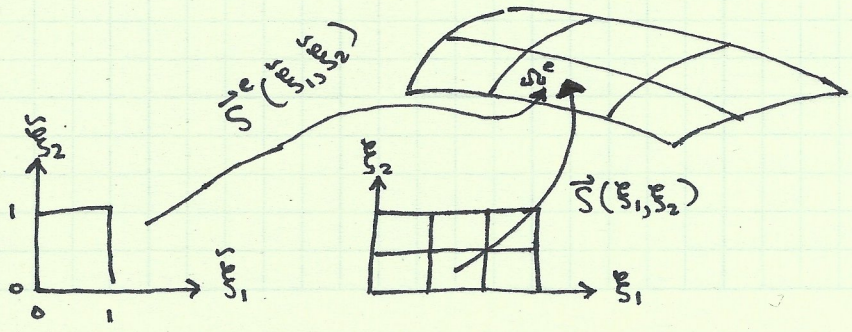
and thus we see a B-spline surface may be defined globally as:

$$\vec{S}(\xi_1,\xi_2) = \sum_{i \in I} \vec{P}_i N_{i,p}(\xi_1,\xi_2) \qquad \text{(Global View)}$$

or element-wise as:

$$\vec{S}^e(\tilde{\xi}_1,\tilde{\xi}_2) = \sum_{a=1}^{n_{loc}} \vec{P}_a^{b,e} B_a(\tilde{\xi}_1,\tilde{\xi}_2) \qquad \text{(Local View)}$$

Visually:



Both the global and local views of a B-spline surface are valid, but the local view is more amenable to finite element analysis.

We now present two algorithms for constructing the element extraction operators and the IEN array, respectively, for two-dimensional B-spline basis functions.

```
function  Extraction 2D (n1, n2, p1, p2, Ξ1, Ξ2)
begin

    call Extraction 1D (n1, p1, Ξ1) to obtain nel,1 and {C^(e1,1)}_(e1=1)^(nel,1)

    call Extraction 1D (n2, p2, Ξ2) to obtain nel,2 and {C^(e2,2)}_(e2=1)^(nel,2)

    set nel = nel,1 * nel,2

    for e1 = 1, ..., nel,1

        for e2 = 1, ..., nel,2

            Set e = (e1 - 1) nel,2 + e2          Element #
            set C^e = C^(e1,1) ⊗ C^(e2,2)        Operator

        endloop

    endloop

    return nel and {C^e}_(e=1)^(nel)             Return # of elements and
                                                 operators
end
```

```
function  IEN 2D (n1, n2, p1, p2, Ξ1, Ξ2)
begin

    call IEN 1D (n1, p1, Ξ1) to generate IEN^(1), the connectivity array for
                                                   direction ξ1
    call IEN 1D (n2, p2, Ξ2) to generate IEN^(2), the connectivity array for
                                                   direction ξ2

    set nel,1 = # of columns of IEN^(1)
    set nel,2 = # of columns of IEN^(2)

    for e1 = 1, ..., nel,1

        for a1 = 1, ..., p1 + 1

            Set i1 = IEN^(1)(a1, e1)
```

```
                    for e_2 = 1, ..., n_el,2

                        for a_2 = 1, ..., p_2+1

                            Set i_2 = IEN^(2)(a_2, e_2)

                            Set e = (e_1 - 1) n_el,2 + e_2
                            Set a = (a_1 - 1)(p_2 + 1) + a_2
                            Set i = (i_1 - 1) n_2 + i_2

                            Set IEN(a, e) = i

                        endloop

                    endloop

                endloop

            endloop

            return IEN

        end
```

Extraction of NURBS surfaces works in the same manner as NURBS curves. Namely, we perform extraction on the projective B-spline surface. We omit details here as they are largely the same as the NURBS curve setting.