# Pre-evaluation and Interactive Editing of B-spline and GERBS Curves and Surfaces.

Arne Lakså

*UiT - The Arctic University of Norway*
*Po.box. 385, N-8505 Narvik, Norway.*

**Abstract.** Interactive computer based geometry editing is very useful for designers and artists. Our goal has been to develop useful tools for geometry editing in a way that increases the ability for creative design.

When we interactively editing geometry, we want to see the change happening gradually and smoothly on the screen. Pre-evaluation is a tool for increasing the speed of the graphics when doing interactive affine operation on control points and control surfaces. It is then possible to add details on surfaces, and change shape in a smooth and continuous way.

We use pre-evaluation on basis functions, on blending functions and on local surfaces. Pre-evaluation can be made hierarchically and is thus useful for local refinements. Sampling and plotting of curves, surfaces and volumes can today be handled by the GPU and it is therefore important to have a structured organization and updating system to be able to make interactive editing as smooth and user friendly as possible. In the following, we will show a structure for pre-evaluation and an optimal organisation of the computation and we will show the effect of implementing both of these techniques.

## Introduction

In interactive editing of curves and surfaces, especially of type B-splines, Bézier or Expo-Rational B-splines, we select and then move control points or we move, rotate, scale local curves/surfaces directly on the computer screen. We want the graphics to be updated continuously in a smooth way, for example 30 times a second. Therefore, we try to significantly reduce the number of operations when updating the graphics. There are two ways we can achieve this, using pre-evaluation, and using optimal organization in the recalculations.

## B-splines

B-splines on arbitrary spaced knots was introduced by Curry and Schoenberg in an abstract in 1947, see [1]. The article was actually published nearly 20 years later, in 1966, see [2]. B-splines became more accessible to practical use when Cox-de'Boor recursion formula was introduced in 1972, see [3]. B-splines (the basis functions) is notated in two ways: $b_{d,i}(t) = b(t; t_i, ..., t_{i+d+1})$, where $d$ is the polynomial degree. The recursive formula is:

$$b_{d,i}(t) = b(t; t_i, ..., t_{i+d+1}) = w_{d,i}(t)\, b_{d-1,i}(t) + (1 - w_{d,i+1}(t))\; b_{d-1,i+1}(t) \tag{1}$$

where the termination of the recursion is

$$b_{0,i}(t) = b(t; t_i, t_{i+1}) = \begin{cases} 1, & t_i \le t < t_{i+1}, \\ 0, & \text{otherwise}, \end{cases}$$

and the mapping of the domain of each basis function onto $[0, 1]$,

$$w_{d,i}(t) = \frac{t - t_i}{t_{i+d} - t_i}. \tag{2}$$

As we can see in (1) is it only one basis function starting at a knot, and the domain of a basis function spans $d + 1$ knot intervals.

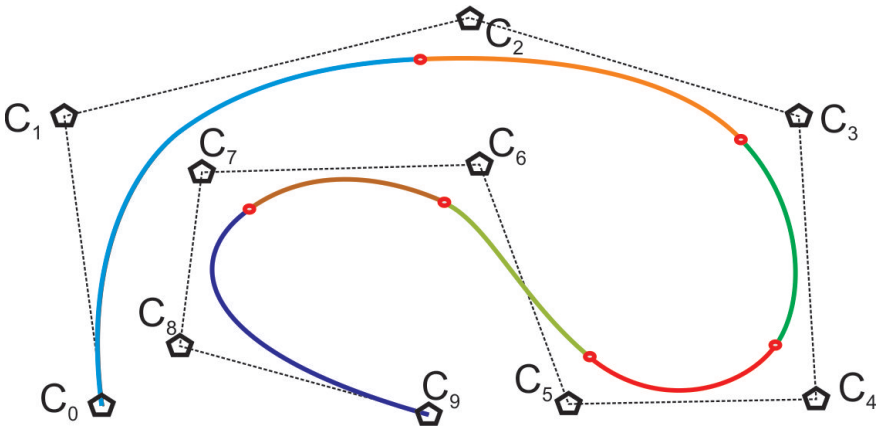


**FIGURE 1.** A B-spline curve and the control polygon, where each piece of the curve is marked. The pentagons are the control points, the red circles marks the knot values. The degree is 3, and together with the 10 control points it requires that the knot vector has 14 elements. The curve is $C^2$-smooth.

