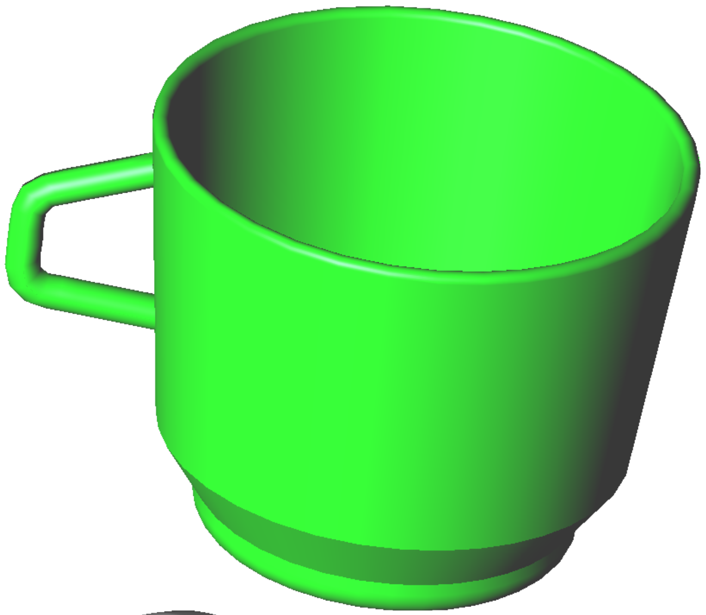
CIMAT

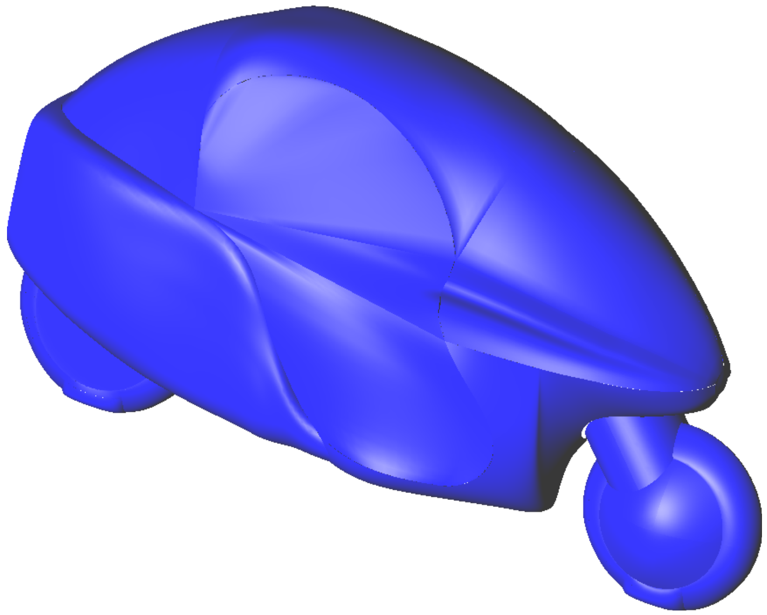
Isogeometric Analysis

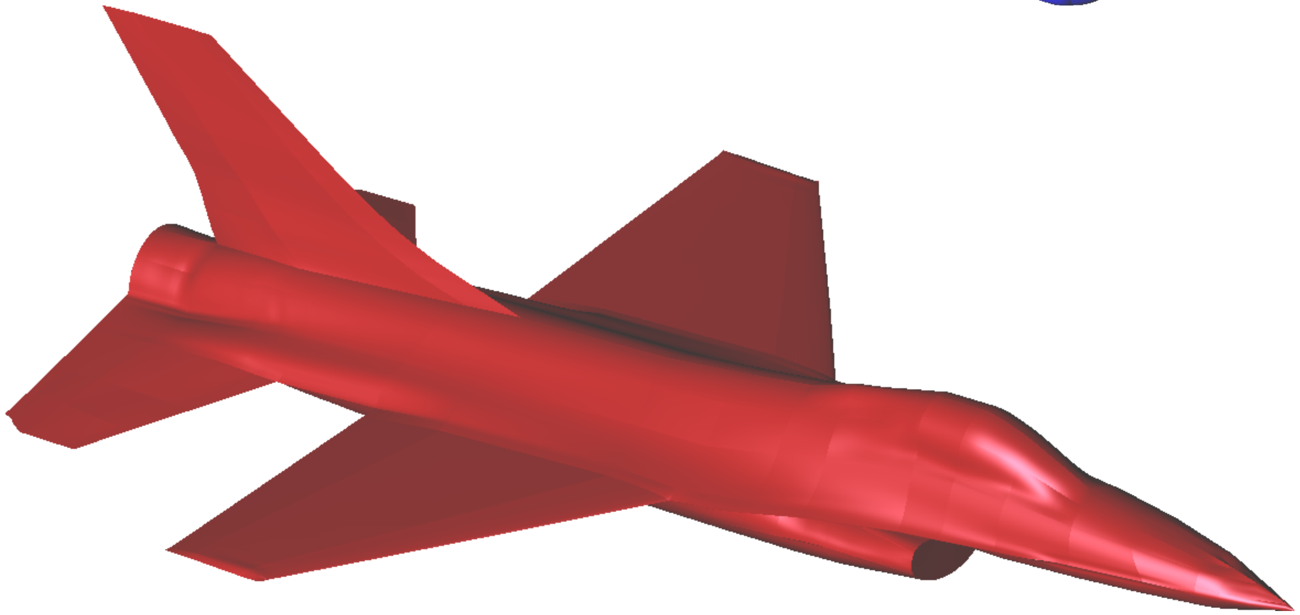
Miguel Vargas-Félix
[*miguelvargas@cimat.mx*](mailto:miguelvargas@cimat.mx)
[*http://www.cimat.mx/~miguelvargas*](http://www.cimat.mx/~miguelvargas)

NURBS (Non-Uniform Rational B-Spline)

These splines are used to describe **geometric designs** with complex shapes

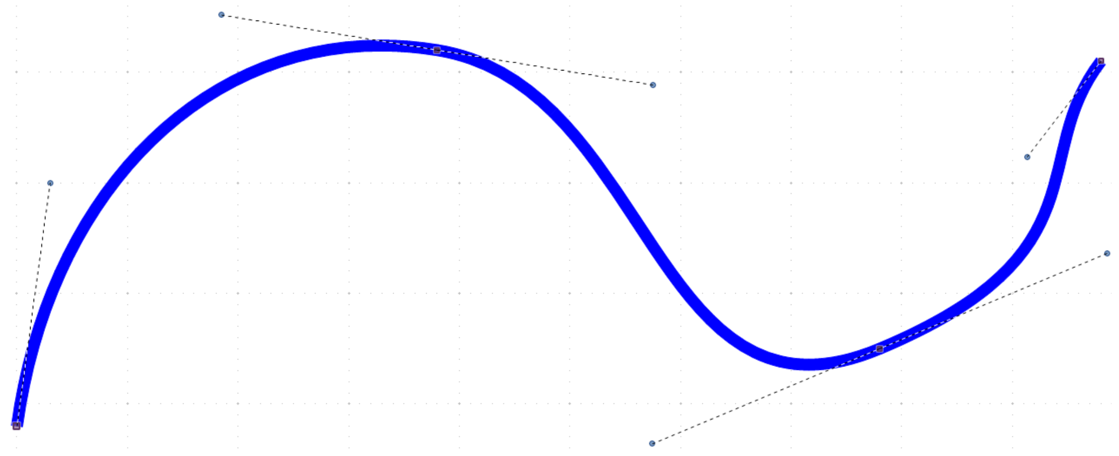


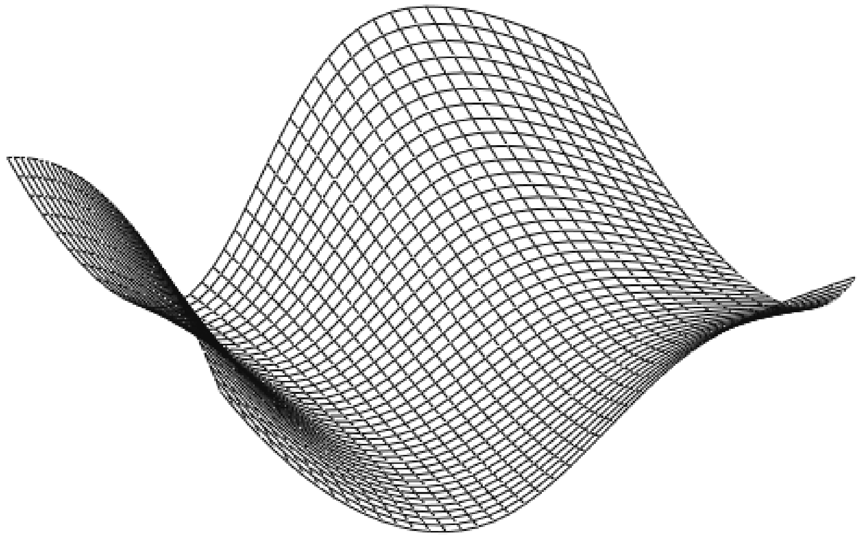




Why NURBS?

Splines are used because they are useful to describe **free-form shapes** in 2D or 3D.





There are several kind of splines to describe graphical objects

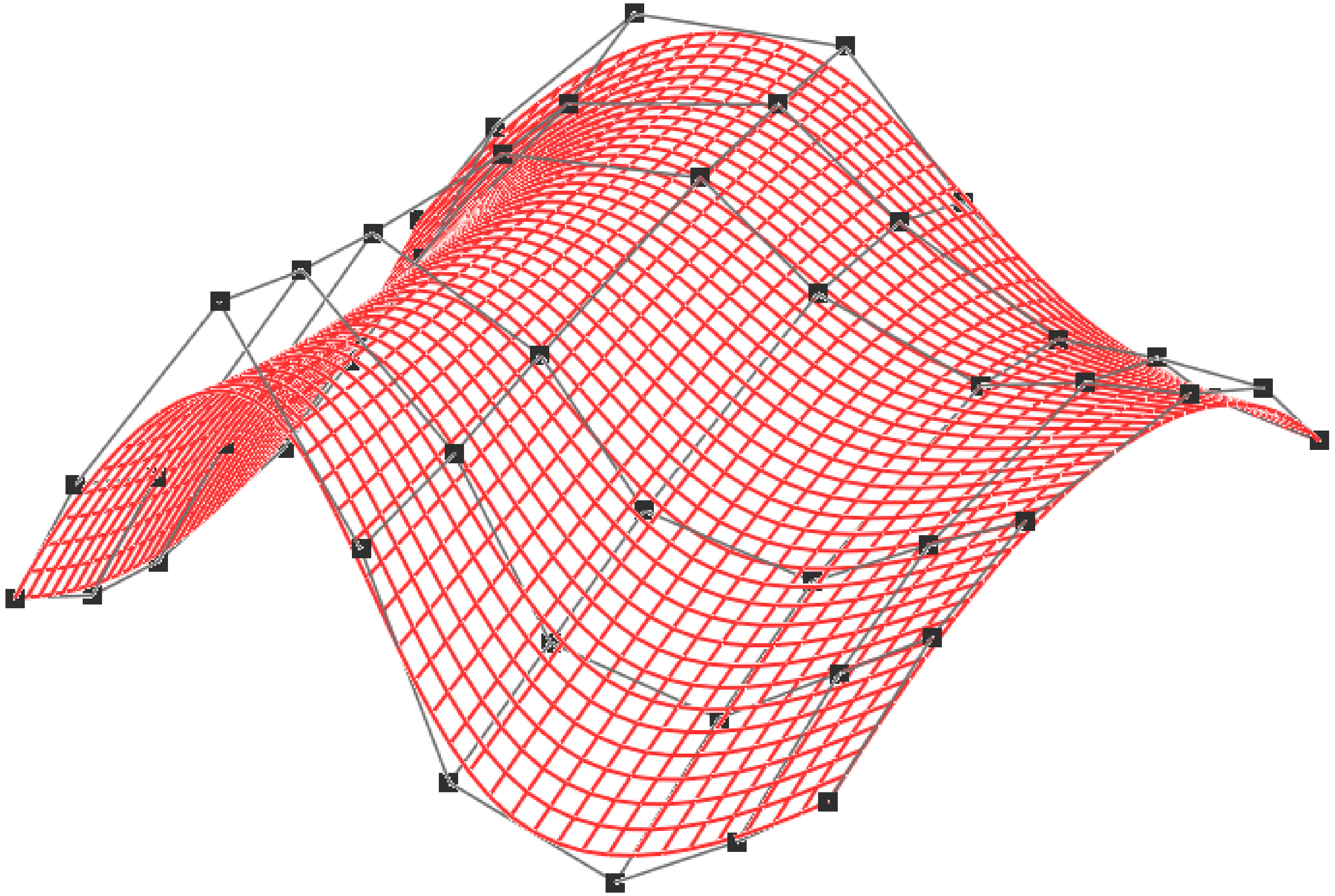
- Bézier splines (artistic design software)
- B-Splines (special case of NURBS)
- NURBS
- Others (Kochanek–Bartels splines, Hermite splines, etc)

The advantege of NURBS is that these give an **exact representation** of conic shapes and cuadratic surfaces.

