Computation of B-spline Basis Functions:

The local support of B-spline basis functions suggests there must be an efficient means to compute them. Indeed, supposing that $\xi \in [\xi_l, \xi_{l+1})$, only basis functions:

$$N_{i,j} \qquad \begin{array}{l} i = l-j, \dots, l \\ j = 0, \dots, p \end{array}$$

are nonzero. Moreover, we have the relationships:

$$\boxed{\underline{p=0}: \qquad N_{l,0} = 1}$$

$$\underline{p=1}: \qquad N_{l,1} = \left(\frac{\xi - \xi_l}{\xi_{l+1} - \xi_l}\right) N_{l,0} = \underline{\mathcal{B}_{1,l}}\, N_{l,0}$$

$$N_{l-1,1} = \left(\frac{\xi_{l+1} - \xi}{\xi_{l+1} - \xi_l}\right) N_{l,0} = (1 - \underline{\mathcal{B}_{1,l}}) N_{l,0}$$

$$\underline{p=2}: \qquad N_{l,2} = \left(\frac{\xi - \xi_l}{\xi_{l+2} - \xi_l}\right) N_{l,1} = \mathcal{B}_{2,l}\, N_{l,0}$$

$$N_{l-1,2} = \left(\frac{\xi_{l+2} - \xi}{\xi_{l+2} - \xi_l}\right) N_{l,1} + \left(\frac{\xi - \xi_{l-1}}{\xi_{l+1} - \xi_{l-1}}\right) N_{l-1,1}$$

$$= (1 - \mathcal{B}_{2,l}) N_{l,1} + \mathcal{B}_{2,l-1} N_{l-1,1}$$

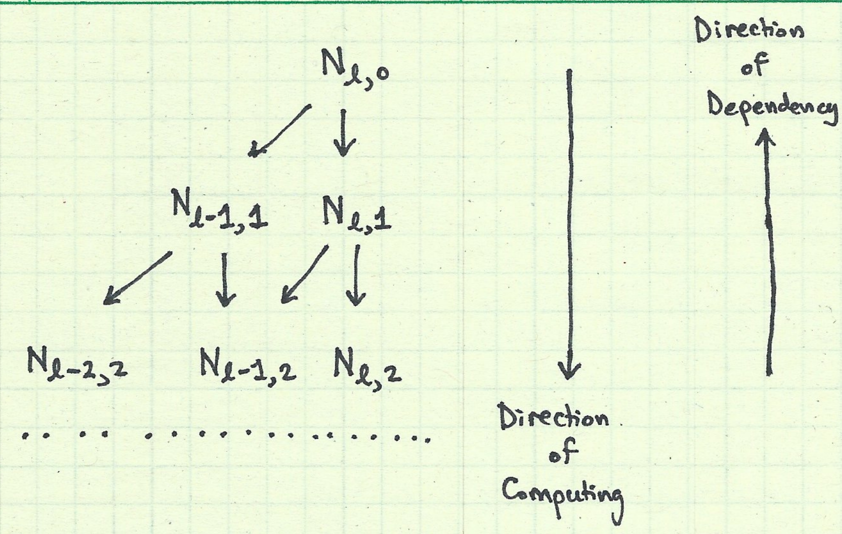
$$N_{l-2,2} = \left(\frac{\xi_{l+1} - \xi}{\xi_{l+1} - \xi_{l-1}}\right) N_{l-1,1} = (1 - \mathcal{B}_{2,l-1}) N_{l-1,1}$$

$$\boxed{\begin{aligned} \underline{p \geq 0}: \quad N_{l,p} &= \mathcal{B}_{p,l}\, N_{l,p-1} \\ N_{l-k,p} &= (1 - \mathcal{B}_{p,l-k+1}) N_{l-k+1,p-1} \\ &\quad + \mathcal{B}_{p,l-k} N_{l-k,p-1} \\ &\quad \underline{\text{for}}\ k = 1, \dots, p-1 \\ N_{l-p,p} &= (1 - \mathcal{B}_{p,l-p+1}) N_{l-p+1,p-1} \end{aligned}}$$

where:

$$\boxed{\mathcal{B}_{k,i} = \frac{\xi - \xi_i}{\xi_{i+k} - \xi_i}}$$

This suggests we can compute $\underline{\text{all}}$ $\frac{1}{2} p(p+1)$ nonzero basis functions in $O(p^2)$ time by exploiting the triangular scheme:



As opposed to our earlier algorithms, our new algorithm is not recursive but rather constructive. Pseudocode is listed below.

function Compute Spline Basis (ξ)
begin
  find ℓ s.t. $\xi \in [\xi_\ell, \xi_{\ell+1})$ via binary search
  initialize $N_{i,k} = 0$ for $k = 0, \dots, p$ and $i = \ell-k, \dots, \ell$
  set $N_{\ell,0} = 1$
  for $k = 1, \dots, p$
    for $i = \ell-k+1, \dots, \ell$
      set $\beta_{k,i} = (\xi - \xi_i)/(\xi_{i+k} - \xi_i)$
      update $N_{i,k}$ += $\cancel{\alpha}\,\beta_{k,i} N_{i,k-1}$
      update $N_{i-1,k}$ += $(1 - \cancel{\alpha}\,\beta_{k,i}) N_{i,k-1}$
    endloop
  endloop
  return basis functions and ℓ
end

Above, the notation += indicates:

$$x \mathrel{+}= y \quad \Rightarrow \quad x = x + y$$

The function Compute Spline Basis has a few redundant computations and memory accesses/updates, but it is still quite efficient. The NURBS Book by Piegl and Tiller has a more efficient algorithm but it is more difficult to digest.

The function Compute Spline Basis also suggests another means of computing the value of a B-spline curve.

```
function  Compute Spline Curve (ξ)
begin
          call  Compute Spline Basis (ξ)
          return   P_{l-p} * N_{l-p,p} + ... + P_l * N_{l,p}
end
```

$$\vec{P}_{\ell-p} * N_{\ell-p,p} + \dots + \vec{P}_{\ell} * N_{\ell,p}$$

We will also use Compute Spline Basis later to compute NURBS basis functions and NURBS curves.