A B-spline curve is determined by a degree $d$, a knot vector and control points, where the interval between two consecutive knots defines the domain of a single polynomial curve of degree $d$. Over a single knot value is the function $C^{d-1}$-continuous (smooth). The B-spline spline curve expression is,

$$c(t) = \sum_{i=0}^{n-1} c_i \, b_{d,i}(t). \tag{3}$$

In Figure 1 is a spline curve illustrated. The control polygon and each piece of the spline curve are marked. In the example is the knot vector $\mathbf{t} = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 7, 7, 7\}$. As we can see, the curve consist of 7 parts. Each part is a $3^{\text{th}}$ degree polynomial curve.

Since the domain of a basis function is restricted to maximum $d + 1$ knot intervals will the consequences of moving one control point only be a local change of the total curve. For example will only the first piece be affected if we move $C_0$. If we move $C_4$ then piece number 2,3,4 and 5 will change. The general rule is that moving point $C_i$ affect the curve $c(t),\ t \in [t_i, t_{i+d+1}]$.

## Factorization of B-splines

To study B-splines, it is convenient to factorize the basis functions that are not zero over one knot interval. We know that the number of basis functions that is not zero on one knot interval is $d + 1$. Because of this we can simplify the expression (3) to the following,

$$c(t) = \sum_{j=i-d}^{i} c_j b_{d,j}(t) = \left[b_{d,i-d}(t),\ \cdots,\ b_{d,i-1}(t),\ b_{d,i}(t)\right] \begin{pmatrix} c_{i-d} \\ \cdots \\ c_{i-1} \\ c_i \end{pmatrix}, \qquad \textit{when} \quad t_i \le t < t_{i+1}. \tag{4}$$

Further, it follows that between the knots $t_i$ and $t_{i+1}$ can a polynomial B-spline curve be factorized as described in article [4]. In the following example, the factorizing of (4) is implemented on a $3^{\text{th}}$ degree B-spline curve,

$$\begin{aligned} c(t) = \begin{pmatrix} 1 - w_{1,i}(t) & w_{1,i}(t) \end{pmatrix} \begin{pmatrix} 1 - w_{2,i-1}(t) & w_{2,i-1}(t) & 0 \\ 0 & 1 - w_{2,i}(t) & w_{2,i}(t) \end{pmatrix} \\ \begin{pmatrix} 1 - w_{3,i-2}(t) & w_{3,i-2}(t) & 0 & 0 \\ 0 & 1 - w_{3,i-1}(t) & w_{3,i-1}(t) & 0 \\ 0 & 0 & 1 - w_{3,i}(t) & w_{3,i}(t) \end{pmatrix} \begin{pmatrix} c_{i-3} \\ c_{i-2} \\ c_{i-1} \\ c_i \end{pmatrix}, \end{aligned} \tag{5}$$

where $i$ is determined by $t_i \le t < t_{i+1}$, and where $w_{d,i}(t)$ is defined in expression (2). A more complete matrix notation is described further in [4].

## The B-spline-Hermite matrix

For a 3th degree B-spline curve, described in (5), vi get the following expression for the curve and for the 1st, the 2nd and the 3th order derivatives,

$$\begin{aligned} c(t) &= T_1(t)T_2(t)T_3(t)\,\mathbf{c} = T^3(t)\,\mathbf{c} \\ c'(t) &= 3\,T_1(t)T_2(t)T_3'\,\mathbf{c} = 3\,T^2(t)T'\,\mathbf{c} \\ c''(t) &= 6\,T_1(t)T_2'T_3'\,\mathbf{c} = 6\,T(t)T'^2\,\mathbf{c} \\ c'''(t) &= 6\,T_1'T_2'T_3'\,\mathbf{c} = 6\,T'^3\,\mathbf{c} \end{aligned} \tag{6}$$

where $T_1(t)$, $T_2(t)$ and $T_3(t)$ are the factor matrices described from left hand side in (5), $T'$ is the derivative of a factor matrix (that is a matrix of constants in the polynomial case) and $\mathbf{c}$ is the vector of control points described on right hand side in (5). On matrix form (6) gives,