NURBS are **invariant under affine transformations** (traslation, rotation, scalling, etc)

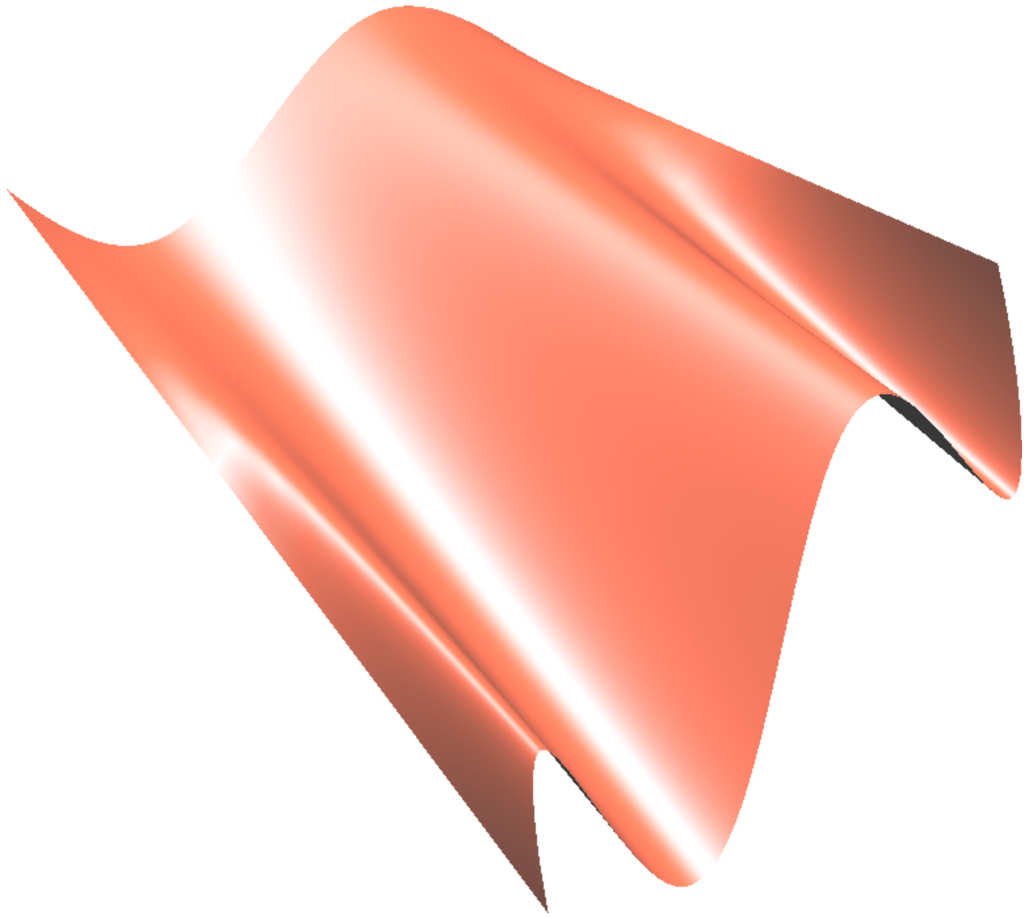
NURBS

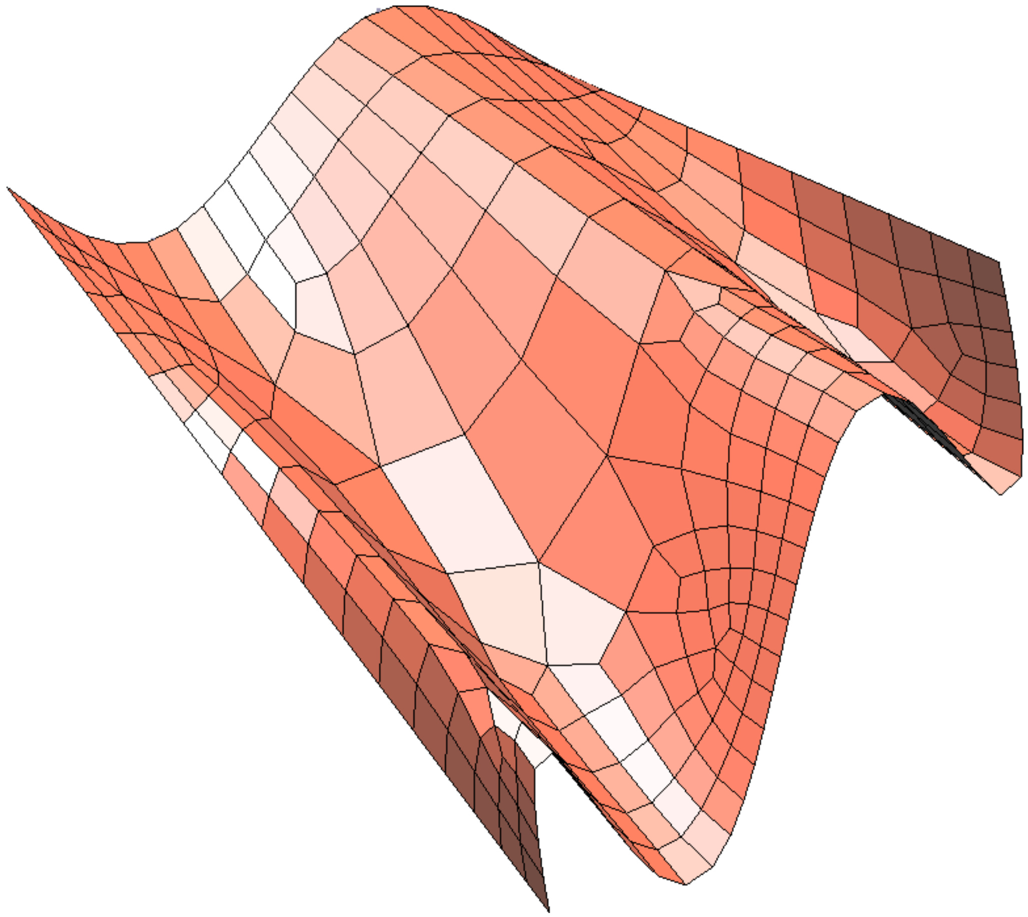
NURBS are defined by **control points**, these are coordinates that are used to manipulate the shape of the surface.



Finite element method

CAD programs (like GiD) use NURBS to create **smooth curves and surfaces**.



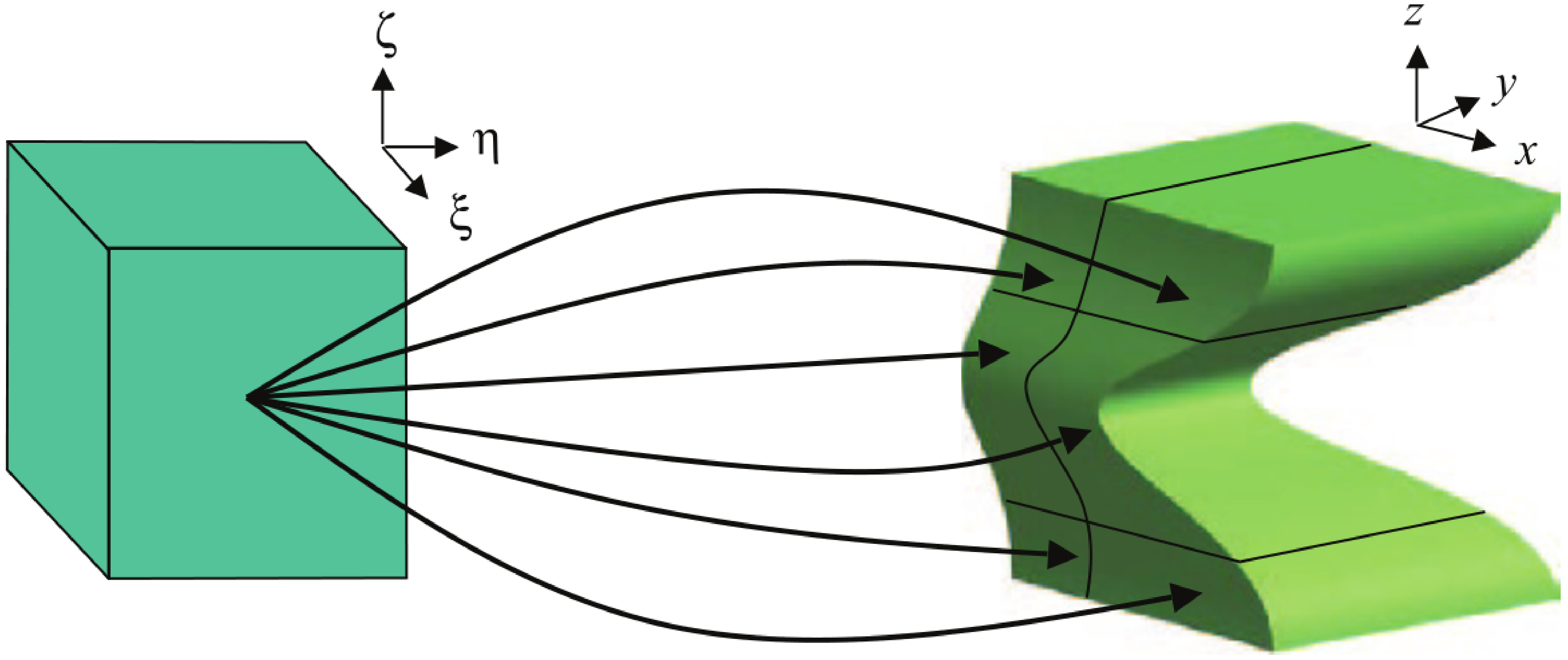


Finite element method uses a mesh that is an **approximation** to the real geometry. Each element is an independent geometric entity.

The original **geometry description is lost**.

