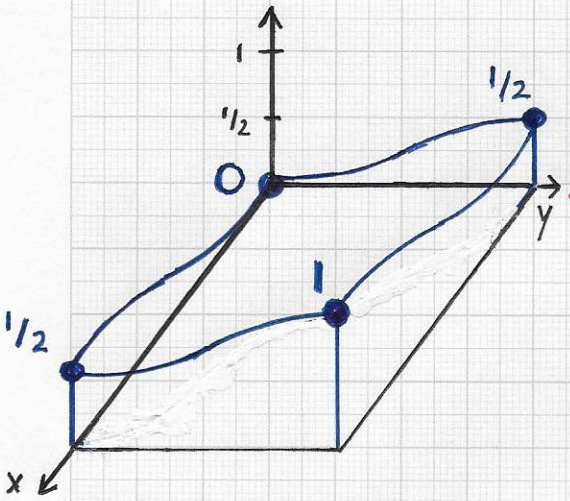
## ■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

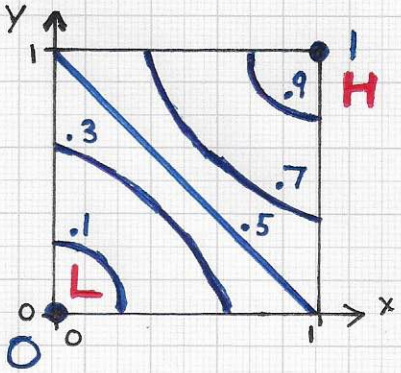
• Laplacian eigenfunctions and neural networks: ...




Sketch of graph of "tensor sum"

$F(x,y) = (H_1(x) + H_1(y))/2$

$= (-2x^3 + 3x^2 - 2y^3 + 3y^2)/2.$




Sketch of contours (iso-lines) of $F(x,y)$. The contours indicate the close-to-0 (L) and the close-to-1 (H) values and respective regions.

One can view $F(x,y)$ as an implementation of a "FUZZY OR."

In summary, the constructed overall bivariate decider function $F(x,y)$ is the tensor product of the two univariate Hermite polynomials $H_1(x) = -2x^3 + 3x^2$ and $H_1(y) = -2y^3 + 3y^2$. Further, since $F(x,y) = H_1(x) \cdot H_1(y)$ is a PRODUCT of functions, $F(x,y)$ effectively implements an approximation of the **AND** Boolean function. If one wanted to construct a function $F(x,y)$ that implements a numerical approximation of the **OR** Boolean function, one would have to use a different combination operator for $H_1(x)$ and $H_1(y)$. For example, one could define the combination as $F(x,y) = (H_1(x) + H_1(y))/2$, illustrated in the figures (left, top). An alternative for the approximation of the **OR** Boolean function — better approximation — is obtained by using the following combination:

$$F(x,y) = 1 - (-2(1-x)^3 + 3(1-x)^2) \cdot (-2(1-y)^3 + 3(1-y)^2)$$

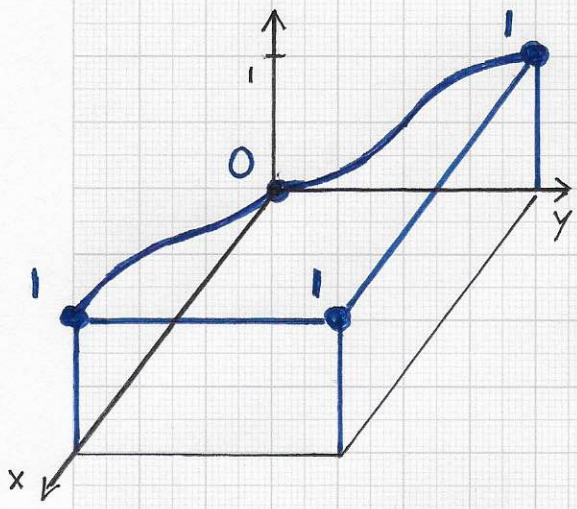
$$= 1 - H_1(\bar{x}) \cdot H_1(\bar{y}),$$

where $\bar{x} = 1-x$ and $\bar{y} = 1-y$.

...