$$\begin{pmatrix} c(t) \\ c'(t) \\ c''(t) \\ c'''(t) \end{pmatrix} = \begin{pmatrix} T^3(t) \\ 3\,T^2(t)T' \\ 6\,T(t)T'^2 \\ 6\,T'^3 \end{pmatrix} \begin{pmatrix} c_{i-3} \\ c_{i-2} \\ c_{i-1} \\ c_i \end{pmatrix} = M_{3,i}(t)\,\mathbf{c}$$

where the B-spline-Hermite matrix is

$$M_{3,i}(t) = \begin{pmatrix} T^3(t) \\ 3\,T^2(t)T' \\ 6\,T(t)T'^2 \\ 6\,T'^3 \end{pmatrix} = \begin{pmatrix} b_{3,i-3}(t) & b_{3,i-2}(t) & b_{3,i-1}(t) & b_{3,i}(t) \\ Db_{3,i-3}(t) & Db_{3,i-2}(t) & Db_{3,i-1}(t) & Db_{3,i}(t) \\ D^2b_{3,i-3}(t) & D^2b_{3,i-2}(t) & D^2b_{3,i-1}(t) & D^2b_{3,i}(t) \\ D^3b_{3,i-3}(t) & D^3b_{3,i-2}(t) & D^3b_{3,i-1}(t) & D^3b_{3,i}(t) \end{pmatrix}. \tag{7}$$

This matrix, (7), is for degree 3. A general expression for the B-spline-Hermite matrix (or Bernstein-Hermite matrix in Bézier case) is:

$$M_{d,i}(t) \in \mathbb{R}^{\,d+1\times d+1} \tag{8}$$

An algorithm for computing the B-spline-Hermite matrix follows. In Bézier/Bernstein case we replace $w_{d,i}(t)$ with $t$.

```
Matrix⟨double⟩ M( int d, int i, double t )
    Matrix⟨double⟩ 𝔐(d+1,d+1);                    // The return matrix, dimension (d + 1) × (d + 1).
    𝔐_{d−1,0} = 1 − w_{1,i}(t);
    𝔐_{d−1,1} = w_{1,i}(t);                      // The general Cox/deBoor like algorithm for
    for ( int r=d-2, s=i-d+1; r ≥ 0; r − − )     // - B-spline/Bernstein, computing the triangle
        𝔐_{r,0} = (1 − w_{d−r,s+r}(t)) 𝔐_{r+1,0}; // - of all values from B-splines
        for ( int j=1; j < d − r; j++ )          // - of degree 1 to d, respectively in each row.
            𝔐_{r,j} = w_{d−r,s+r+j−1}(t) 𝔐_{r+1,j−1} + (1 − w_{d−r,s+r+j}(t)) 𝔐_{r+1,j};
        𝔐_{r,d−r} = w_{d−r,i}(t) 𝔐_{r+1,d−r−1};
    𝔐_{d,0} = −δ_{1,i};
    𝔐_{d,1} = δ_{1,i};                           // Multiply all rows except the upper one
    for ( int k=2; k ≤ d; k++ )                  // - with the derivative matrices in the
        for ( int r = d; r > d − k; r − − )      // - expression (6), and the degree,
            𝔐_{r,k} = k δ_{k,i−k+1} 𝔐_{r,k−1};    // - so every row extends the number
            for ( int j = k − 1; j > 0; j − − )  // - of elements to d+1.
                𝔐_{r,j} = k (δ_{k,i−j} 𝔐_{r,j−1} − δ_{k,i−j+1} 𝔐_{r,j});
            𝔐_{r,0} = −k δ_{k,i} 𝔐_{r,0};
    return 𝔐;
```

In the algorithm is $\delta_{d,i} = w'_{d,i}(t) = (t_{i+d} - t_i)^{-1}$. The table below shows the number of multiplications/divisions depending on the degree d. The order of the algorithm can be computed to be exact $O\left(\frac{1}{3}d^3 + 2d^2 + \frac{14}{3}d - 5\right)$.

| degree | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| multiplications | 2 | 15 | 36 | 67 | 110 | 167 | 240 | 331 | 442 | 575 |

# B-splines and pre-evaluation

**The domain of a B-spline function -** is determined by the first and the last value of an increasing knot vector. Given a B-spline curve $c(t) = \sum_{i=0}^{n-1} c_i\, b_{d,i}(t)$, where

- ◊ $d$ is the polynomial degree,
- ◊ $\{c_i\}_{i=0}^{n-1}$ is the control polygon, that is $n$ control points typically in $\mathbb{R}^2$ or $\mathbb{R}^3$,
- ◊ and $\{t_i\}_{i=0}^{n+d}$ is the knot vector, a sequence of $n + d + 1$ increasing (or equal) real values.