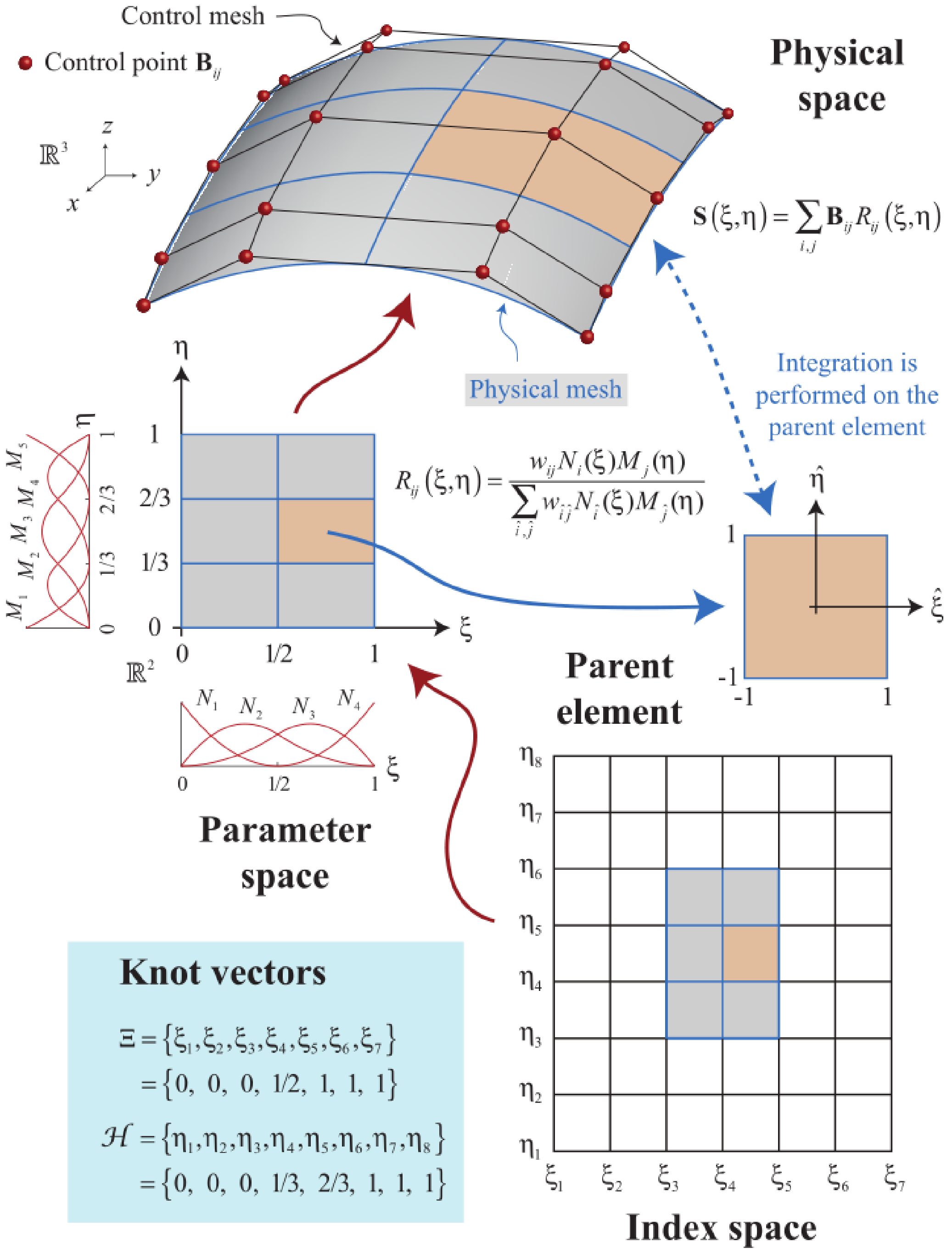
Discretization

In the **finite element method**, a domain is discretized into several elements, each element is mapped to an element in with normalized coordinates.



The normalized element (left) mapped to elements in the finite element discretization (right).

Isogeometric Analysis


Control mesh
Control point $\mathbf{B}_{ij}$
Physical space
$\mathbb{R}^3$
$\mathbf{S}(\xi,\eta)=\sum_{i,j}\mathbf{B}_{ij}R_{ij}(\xi,\eta)$
Physical mesh
Integration is performed on the parent element
$R_{ij}(\xi,\eta)=\frac{w_{ij}N_i(\xi)M_j(\eta)}{\sum_{i,j}w_{ij}N_i(\xi)M_j(\eta)}$
$\mathbb{R}^2$
M_1 M_2 M_3 M_4 M_5
N_1 N_2 N_3 N_4
Parameter space
Parent element
$\hat{\eta}$
$\hat{\xi}$
Index space
η_1 … η_8
ξ_1 … ξ_7

Knot vectors

$$\Xi=\{\xi_1,\xi_2,\xi_3,\xi_4,\xi_5,\xi_6,\xi_7\}=\{0,\ 0,\ 0,\ 1/2,\ 1,\ 1,\ 1\}$$

$$\mathcal{H}=\{\eta_1,\eta_2,\eta_3,\eta_4,\eta_5,\eta_6,\eta_7,\eta_8\}=\{0,\ 0,\ 0,\ 1/3,\ 2/3,\ 1,\ 1,\ 1\}$$

Created by Hughes, Cottrell and Bazilevs [Hugh05] is a recent development to overcome several limitations existing in the finite element method.

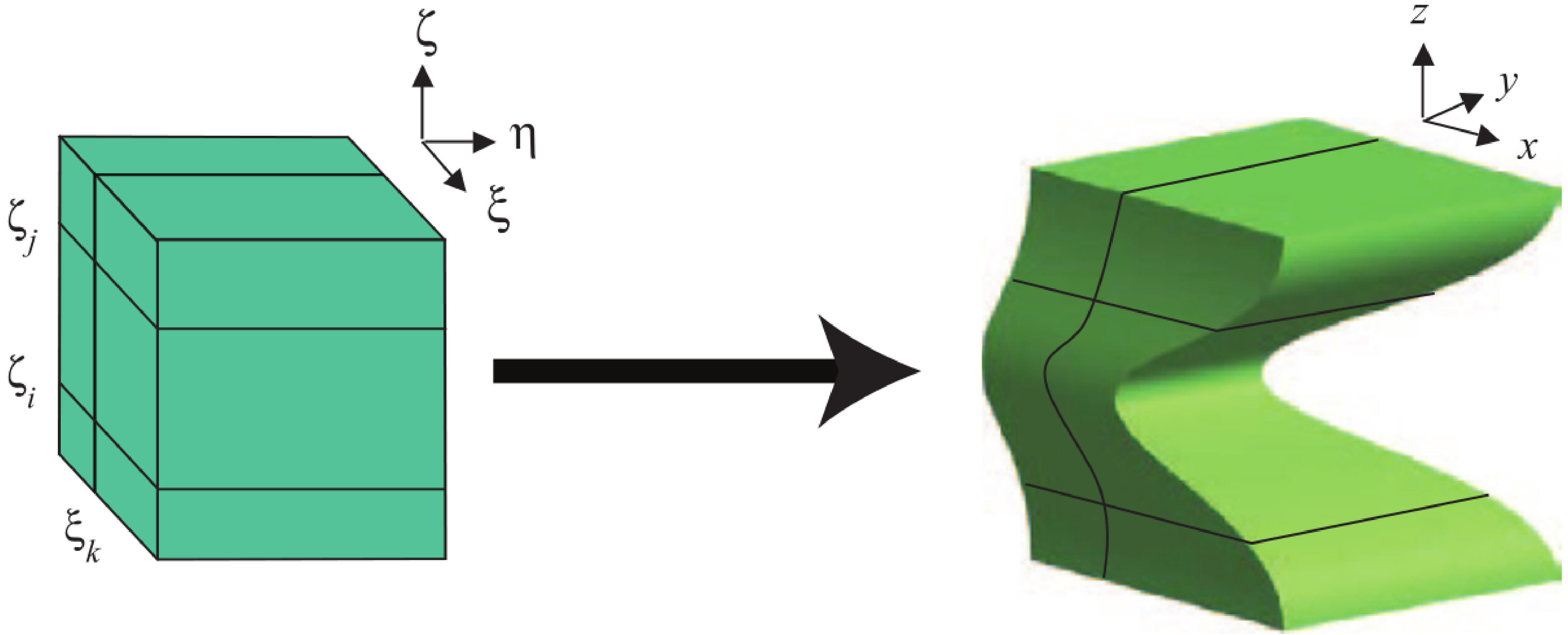
The idea of the Isogeometric Analysis (IGA) is to use **NURBS as definition of elements** in the sub-domain.

Keep geometry information in the simulation program.

By having the geometry the simulation can **mesh automatically** the geometry.

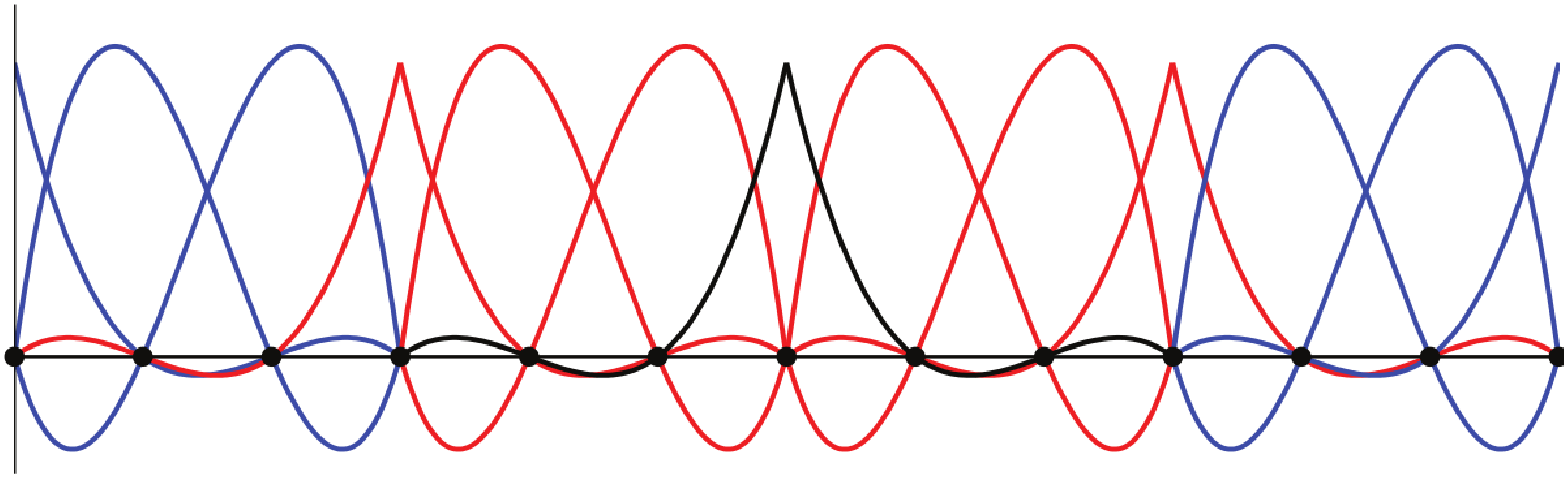
Discretization

Isogeometric analysis maps all the domain into a normalized **patch**. The discretization is done in the patch, the **elements then are easy created using a structured mesh**.



Comparison of Basis Functions

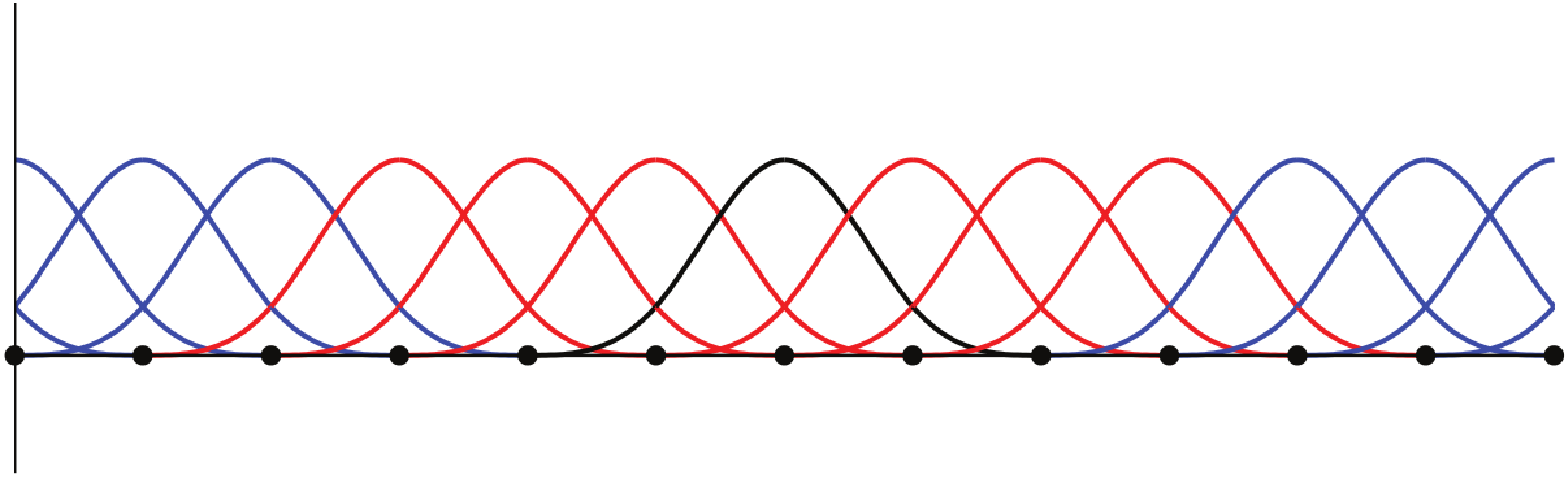
Traditional base functions used in the finite element method are **not smooth between elements**, only C^0 continuity is guaranteed.