## ■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

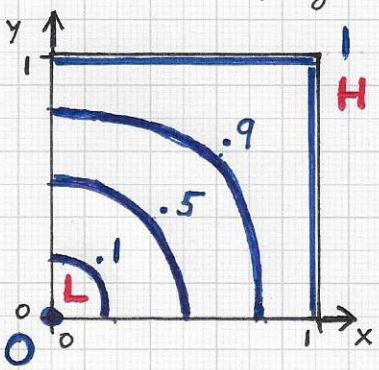
• Laplacian eigenfunctions and neural networks:...




Sketch of the graph of

$F(x,y) = 1 - H_1(\bar{x}) \cdot H_1(\bar{y})$

(see previous page).




Sketch of contours (iso-Lines) of $F(x,y)$. These contours indicate that the area of the close-to-1 (H) region is enlarged.

One can view $F(x,y)$ as another implementation of a "FUZZY OR."

The last definition of the bivariate decider function $F(x,y)$ can be interpreted as yet another numerical approximation of the Boolean OR function; its behavior is sketched in the figures (left). This definition of $F(x,y)$ satisfies the equations $F(x,0) = H_1(x)$ and $F(0,y) = H_1(y)$, in addition to $F(x,1) = F(1,y) = 1$. This behavior of $F(x,y)$ makes it a "strong" numerical realization of the OR function. This discussion of decider functions has used "x" and "y" as variables — instead of using p-variables — in order to cover and describe construction and function behavior in a general setting (but still related to Boolean logic).

• **Note.** In the context of actual material classes and classification, one must keep in mind that the functions $F_i(p_i)$ are SCALE-SPECIFIC ($i = sc = 1 \ldots H$); one must consider a scale's importance. ...

## ■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...

Viable and desirable combinations of **LOW** and **HIGH** values of two univariate decider functions $F_1(p_1)$ and $F_2(p_2)$:

| $F_1(p_1)$ | $F_2(p_2)$ | combination | | | |
|---|---|---|---|---|---|
| $L_1$ | $L_2$ | L | L | L | L |
| $H_1$ | $L_2$ | L | H | L | H |
| $L_1$ | $H_2$ | L | L | H | H |
| $H_1$ | $H_2$ | H | H | H | H |

Theoretically, $2^4 = 16$ combination quadruples are possible. The table only includes the four combination quadruples that are practically relevant and meaningful for material classification.

**LOW** and **HIGH** must be viewed in the context of the values of $F_1$ and $F_2$ and the scales, i.e., scales 1 and 2. Thus, $L_1$ and $L_2$ ($H_1$ and $H_2$) are semantically different. This becomes evident in combinations two and three of the four viable quadruples.

The values of $L_1$, $L_2$ and L are assumed to be close to 0, and the values of $H_1$, $H_2$ and H are assumed to be close to 1.

The combinatorial table included here (left) lists the four "logically meaningful" ways to combine **LOW** and **HIGH** values of two functions $F_1(p_1)$ and $F_2(p_2)$. **Two values $L_1$ and $L_2$ should always produce another L value; two values $H_1$ and $H_2$ should always produce another H value. BUT: Combining $H_1$ and $L_2$ can produce an L or H value; combining $L_1$ and $H_2$ can produce an L or H value.** Thus, these combinations are context-sensitive. Considering the relationship to Boolean logic, the four **meaningful** combinations are

$$(L, L, L, H)^T = F_1 \wedge F_2$$
$$(L, H, L, H)^T = F_1$$
$$(L, L, H, H)^T = F_2$$
$$(L, H, H, H)^T = F_1 \vee F_2$$

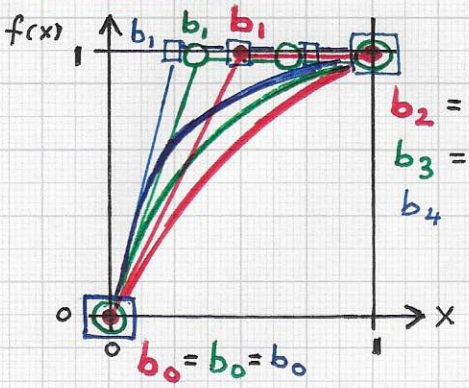
• **Note.** This discussion did not consider the "RAMP" parts of decider functions. They must be treated properly in all constructions.

...

## ■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...

Potentially, one can use polynomials in Bernstein-Bézier form, or rational functions defined by Bernstein-Bézier polynomials, to represent decider functions:




Quadratic, cubic and quartic Bernstein-Bézier polynomials with coefficients / control points

$\{b_i\}_{i=0}^{2}$, $\{b_i\}_{i=0}^{3}$, $\{b_i\}_{i=0}^{4}$.