The basis functions $b_{d,i}(t)$ is determined by the knot vector. For B-spline, the partition of unity is important, which means that the basis functions always sums up to 1. Thus, it follows that we must have $d + 1$ basis functions that is different from zero between knots. Therefore, the actual domain of a B-spline curve is restricted to $[t_d, t_n]$.

**Where to sample -** on a curve or surface is a question we have to address. This because, to plot a curve or a surface we need to tessellate.

- We first decide the number of samples, $m$.
- We then need a vector with $m$ parameter values, denoted $\mathcal{S}$ that shows where to sample.

A uniform distribution is an example of a vector of parameter values for sampling:

$$\mathcal{S} = \{t_d + j\, dt\}_{j=0}^{m-1}, \quad \text{where } dt = \frac{t_n - t_d}{m-1},\ d \text{ is the polynomial degree},\ n \text{ is the number of control points},\ t_i \text{ is a knot.}$$

**Pre-evaluation data and algorithms -** for computing them is the next step. Pre-evaluation data includes:

- ◊ A vector $\mathbf{I} = \{s, e\}_{i=0}^{n-1}$, two indices, where $\begin{cases} \mathcal{S}_{I_{i-1,s}} < t_i \le \mathcal{S}_{I_{i,s}} \\ \mathcal{S}_{I_{i,e}} \le t_{i+k} < \mathcal{S}_{I_{i+1,e}} \end{cases}$ and $t_i$ is a knot and $i = 0, n-1$ must be handled.
- ◊ A vector $\mathcal{M} = \{M_{i,d}(\mathcal{S}_j)\}_{j=0}^{m-1}$, where $M_{i,d}(t)$ is defined in (8), and where parameter $i$ follows from $\mathbf{I}_{i,s} \le j \le \mathbf{I}_{i+1,s}$.

Algorithms to create an index vector $\mathbf{I}$ follows om left hand side below, and to create the vector of B-spline Hermite matrices $\mathcal{M}$ to the right.

```
for ( int i = 0, s = 0, e = 0; i < n; i++ )
    while (S_s < t_i)  s++;
    I_{i,s} = s;
    while (S_{e+1} ≤ t_{i+k} && e < m − 1)  e++;
    I_{i,e} = e;
```

```
for ( int j = 0, i = d; j < m; j + + )
    while (t_{i+1} < S_j && i < n − 1)  i++;
    M_j = M(d, i, S_j);
```

**The initial sampling –** is straight forward. We denote the sampling vector for $\mathbf{P}$. The sampling vector $\mathbf{P}$ has $m$ elements and each element is a vector of 3D-vectors. The first 3D-vector is the position, the next is the first derivative, then the second derivative, ... until the derivative with the desired order. The formula for the computation is quite simple,

$$\mathbf{P}_j = \mathcal{M}_j * \mathbf{c}$$

where $\mathbf{P}_j$ is element number $j$ in $\mathbf{P}$, $\mathcal{M}_j$ is a B-spline Hermite matrix from the pre-evaluation data and $\mathbf{c}$ is a selection of control points, where the selection is decided in the index vector $\mathbf{I}$ from the pre-evaluation data.

An algorithm for an initial computing of a sampling vector $\mathbf{P}$ is,

- $\mathbf{I}$ and $\mathcal{M}$ are the pre-evaluation data described above,
- $g$ the number of subsequent derivatives to compute,
- and $d$ is the polynomial degree.

```
for ( int  j = 0,  i = d; j < m; j + + )
    while (I_{i+1,s} == j && i < n − 1)  i++;
    for ( int  r = 0; r ≤ g; r + + )
        P_{j,r} = M_{j,r,0} * c_{i−d};
        for ( int  s = 1; s ≤ d; s + + )
            P_{j,r} + = M_{j,r,s} * c_{i−d+s};
```

**Tensor product surfaces -** with the following formula

$$s(u,v) = \sum_{i=0}^{n_1-1} \left( \sum_{j=0}^{n_2-1} c_{i,j}\, b_{d_2,j}(v) \right) b_{d_1,i}(u),$$

is comparable with curves. We only have to double the data sets. Thus we need

- two vectors of parameter values, $\mathcal{S}^u$ and $\mathcal{S}^v$, to show where to sample in the two parameter directions,
- two vectors of indices, $\mathrm{I}^u$ and $\mathrm{I}^v$, of size $n_u$ and $n_u$ to indicate the domain of the basis functions,
- two vectors of matrices, $\mathcal{M}^u$ and $\mathcal{M}^v$ with elements $M(d,i,t)$, one for each parameter direction.