This figure shows the standard cubic finite element basis functions with equally spaced nodes.

The idea of the Isogeometric Analysis is to use NURBS to define the elements [Hugh05], NURBS are built from **B-splines**.

The **NURBS base functions** fill all the **patch** (domain), they are smooth between elements. C^1 **continuity is guaranteed**, if a higher order of B-splines are used, then C^p, $p>1$ continuity can be achieved inside the domain.

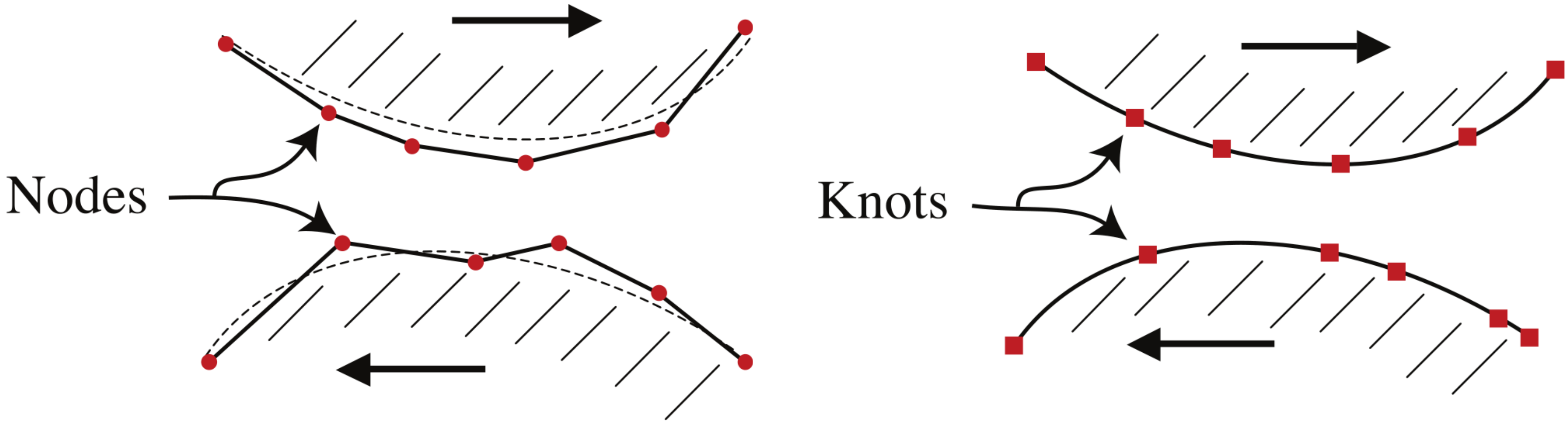


This figure shows an example of cubic B-spline basis functions with equally spaced **knots**.

Some advantages of IGA

Accurate modelation of contact surfaces

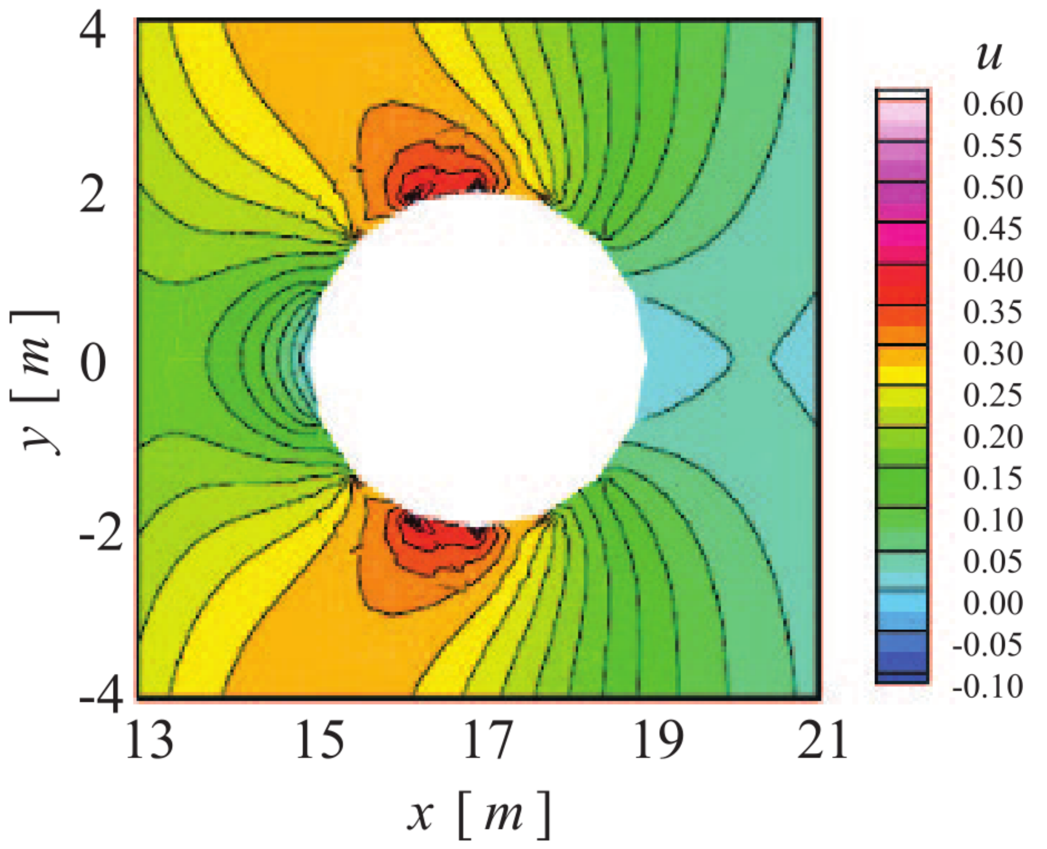
By using an exact description of contact surfaces isogeometric analysis can be used to **simulate friction** between surfaces.

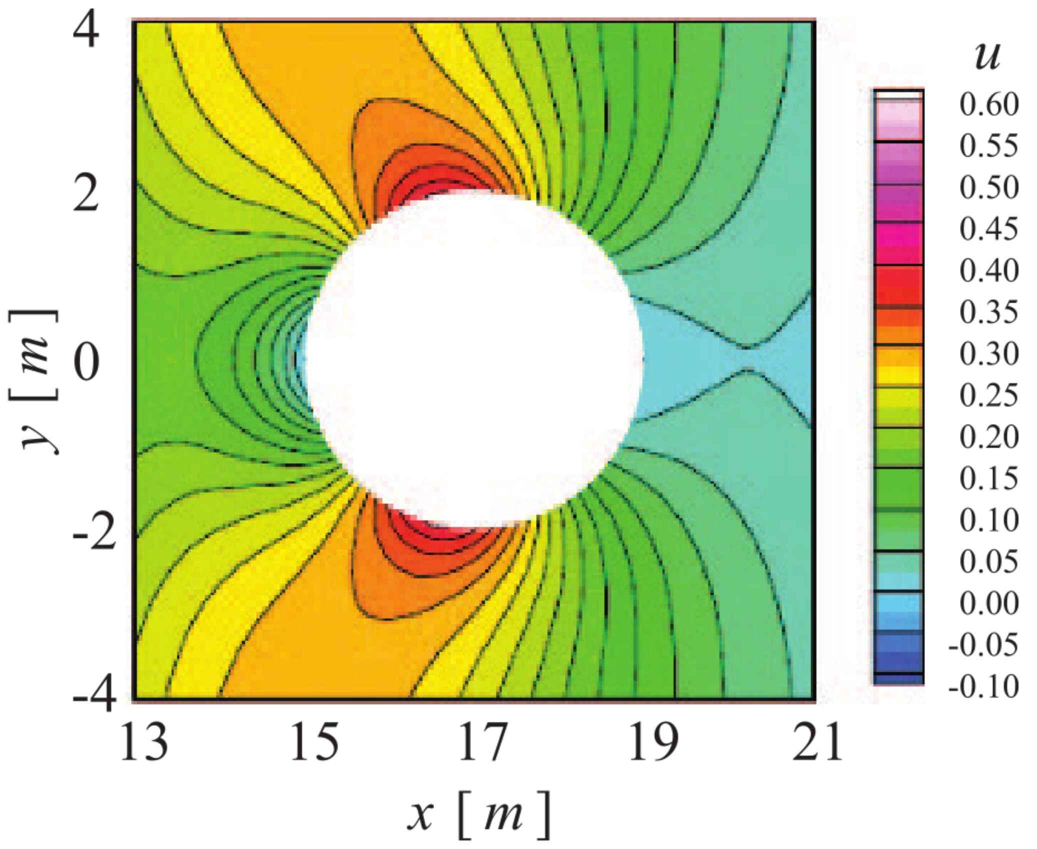


This is problematic to be done with finite element discretizations.

More accurate solutions

Many problems have **sensitivity to the discretization**, for example, the figure shows a solution of the two-dimensional Boussinesq equations. The x-component of velocity obtained using 552 triangles with fifth order polynomials on each triangle [Cott09].



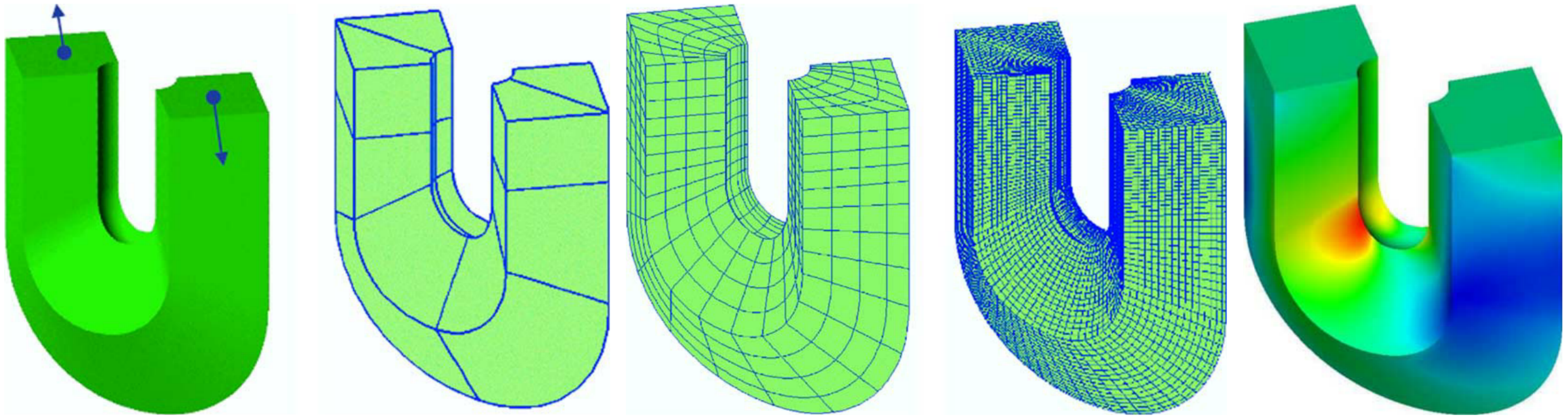


On the left, the elements are straight-sided. The **spurious oscillations** in the solution on the left are due to the use of straight-sided elements for the geometric approximation. On the right, the cylinder is approximated by elements with curved edges, and the oscillations are eliminated.