The actual geometrical control POINTS sketched in the figure have coordinate tuples

$(x_i, b_i)^T = (i/n, b_i)^T$, $i = 0 \ldots n$.

These points define the three control polygons shown in the illustration. The sketched graphs of the three polynomial functions resemble the control polygons.

Of course, the fact that univariate decider functions $F_{sc}(p_{sc})$ relate to scale-specific material class characteristics makes their proper combination context-sensitive. Therefore, combinations must consider answers to questions like the following: What is the goal? How important are certain scales relative to other scales? What kind of Boolean logic operator / circuit would an expert employ when defining combinations?

• Note. For the definition and representation of univariate and overall, multivariate decider functions one can and should also consider Bernstein-Bézier polynomials and their rational extensions. They provide flexibility and numerical, computational stability. They are generally defined as

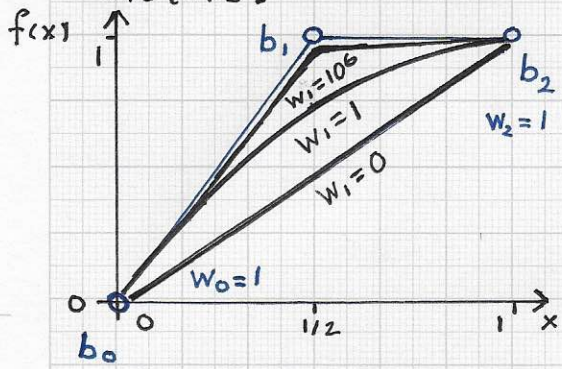
$$f(x) = \sum_{i=0}^{n} b_i B_i^n(x), \quad x \in [0,1],$$

where $B_i^n(x) = \binom{n}{i}(1-x)^{n-i} \cdot x^i$, $i = 0 \ldots n$.

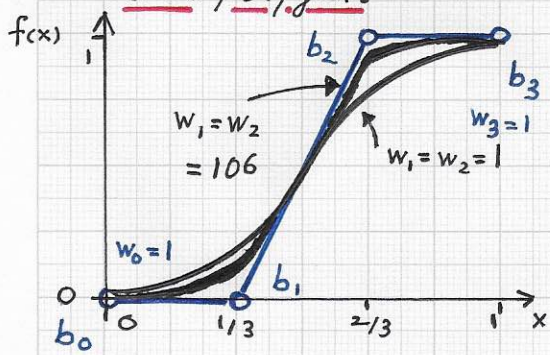
## ■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• *Laplacian eigenfunctions and neural networks:...*

Rational functions of Bernstein-Bézier polynomials permit the use of weights, effectively serving as shape parameters:




Sketch of rational-quadratic Bernstein-Bézier function, where weights $w_0$ and $w_2$ are constant (1) and $w_1$ varies from 0 to $10^6$. This example shows how a weight parameter, $w_1$, can be used to pull the graph of $f(x)$ towards the control polygon.




Sketch of rational-cubic Bernstein-Bézier function. Increasing the inner weights $w_1$ and $w_2$ relative to $w_0$ and $w_3$ makes the "RAMP" part steeper.

The more general definition uses rational functions, specifically functions of the form

$$f(x) = \sum_{i=0}^{n} w_i b_i B_i^n(x) \Big/ \sum_{i=0}^{n} w_i B_i^n(x),$$

$$x \in [0,1],\ w_i \geq 0,\ i = 0 \ldots n.$$

One can interpret the $w_i$ values as "weights"; high $w_i$ values "pull" the graph $(x, f(x))^T$ towards the associated control points $(i/n, b_i)^T$. This effect is sketched in the left figure (top) for a rational function defined by quadratic Bernstein-Bézier polynomials: $f = \sum_{i=0}^{2} w_i b_i B_i^2 / \sum_{i=0}^{2} w_i B_i$.

• **Note.** Via concepts from projective geometry it is possible to treat and calculate rational-polynomial Bernstein-Bézier functions of this kind like NON-rational functions. This can be achieved by performing the following steps:

i) define $\mathbb{b}_i^* = (w_i b_i, w_i)^T$.

ii) compute $\mathbb{f}^*(x) = \sum_{i=0}^{n} \mathbb{b}_i^* B_i^n(x)$.

iii) "project": $f(x) = f^*(x) / w^*(x)$, where $\mathbb{f}^*(x) = (f^*(x), w^*(x))^T$.

...