The sampling $\mathbf{P}$ is here a matrix. This sampling matrix has $m_u \times m_v$ elements, where each element is a matrix of 3D-vectors. The 3D-vector with index (0,0) is the position, the partial derivatives is organized in the following way

$$M_{3,i}(t) = \begin{pmatrix} S & S_v & S_{vv} \\ S_u & S_{uv} & S_{uvv} \\ S_{uu} & S_{uuv} & S_{uuvv} \end{pmatrix}.$$

The formula is just an extension of the formula for curves, the formula is,

$$\mathbf{P}_{i,j} = \mathcal{M}_i^u * \mathbf{c} * \mathcal{M}_j^{vT},$$

where $\mathbf{P}_{i,j}$ is an element in $\mathbf{P}$ with indices $(i,j)$, $\mathcal{M}_i^u$ and $\mathcal{M}_j^v$ are B-spline Hermite matrices from the pre-evaluation data, and $\mathbf{c}$ is a sub matrix, a selection of control points, where the selection is described in the index vectors $\mathbf{I}^u$ and $\mathbf{I}^v$ from the pre-evaluation data.

The algorithm is the same as for curves except for that we have go through the algorithm two times, and we need to store the result of the first time in a temporary matrix of 3D-vectors.

## Interactive editing and smooth re-plotting of B-splines

In interactive editing, we select and move control points directly on the computer screen. We want the graphics to be continuously updated in a smooth way, al least 30 times a second. Therefore, we want to significantly reduce the number of operations when updating the graphics.

If we have pre-evaluated and are doing interactive editing by moving control points, we have the following algorithm to update the sampling vector $\mathbf{P}$ in the curve case:

Control point $c_i$ is moved, then an algorithm for updating is,

- $\mathbf{I}$ and $\mathcal{M}$ are the pre-evaluation data described in the previous section,
- $g$ is the number of subsequent derivatives to compute,
- $\triangle\, c_i$ is the translation of the control point $c_i$.

```
for ( int  j = I_{i,s}, k = 0; j ≤ I_{i,e}; j++ )
    if (j == I_{i+k,s})  k++;
    for ( int  r = 0; r ≤ g; r++ )
        P_{j,r} += M_{j,r,d-k} * △ c_i;
```

We can observe that the computation cost is reduced dramatically. If we compare the computation cost when using pre-evaluation and optimal computational structure with an ordinary replot, that is an initial computing a curve or surface, we see the following:

- The structure reduce the computation to $\frac{1}{n}*$ (original computation), where n is the number of control points
- The pre-evaluation reduce the computation to approximately $\frac{1}{\frac{1}{3}d^3+2d^2+\frac{14}{3}d-5}*$ (original computation), where d is the polynomial degree