Independence from mesh generators

Because in the isogeometric analysis the original geometry description is always known, the simulation program can automatically re-mesh the domain as needed to reach the accuracy needed. This makes **multigrid** algorithms easy to apply to this formulation.

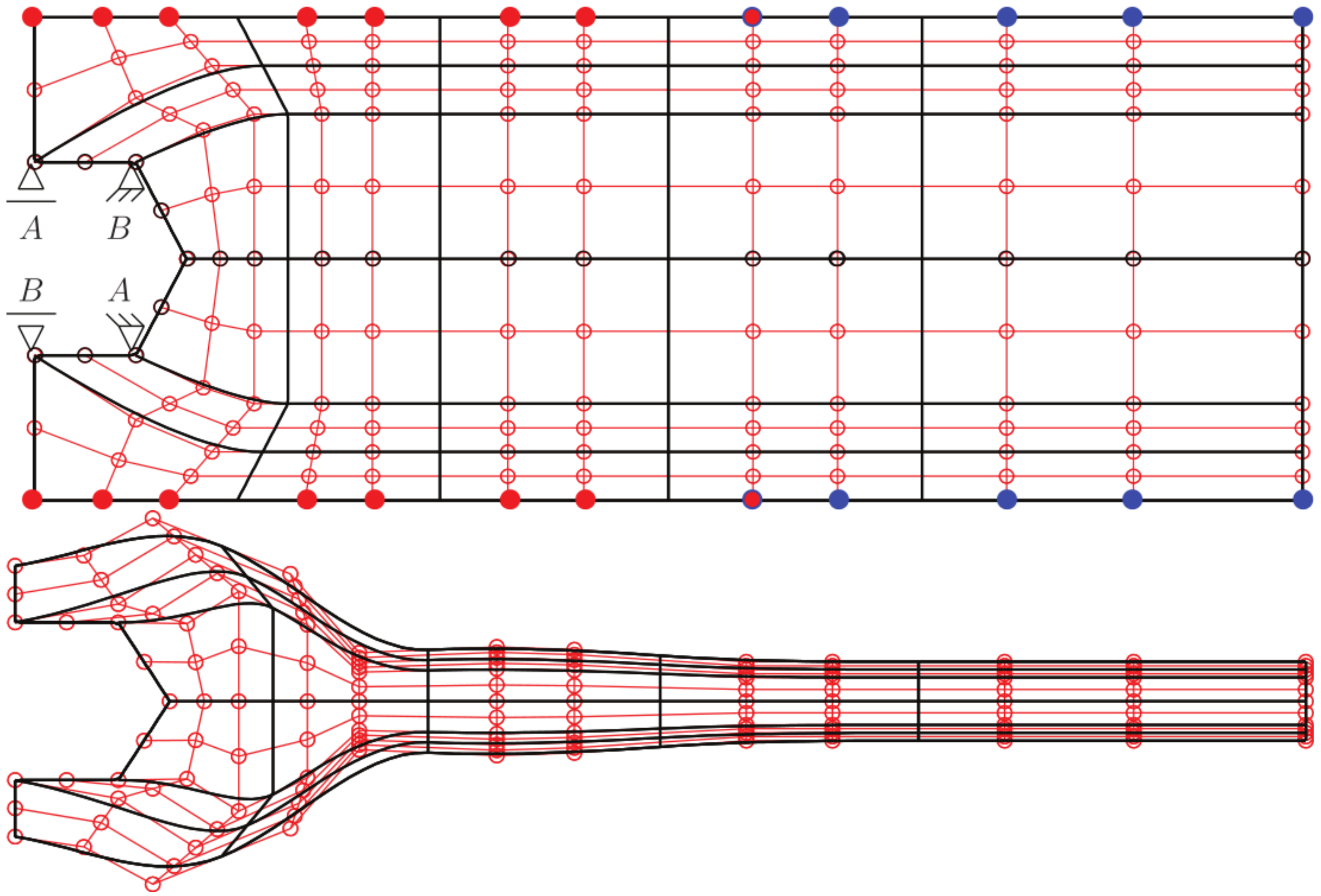
> "Automatic **adaptive** mesh refinement has not been as widely adopted in industry as one might assume from the extensive academic literature, because mesh refinement requires access to the exact geometry and thus seamless and automatic communication with CAD, which simply does not exist." [Cott09].



Having a easy way to generate mesh is also important for large and complex domains, where in the finite element formulation is common that mesh generation is more **time consuming** than the solution of the problem itself.

Shape optimization

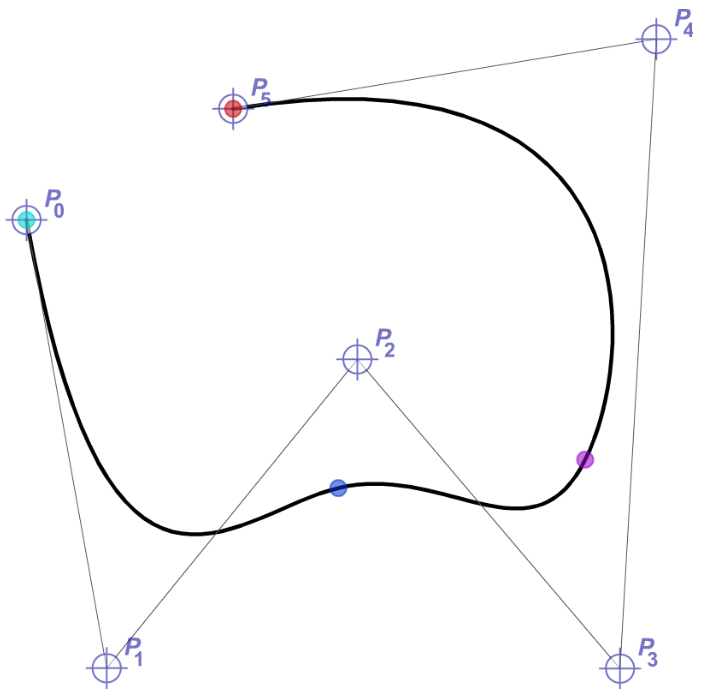
By having the NURBS control points of the geometry all the time, a simulation program can **up-date the geometry on the fly**.



The number of degrees of freedom of the control points is by far less that the degrees of freedom of the mesh.

B-splines for curves

A B-Spline is a particular case of NURBS, so, the following also apply to NURBS.



A B-spline curve is defined as [Pieg97],

$$\mathbf{C}(u)=\sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i.$$

- $\mathbf{P}_i$ are control points (or coordinates), for 2D B-Splines $\mathbf{P}_i\in\mathbb{R}^2$, for 3D B-Splines $\mathbf{P}_i\in\mathbb{R}^3$.
- $N_{i,p}(u)$ are piecewise polynomial functions forming a basis for the vector space of all piecewise polynomial functions of degree p, for a fixed knot vector $\mathbf{U}=\{u_i\}$, $0\le i\le m$.

Basis functions

Let $\mathbf{U}=\{u_{0,}u_{1,}\ldots,u_m\}$ be a nondecreasing sequence of real numbers, such that $u_i \leq u_{i+1}$ for $i=0,1,\ldots,m-1$. The u_i are called knots, and **U** is the **knot vector**.

The i-th B-spline basis function of ***p*-degree**, denoted by $N_{i,p}(u)$ are defined recursively with the **Cox-de Boor** recursion formula.

For $p=0$ $$N_{i,0}(u)=\begin{cases}1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise}\end{cases}.$$

And for $p=1,2,3,\ldots$

$$N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u).$$

In a knot vector, **multiplicity** of knots can exist!! (this is important, also complicates things)

$$u_i=u_{i+1}$$

So, **division by zero** can occur in the previous formulation, for those cases we will use the following definition

$$\frac{x}{0}\overset{\text{def}}{=}0.$$

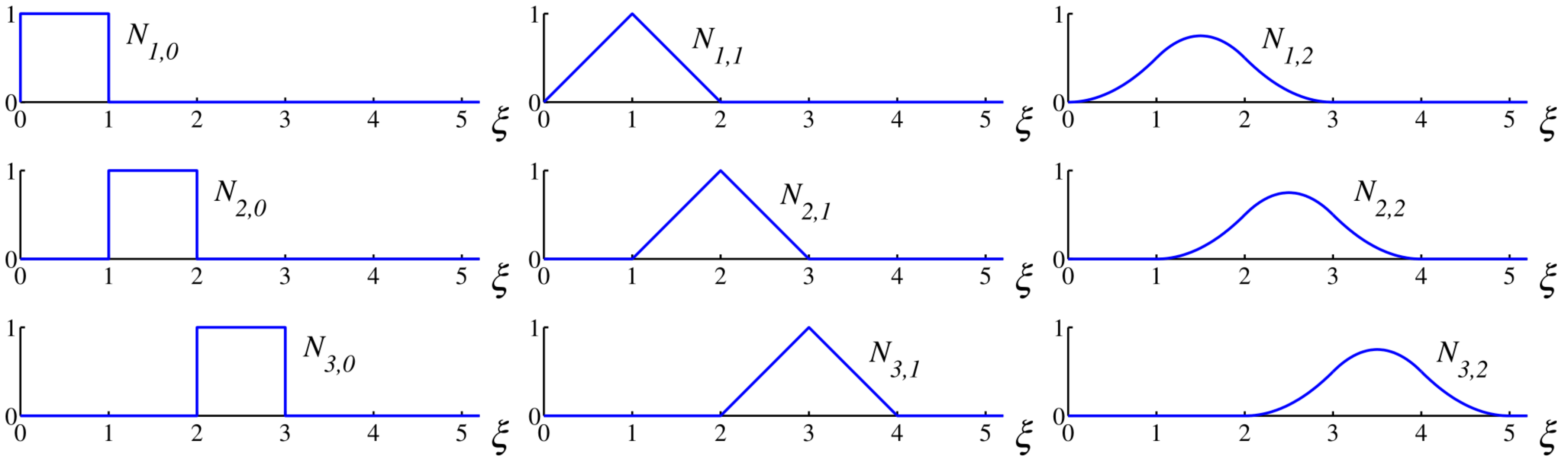
Shape of the basis functions

Let $\mathbf{U}=\{u_0, u_1, \dots, u_m\}$ be a nondecreasing sequence of real numbers, such that $u_i \le u_{i+1}$ for $i=0,1,\dots,m-1$. The u_i are called knots, and U is the **knot vector**.

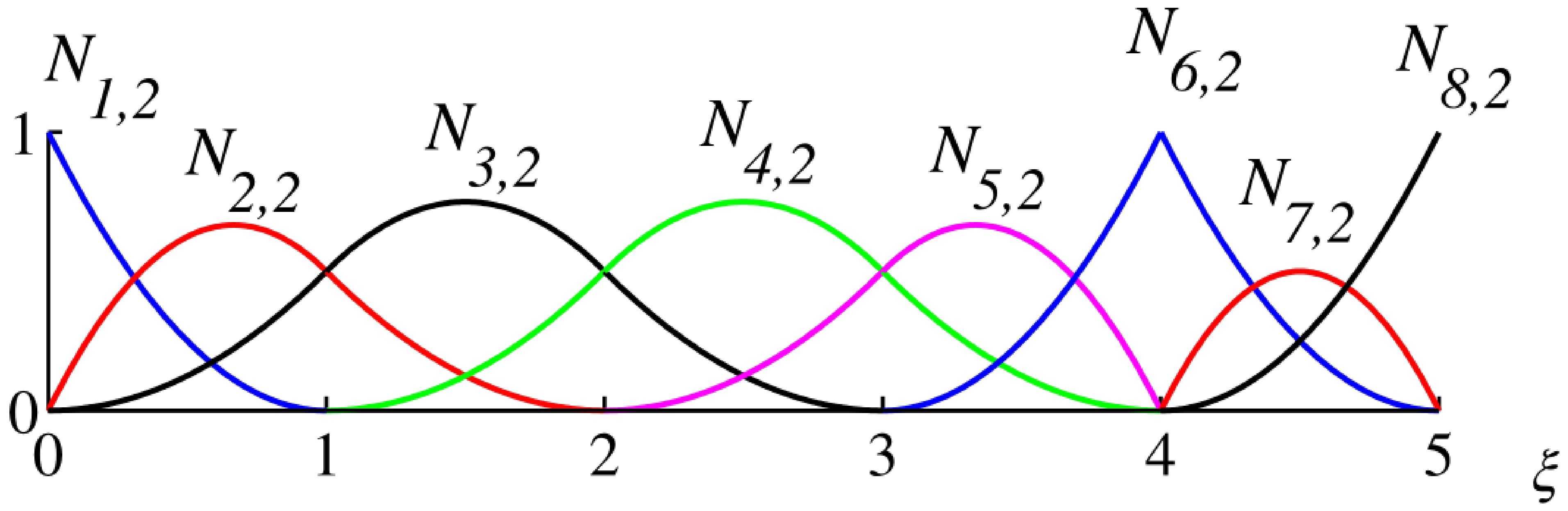
The i-th B-spline basis function of ***p*-degree**, denoted by $N_{i,p}(u)$ are defined recursively with the Cox-de Boor recursion formula.