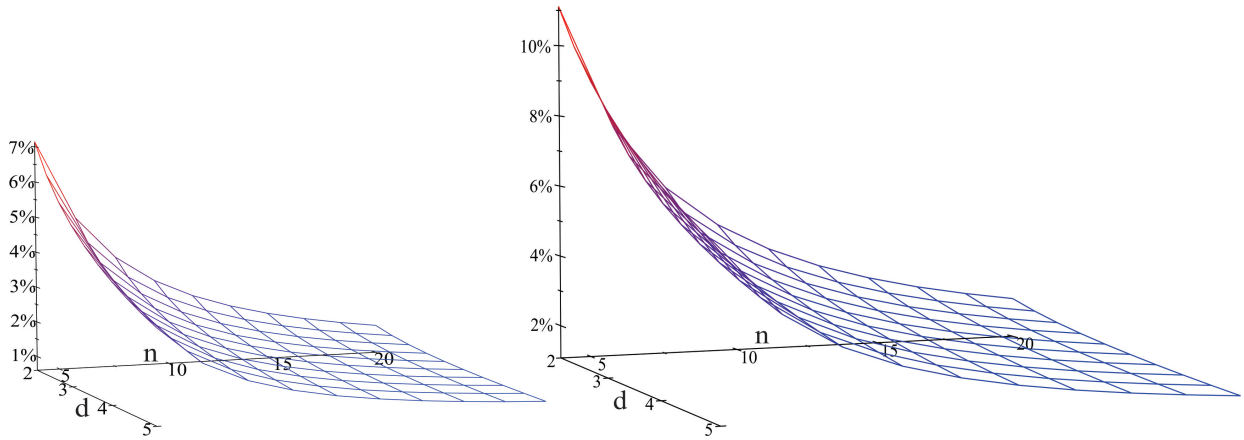


**FIGURE 2.** We have two plots that both describe the efficiency of pre-evaluation and optimal organization. To the left is a plot for only recalculating the position, to the right is also first derivatives included. For both plots is the right axis (towards us) for the polynomial degree $d$, the left axis is for the number of control points, $n$. The hight is the percentage to a complete recalculation.

We usually only need the position to plot a curve. If so, we get the following expression for the percentage resource usage by re-plotting, in conjunction with the initial plotting.

$$\frac{3dm}{n\left(\left(\frac{1}{3}d^3+2d^2+\frac{14}{3}d-5\right)m+3dm\right)}=\frac{3d}{n\left(\frac{1}{3}d^3+2d^2+\frac{23}{3}d-5\right)}.$$

For a surface we usually want shading. This means that we need normals and thus the first-order partial derivatives. We then get the following expression for the percentage resource use,

$$\frac{6dm}{n\left(\left(\frac{1}{3}d^3+2d^2+\frac{14}{3}d-5\right)m+6dm\right)}=\frac{6d}{n\left(\frac{1}{3}d^3+2d^2+\frac{32}{3}d-5\right)}.$$

In Figure 2 is two plots showing the percentage resource usage when we use pre-evaluation and optimal organization before re-plotting. The number of operations depends on the polynomial degree and the number of control points. On left hand side is only the function value calculated, but on right hand side is also the first derivatives calculated. For shading of surfaces we need normals, and thus the two first order partial derivatives.

An example is that if we move one control point then the cost of recomputing a curve of polynomial degree 3 with 10 control points is reduced to approximately 4% of the original computation. Another example is a surface of degree 3 and with $10 \times 10$ control points. If we move one point the cost is less than 0.5% of the original computation.

## ERBS

ERBS - Expo-Rational B-splines were first mentioned in [5] and [6], and then among others in [7]. In [8], Generalized Expo-Rational B-splines are described using B-functions. The formula for a simple version of an Expo-Rational B-function is,

$$B(t) = 1.65713768 \int_0^t e^{-\frac{\left(t-\frac{1}{2}\right)^2}{t(1-t)}}\, ds, \tag{9}$$

In expression (5), a factorisation of a B-spline curve is described. In this expression we find $w_{d,i}(t)$ for $i = 1, 2, 3$. If we now replace $w_{d,i}(t)$ with $B \circ w_{d,i}(t)$ we get an Expo-Rational B-spline curve.

Because of the infinite smoothness of the Expo-Rational B-function, it has been possible to use $2^{\text{nd}}$ order B-spline curves and surfaces in a blending-spline construction. Here we replace control points with local curves or surfaces. The basis functions now have a minimal support, over two knot intervals, and we can not only translate control points, but also rotate and scale them, because they now are local curves or surfaces.

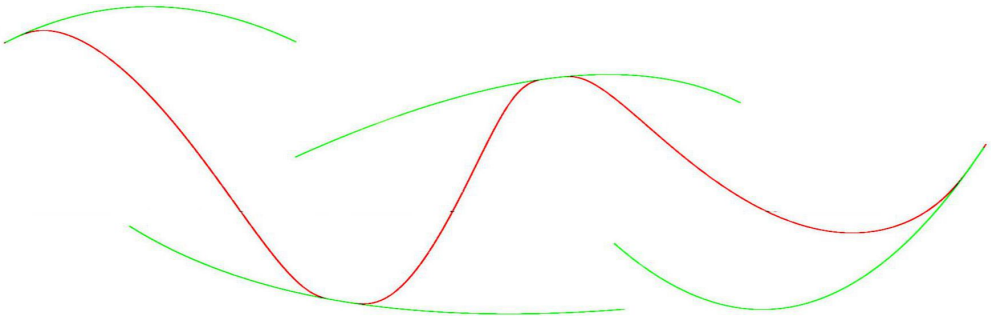

**FIGURE 3.** An Expo-Rational B-spline curve plotted in red, together with its four local curves plotted in green.

The formula for an ERBS-curve of blending type is,

$$c(t) = \sum_{i=0}^{n-1} c_i(t)B_i(t) \qquad \text{where} \quad B_i(t) = \begin{cases} B \circ w_{1,i}(t), & t_i \le t < t_{i+1}, \\ 1 - B \circ w_{1,i+1}(t), & t_{i+1} \le t < t_{i+2}, \end{cases}$$