For $p=0$
$$N_{i,0}(u)=\begin{cases}1 & \text{if } u_i \le u \le u_{i+1} \\ 0 & \text{otherwise}\end{cases}.$$

And for $p=1,2,3,\dots$
$$N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u).$$



For example, a basis for a 2-degree



With a knot vector, $\mathbf{U}=\{u_{0,} u_{1,} \ldots, u_m\}$

$$\mathbf{U}=\{0,0,0,1,2,3,4,4,5,5,5\}\text{, with } m=10.$$

Considerations

- For any u, a maximum of $p+1$ basis functions are **different of zero**.
- $\sum_{i=0}^{p} N_{i,p}(u)=1$, for $u_0 \le u \le u_m$.
- $N_{i,p}(u) \ge 0$, for $u_0 \le u \le u_m$.

Dependence for calculation

The i-th B-spline basis function of ***p*-degree**, denoted by $N_{i,p}(u)$ are **defined recursively** with the Cox-de Boor recursion formula.

For $p=0$
$$N_{i,0}(u)=\begin{cases}1 & \text{if } u_i \le u \le u_{i+1} \\ 0 & \text{otherwise}\end{cases}.$$

And for $p=1,2,3,\ldots$
$$N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u).$$

There is a **dependence** to calculate each $N_{i,p}(u)$,

$$\begin{array}{ccccccc}
N_{0,0} & & & & & & \\
 & N_{0,1} & & & & & \\
N_{1,0} & & N_{0,2} & & & & \\
 & N_{1,1} & & N_{0,3} & & & \\
N_{2,0} & & N_{1,2} & & & & \\
 & N_{2,1} & & N_{1,3} & & & \\
N_{3,0} & & N_{2,2} & \vdots & & & \\
 & N_{3,1} & \vdots & & & & \\
N_{4,0} & \vdots & & & & & \\
\vdots & & & & & &
\end{array}$$

Computational algorithm

Many evaluations of the basis functions are **equal to zero**. Also recursive calculation can be expensive.

An **efficient algorithm** to calculate all the non-vanishing base functions [Pieg97] is:

```
$\mathbf{BasisFunctions}(u, p, \mathbf{U}, \mathbf{N})$
· $i \leftarrow \mathbf{FindSpan}(u, p, \mathbf{U})$
· $N_0 \leftarrow 1$
· for $j \leftarrow 1, 2, \ldots, p$
· · $L_j \leftarrow u - u_{i+1-j}$
· · $R_j \leftarrow u_{i+1} - u$
· · $saved \leftarrow 0$
· · for $r \leftarrow 0, 2, \ldots, j-1$
· · · $temp \leftarrow \frac{N_r}{R_{r+1} + L_{j-r}}$
· · · $N_r \leftarrow saved + R_{r+1} * temp$
· · · $saved \leftarrow L_{j-r} * temp$
· · $N_j \leftarrow saved$
```

The result will be $\mathbf{N} = N_{0,} N_{1,} \ldots N_p$. This algorithm also guarantees that there will be **no division by zero**.

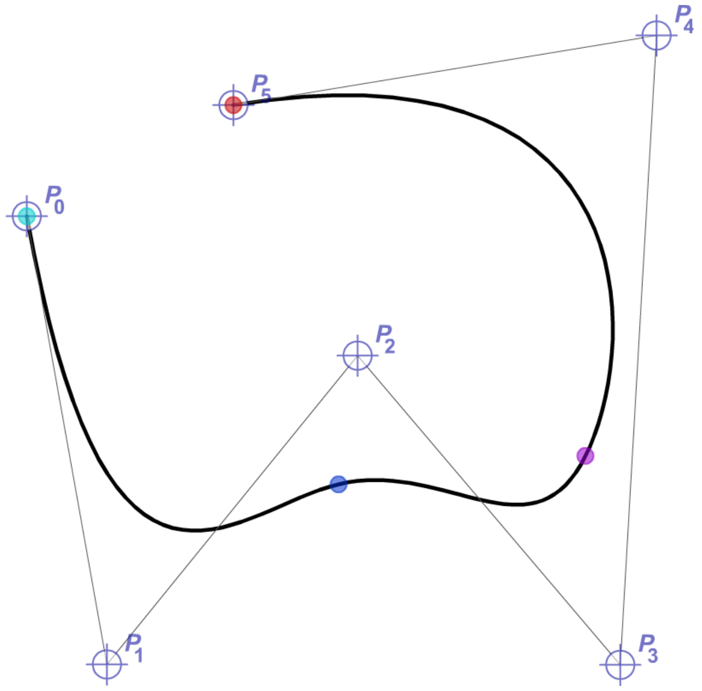
FindSpan algorithm

This algorithm finds the interval or span of $\mathbf{U}=\{u_{0,}u_{1,}\ldots,u_m\}$ where u is located, by making a **binary search**

```
FindSpan(u, p, U)
· $i \leftarrow p$
· $high \leftarrow m-p$
· repeat
· · $middle \leftarrow \frac{(i+high)}{2}$
· · if $u<u_{middle}$
· · · $high \leftarrow middle$
· · else
· · · if $i=middle$
· · · · exit_repeat
· · · $i \leftarrow middle$
```

The result will be the i that satisfy $u \in [u_i, u_{i+1})$.

B-spline curves

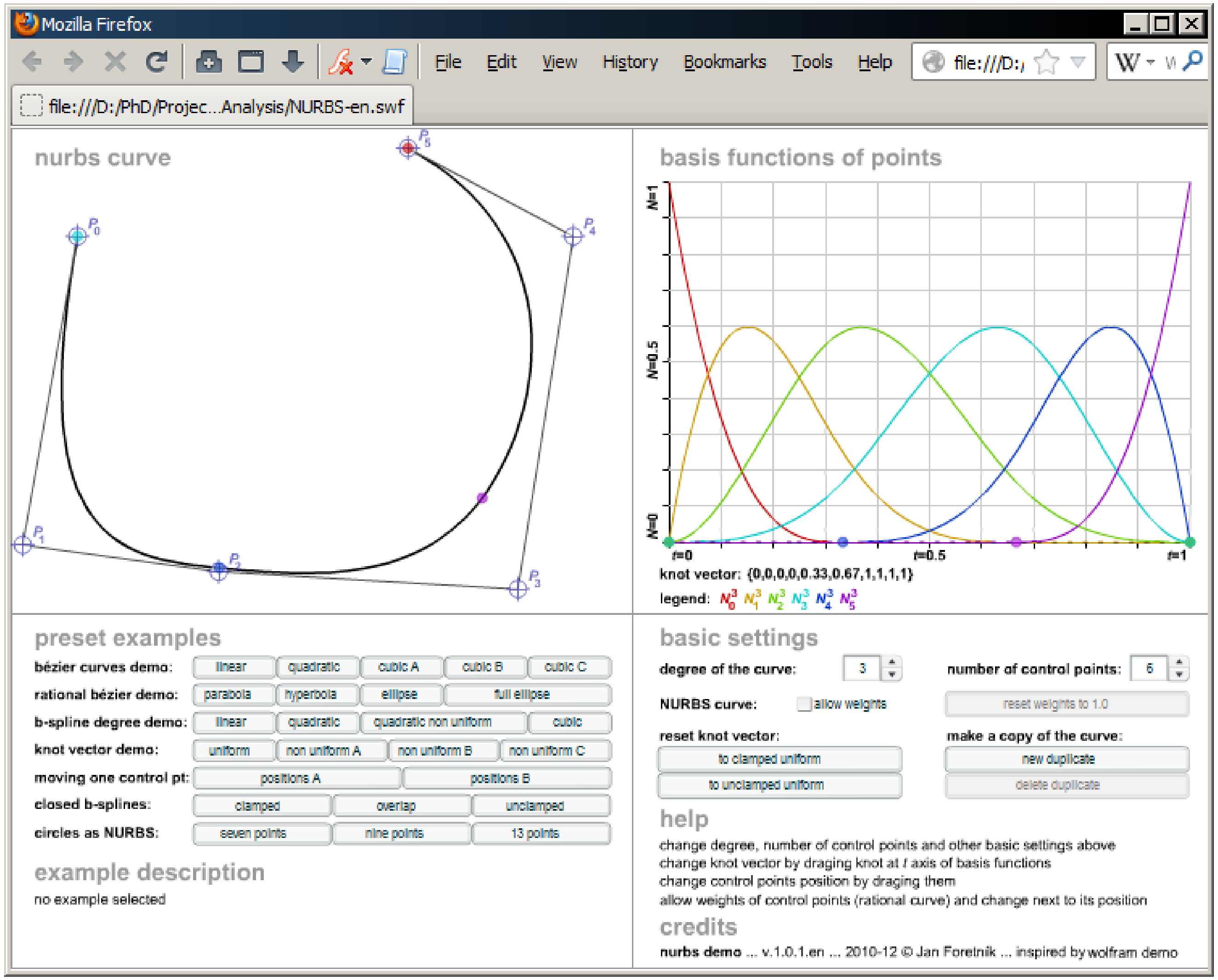


A B-spline curve is defined as [Pieg97],

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i.$$

- $\mathbf{P}_i$ is a **vector of control points** (or coordinates), for 2D B-Splines $\mathbf{P}_i \in \mathbb{R}^2$, for 3D B-Splines $\mathbf{P}_i \in \mathbb{R}^3$.
- $N_{i,p}(u)$ are piecewise polynomial functions forming a **basis** for the vector space of all piecewise polynomial functions of **degree *p***, for a fixed knot vector $\mathbf{U} = \{u_i\}$, $0 \le i \le m$.

Interactive demo


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///D:/PhD/Projec...Analysis/NURBS-en.swf
nurbs curve
basis functions of points
N=1
N=0.5
N=0
t=0
t=0.5
t=1
preset examples
bézier curves demo: linear quadratic cubic A cubic B cubic C
rational bézier demo: parabola hyperbola ellipse full ellipse
b-spline degree demo: linear quadratic quadratic non uniform cubic
knot vector demo: uniform non uniform A non uniform B non uniform C
moving one control pt: positions A positions B
closed b-splines: clamped overlap unclamped
circles as NURBS: seven points nine points 13 points
example description
basic settings
reset weights to 1.0
to clamped uniform
to unclamped uniform
new duplicate
delete duplicate
help
credits

knot vector: {0,0,0,0,0.33,0.67,1,1,1,1}

legend: N_0^3 N_1^3 N_2^3 N_3^3 N_4^3 N_5^3

no example selected

degree of the curve: 3

number of control points: 6

NURBS curve: ☐ allow weights

reset knot vector:

make a copy of the curve:

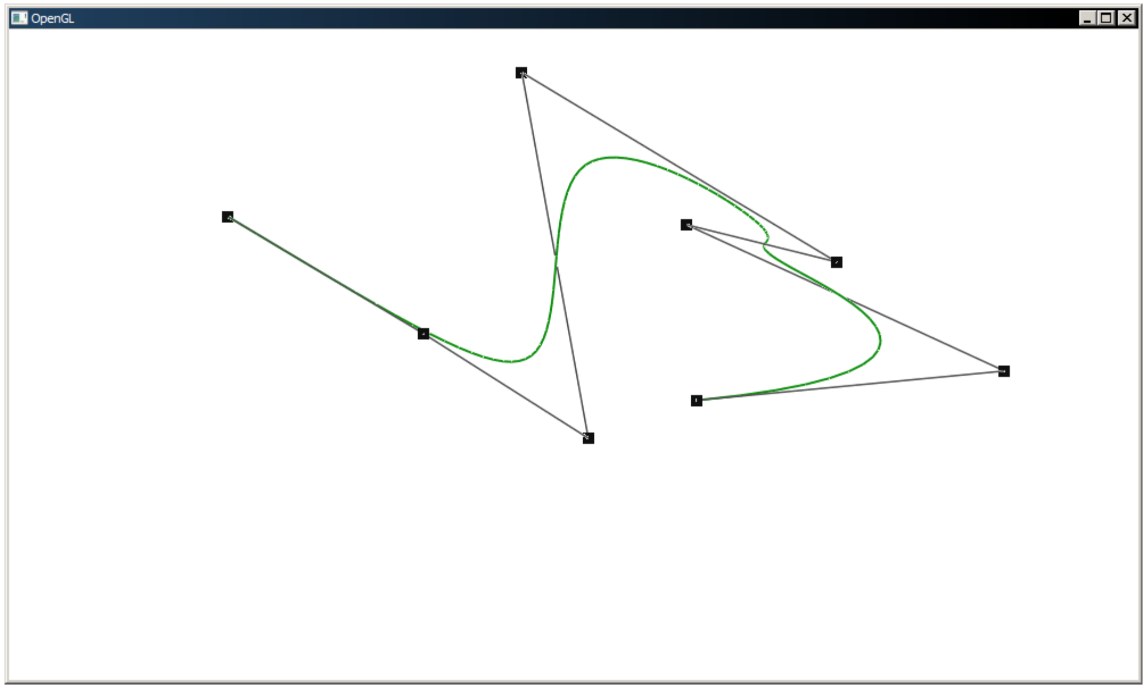
change degree, number of control points and other basic settings above
change knot vector by draging knot at *t* axis of basis functions
change control points position by draging them
allow weights of control points (rational curve) and change next to its position

nurbs demo ... v.1.0.1.en ... 2010-12 © Jan Foretník ... inspired by wolfram demo

This interactive demostration of B-Splines and NURBS, was made by Jan Foretník, it can be found at:

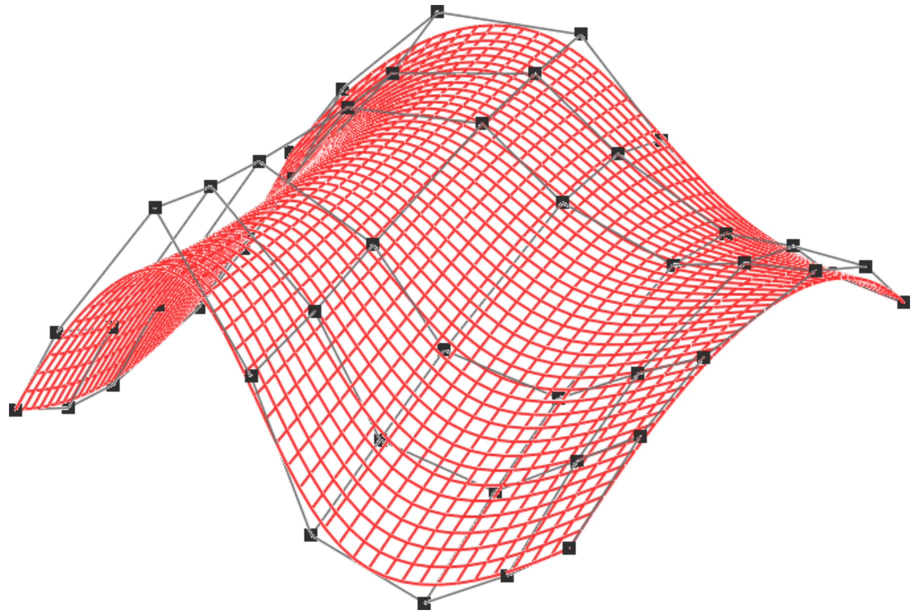
http://geometrie.foretnik.net/?id=index&lang=en

B-Spline curves in 3D



i	$(\mathbf{P}_i)_x$	$(\mathbf{P}_i)_y$	$(\mathbf{P}_i)_z$
0	-4.0	-4.0	-3.0
1	-3.0	0.0	-2.0
2	0.0	1.0	1.0
3	3.0	-4.0	-3.0
4	5.0	2.0	-1.0
5	6.0	-3.0	1.0
6	9.0	2.0	3.0
7	-2.0	6.0	-3.0

B-spline surfaces



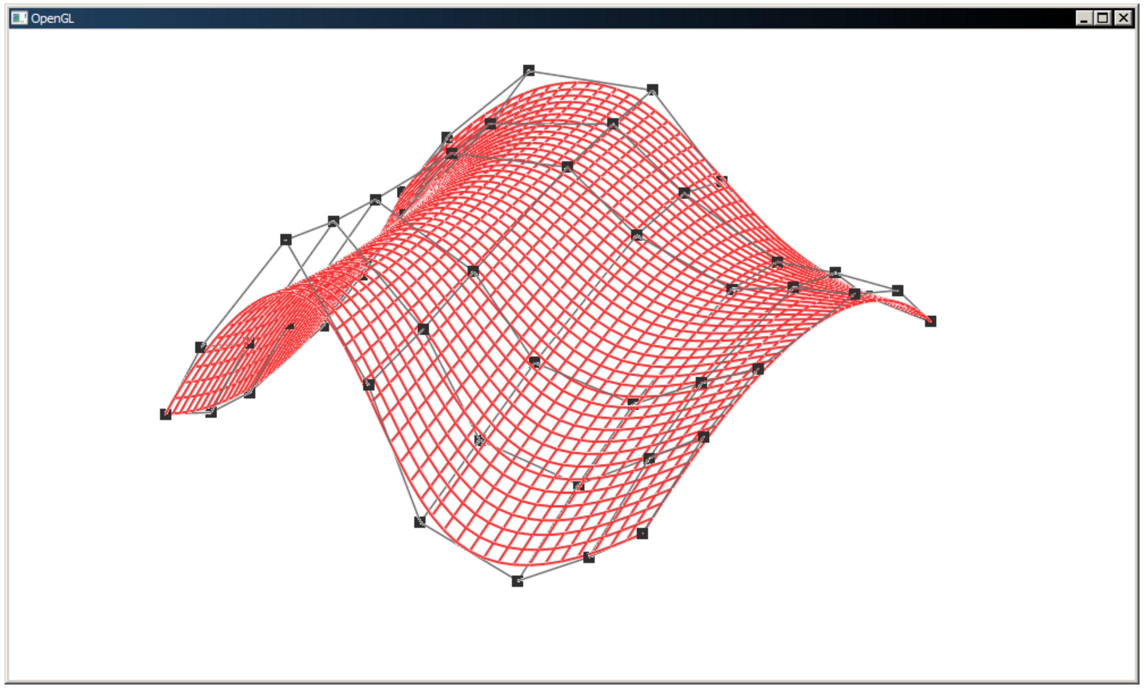
A B-spline surface is defined as [Pieg97],

$$\mathbf{S}(u,v)=\sum_{i=0}^{n_1}\sum_{j=0}^{n_2} N_{i,p}(u)\,N_{j,q}(v)\,\mathbf{P}_{ij}.$$

- $\mathbf{P}_{ij}$ is a **matrix of control points** (or coordinates), for 2D B-Splines $\mathbf{P}_{ij}\in\mathbb{R}^2$, for 3D B-Splines $\mathbf{P}_{ij}\in\mathbb{R}^3$.
- The basis $N_{i,p}(u)$ and $N_{j,q}(v)$ could have different degrees and knot vectors $\mathbf{U}=\{u_i\}$, $0\le i\le m_1$ and $\mathbf{V}=\{v_i\}$, $0\le i\le m_2$.

B-Spline surfaces in 3D

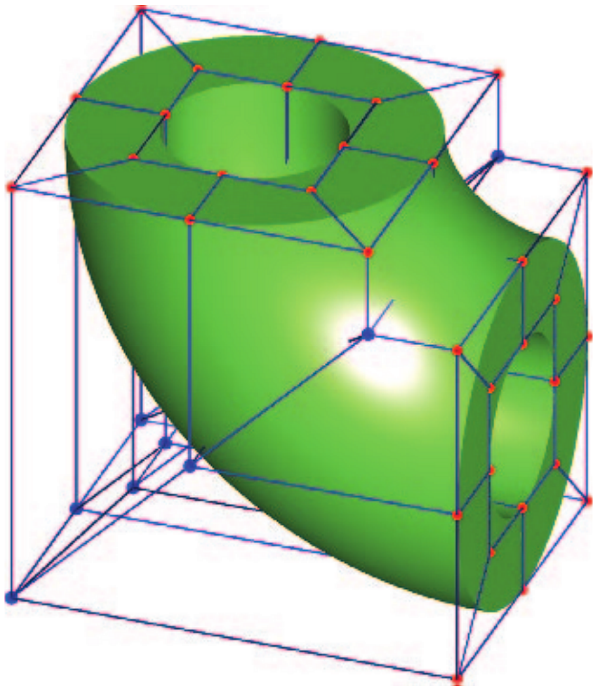
i	j	$(\mathbf{P}_{ij})_x$	$(\mathbf{P}_{ij})_y$	$(\mathbf{P}_{ij})_z$
0	0	-5.00	-5.00	0.00
0	1	-4.20	-5.17	1.50
0	2	-2.35	-5.51	4.01
0	3	-0.67	-5.58	1.59
0	4	0.35	-5.55	-0.84
0	5	2.36	-5.70	-1.52
0	6	3.87	-5.87	-0.71
0	7	5.00	-6.00	0.00
1	0	-5.11	-3.06	-0.70
1	1	-4.28	-3.24	0.85
1	2	-2.37	-3.62	3.67
1	3	-0.52	-3.76	2.02
1	4	0.62	-3.81	0.14
1	5	2.68	-3.99	-0.25
1	6	4.17	-4.17	0.61
1	7	5.32	-4.29	1.28
2	0	-5.24	-1.33	-0.98
2	1	-4.37	-1.51	0.57
2	2	-2.47	-1.86	3.42
2	3	-0.41	-2.05	2.51
2	4	0.84	-2.12	1.04
2	5	2.89	-2.30	0.70
2	6	4.32	-2.44	1.44
2	7	5.49	-2.53	1.97
3	0	-5.54	2.07	-1.02
3	1	-4.62	1.91	0.44
3	2	-2.77	1.64	2.96
3	3	-0.34	1.39	3.25
3	4	1.09	1.28	2.27
3	5	3.05	1.15	1.66
3	6	4.30	1.10	1.95
3	7	5.51	1.10	2.06
4	0	-5.73	3.90	-0.77
4	1	-4.79	3.75	0.62
4	2	-3.00	3.53	2.79
4	3	-0.42	3.28	3.36
4	4	1.06	3.17	2.37
4	5	2.95	3.07	1.46
4	6	4.11	3.07	1.47
4	7	5.34	3.11	1.36
5	0	-6.00	6.00	0.00
5	1	-5.05	5.84	1.29
5	2	-3.30	5.59	2.98
5	3	-0.67	5.26	3.23
5	4	0.81	5.08	1.85
5	5	2.66	4.95	0.50
5	6	3.76	4.95	0.25
5	7	5.00	5.00	0.00



i	$\mathbf{U}_i$
0	0.0000
1	0.0000
2	0.0000
3	0.0000
4	0.4602
5	0.4992
6	1.0000
7	1.0000
8	1.0000
9	1.0000

j	$\mathbf{V}_j$
0	0.0000
1	0.0000
2	0.0000
3	0.0000
4	0.3184
5	0.4641
6	0.6912
7	0.7501
8	1.0000
9	1.0000
10	1.0000
11	1.0000

B-Spline volumes



A B-spline volume is defined as,

$$\mathbf{B}(u,v,w)=\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\sum_{k=0}^{n_3} N_{i,p}(u)\,N_{j,q}(v)\,N_{k,r}(w)\,\mathbf{P}_{ijk}.$$

- $\mathbf{P}_{ijk}$ is a **tensor of rank 3 of control points** (or coordinates), $\mathbf{P}_{ijk}\in\mathbb{R}^3$.
- The basis $N_{i,p}(u)$, $N_{j,q}(v)$ and $N_{k,r}(w)$ could have different degrees and knot vectors $\mathbf{U}=\{u_i\}$, $0\le i\le m_1$, $\mathbf{V}=\{v_i\}$, $0\le i\le m_2$ and $\mathbf{W}=\{w_i\}$, $0\le i\le m_3$.