or factorized as in expression (5 ),

$$c(t) = \begin{pmatrix} 1 - B \circ w_{1,i}(t) & B \circ w_{1,i}(t) \end{pmatrix} \begin{pmatrix} c_{i-1}(t) \\ c_i(t) \end{pmatrix}, \qquad \text{when} \quad t_i \le t < t_{i+1}. \tag{10}$$

The B-function in (10) does not only need to be of Expo-Rational type, there are several other types, Beta-functions, rational function, trigonometric functions, and there are also several other types of Expo-Rational B-functions, all described in [9] and [8].

In Figure 3,an ERBS curve of blending type is plotted. We can see that the curve can be divided in three parts. Each part is the result of blending parts of two green curves. The knot vector of the curve is $\mathbf{t} = \{0, 0, 1, 2, 3, 3$, which then has three knot intervals.

## ERBS and pre-evaluation

The basis function for ERBS is $B \circ w_{1,i}(t)$, where $B$ is defined in (9). For pre-evaluation we define the vector,

$$\mathbf{B}_g(t) = \begin{pmatrix} B \circ w_{1,i}(t) \\ \delta_{1,i} B' \circ w_{1,i}(t) \\ \vdots \\ \delta_{1,i}^g B^{(g)} \circ w_{1,i}(t) \end{pmatrix} \quad \text{where} \quad \delta_{1,i} = w'_{1,i}(t) = \frac{1}{t_{i+1} - t_i}. \tag{11}$$

An Expo-Rational B-spline curve or surface is actually a B-spline curve or surface of order 2, analogous to polynomial degree 1. It has a knot vector, and a vector of local curves instead of control points as polynomial B-splines have. Thus, the domain of a blending spline curve is restricted to $[t_1, t_n]$. For pre-evaluation we have to

- decide the number of samples, $m$.
- We make a vector with $m$ parameter values, denoted $\mathcal{S}$, that shows where to sample. We now need
- a vector $\mathbf{I} = \{s, e\}_{i=0}^{n-1}$, two indices, where $\begin{cases} \mathcal{S}_{I_{i-1,s}} < t_i \le \mathcal{S}_{I_{i,s}} \\ \mathcal{S}_{I_{i,e}} \le t_{i+2} < \mathcal{S}_{I_{i+1,e}} \end{cases}$ and $t_i$ is a knot and $i = 0, n-1$ must be handled,
- a vector $\mathcal{B} = \{B_g(\mathcal{S}_j)\}_{j=0}^{m-1}$, where $\mathbf{B}_g(t)$ is defined in (11), and the number of derivatives $g$ must be chosen.

The algorithms are basically the same as for B-splines, but there are some differences.

- Pre-evaluation must be done both in the central ERBS function and in the local curves/surfaces.
- There must be a map between the indices of pre-evaluation data for the central ERBS and the indices of pre-evaluation data for the local curves/surfaces.

## Interactive editing and smooth re-plotting of ERBS

In interactive editing of blending splines, we first select a local curve/surface, then move, rotate or scale them directly on the computer screen. Also now we want the graphics to be continuously updated in a smooth way, for example 30 times a second. Therefore, it is even more important than in the B-spline case to reduce the number of operations when updating the graphics.

A translation, rotation, scaling is an affine operation, and an affine operation is represented by a homogenous transformation matrix. The sampling data, position and derivatives from a local curve/surface must be multiplied by this matrix. We have initially made a sampling vector/matrix $\mathbf{P}$ for the ERBS curve/surface and respective sampling vectors/matrices in all local curve/surfaces.

If a transformation is detected on for example a local curve $c_i(t)$. Then the Matrix $M = M_1 - M_0$, the difference between the previous homogenous matrix and the new homogenous matrix, both representing the position and orientation of the local curve, must be multiplied with the pre-evaluated sampling data from the local curve $c_i(t)$. An algorithm for updating the samplings vector $\mathbf{P}$ is,

- $\mathbf{I}$ and $\mathcal{B}$ are the pre-evaluation data described in the previous section,
- $g$ is the number of subsequent derivatives to compute,
- $\triangle c_i$ is the pre-evaluated date from the local curve multiplied with the difference matrix.

$$
\begin{aligned}
&\textit{for}\ (\ \textit{int}\ \ j = \mathbf{I}_{i,s};\ j < \mathbf{I}_{i+1,s};\ j++\ )\\
&\quad \mathbf{P}_{j,0} \mathrel{+}= (1 - \mathcal{B}_{j,0}) * \triangle c_{i,0};\\
&\quad \textit{for}\ (\ \textit{int}\ \ r = 1;\ r \le g;\ r++\ )\\
&\quad\quad \mathbf{P}_{j,r} \mathrel{-}= \mathcal{B}_{j,r} * \triangle c_{i,r};\\
&\textit{for}\ (\ \textit{int}\ \ j = \mathbf{I}_{i+1,s};\ j \le \mathbf{I}_{i,e};\ j++\ )\\
&\quad \mathbf{P}_{j,0} \mathrel{+}= \mathcal{B}_{j,0} * \triangle c_{i,0};\\
&\quad \textit{for}\ (\ \textit{int}\ \ r = 1;\ r \le g;\ r++\ )\\
&\quad\quad \mathbf{P}_{j,r} \mathrel{+}= \mathcal{B}_{j,r} * \triangle c_{i,r};
\end{aligned}
$$

We can observe that the computation cost is significant reduced, and about the same as for polynomial B-splines, which means that interactive editing with smooth changes is possible even with geometry with large data sets

## ACKNOWLEDGMENTS

The *R&D group in mathematic and geometric modeling, numerical simulations, programming and visualization* at UiT - The Arctic University of Norway has worked with splines as long as the group has existed, over 20 years, and ERBS for 14 year. This work is the backbone for this paper. Therefore, thanks to my colleagues Lubomir Dechevesky, Børre Bang, Rune Dalmo and Jostein Brattlie, former Phd students Joakim Gundersen, Arnt Kristoffersen, Peter Zanaty, today Phd students Aleksander Pedersen, Tanita Fossli Brustad, Tatiana Kravetc and Hans Olofsen.

## REFERENCES

[1] B. H. Curry and I. J. Schoenberg, "On Pòlya frequency functions IV: The spline functions and their limits", Bull. Amer. Math. Soc. **53**, p. 1114 (1947), abstract 380t.

[2] B. H. Curry and I. J. Schoenberg, "On Pòlya frequency functions IV: The fundamental spline functions and their limits," J. d'Analyse Math. **17**, 71–107 (1966).

[3] C. de Boor, "On calculation with B-splines," in Journal of Approximation Theory **6**, 50–62 (1972).

[4] A. Lakså, "A method for sparse-matrix computation of b-spline curves and surfaces," in *Large-Scale Scientific Computing*, Lecture Notes in Computer Science, Vol. 5910, edited by I. Lirkov, S. Margenov, and J. Waśniewski (Springer Berlin Heidelberg, 2010), pp. 796–804.

[5] A. Lakså, B. Bang, and L. T. Dechevsky, "Exploring expo-rational B-splines for curves and surfaces," in *Mathematical methods for curves and surfaces: Tromsø 2004*, Mod. Methods Math. (Nashboro Press, Brentwood, TN, 2005), pp. 253–262.

[6] L. T. Dechevsky, A. Lakså, and B. Bang, "Expo-Rational B-splines," International Journal of Pure and Applied Mathematics **27**, 319–369 (2006).

[7] A. Lakså, *Basic properties of Expo-Rational B-splines and practical use in Computer Aided Geometric Design*, unipubavhandlinger No. 606 (Unipub, Oslo, 2007).

[8] A. Lakså, "Non polynomal b-splines," (2015) p. 030001, http://aip.scitation.org/doi/pdf/10.1063/1.4936700 .

[9] L. T. Dechevsky, B. Bang, and A. Lakså, "Generalized Expo-Rational B-splines," International Journal of Pure and Applied Mathematics **57**, 833–872 (2009).