Now, the NURBS

NURBS curve

A NURBS (**Non-Uniform Rational B-Spline**) curve is defined as,

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) W_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) W_i}.$$

There is a **weight** for each control point, W_i for $i=0,1,2,\ldots,n$.

Note that if $W_i=1$ for $i=0,1,2,\ldots,n$ then $\mathbf{C}(u)$ corresponds to the definition of a B-Spline.

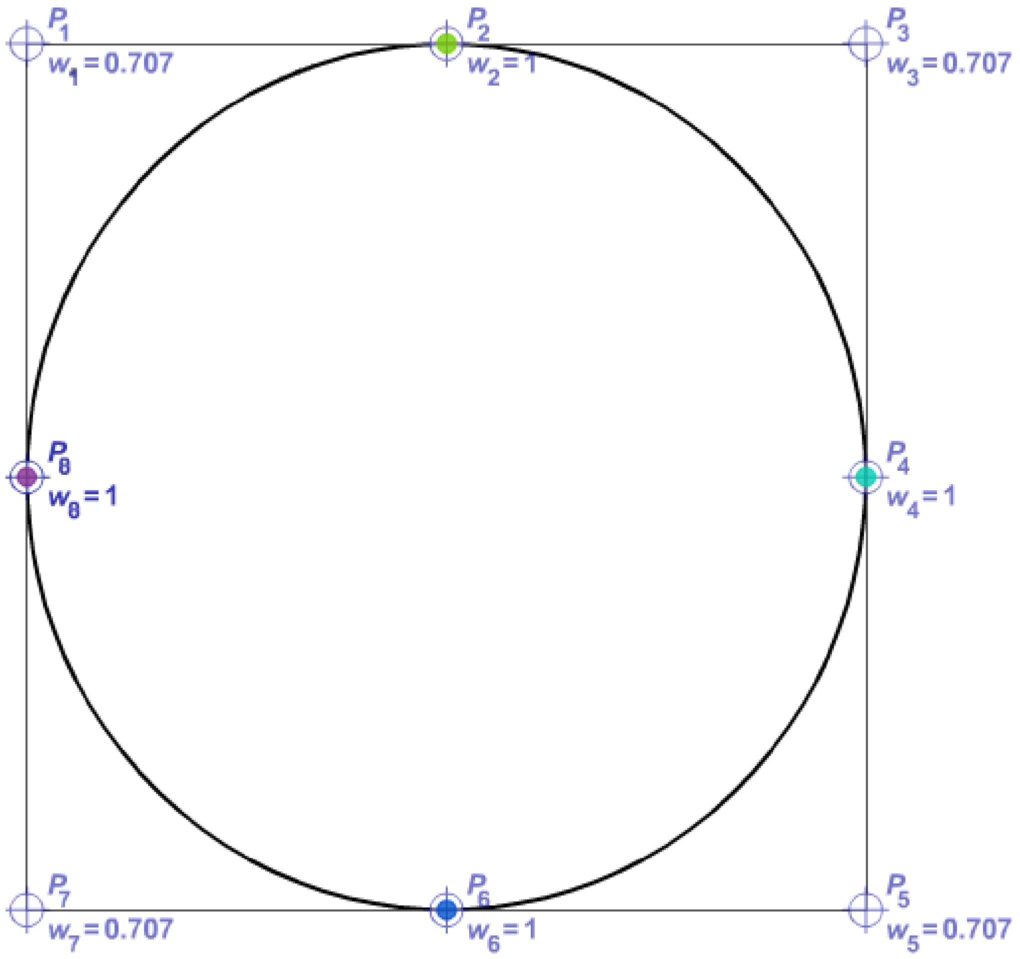
Non-uniform: Means that the entries of the knot vector, where the $N_{i,p}(u)$ is defined, are not periodic (equally spaced).

Rational: The use of weights allows more control of the NURBS.

Advantages

Conic functions (circle, ellipses, etc.) can not be described accurately with B-Splines, these can only approximate.

NURBS can represent exactly these conic functions.



NURBS surface

A NURBS surface is defined as,

$$\mathbf{S}(u,v)=\frac{\sum_{i=0}^{n_1}\sum_{j=0}^{n_2} N_{i,p}(u)\,N_{j,q}(v)\,W_{ij}\,\mathbf{P}_{ij}}{\sum_{i=0}^{n_1}\sum_{j=0}^{n_2} N_{i,p}(u)\,N_{j,q}(v)\,W_{ij}}.$$

Weights are W_{ij} for $i=0,1,2,\ldots,n_1$ and $j=0,1,2,\ldots,n_2$.

NURBS volume

A NURBS volume is defined as,

$$\mathbf{B}(u,v,w)=\frac{\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\sum_{k=0}^{n_3} N_{i,p}(u)\,N_{j,q}(v)\,N_{k,r}(w)\,W_{ijk}\,\mathbf{P}_{ijk}}{\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\sum_{k=0}^{n_3} N_{i,p}(u)\,N_{j,q}(v)\,N_{k,r}(w)\,W_{ijk}}.$$

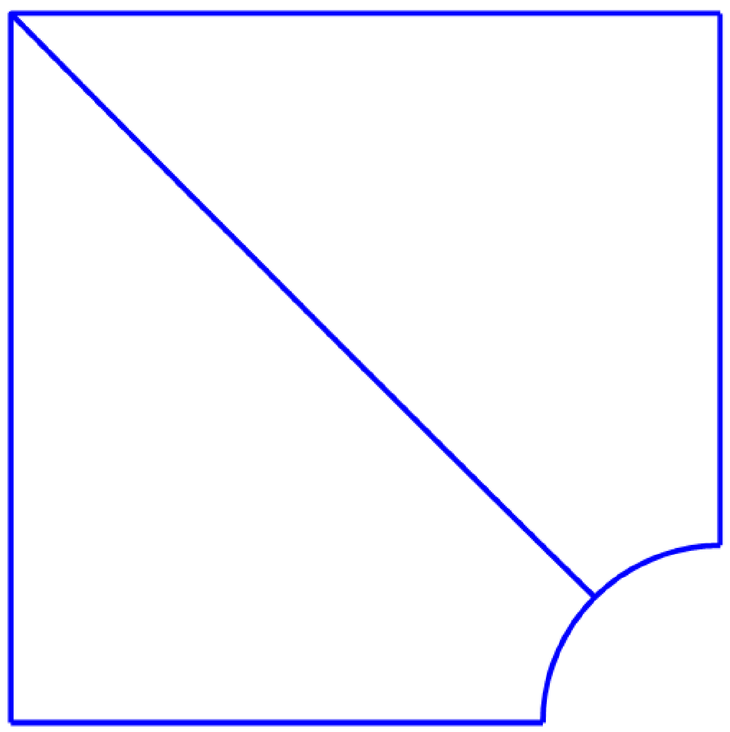
Weights are W_{ijk} for $i=0,1,2,\ldots,n_1$, $j=0,1,2,\ldots,n_2$ and $k=0,1,2,\ldots,n_3$.

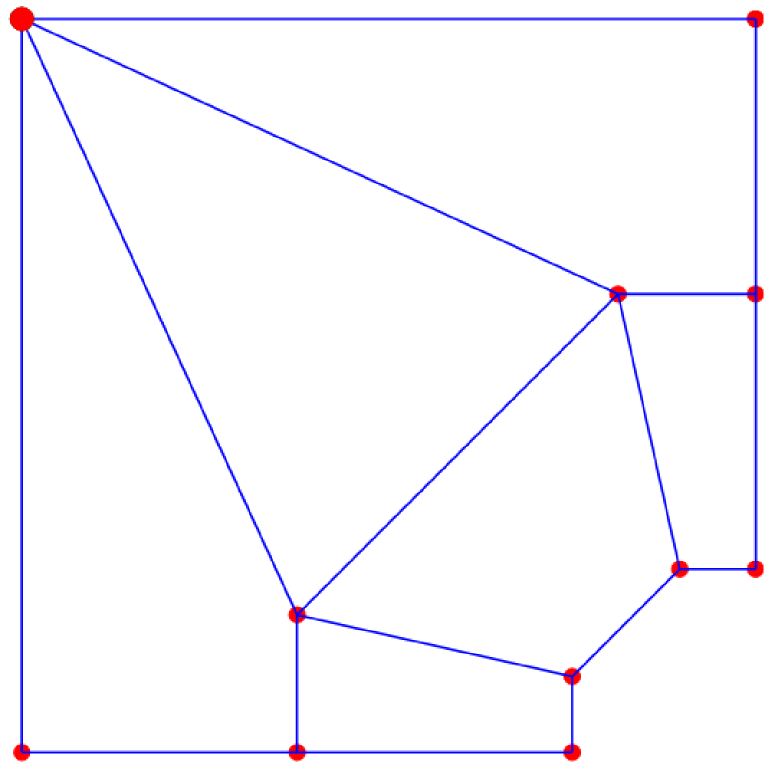
Insertion of knots (a.k.a. meshing)

CAD software will generate NURBS objects (curves, surfaces or volumes) using the **minimun number of control points** possible, i.e. using small knot vectors.

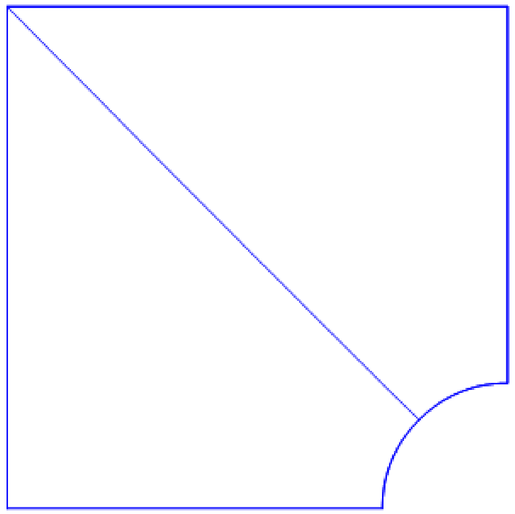
For example, the next figure has knots vectors

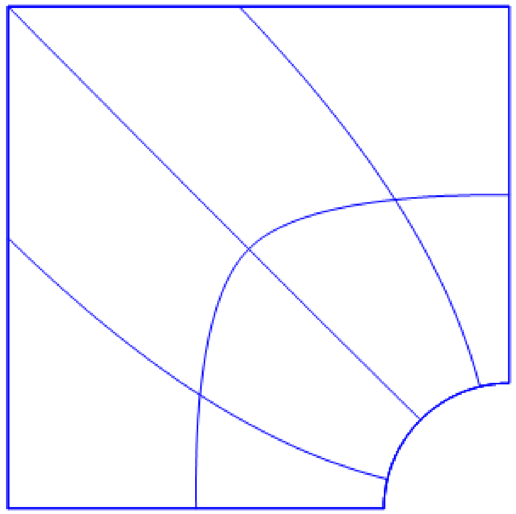
$$\mathbf{U}=\{0,0,0,0.5,1,1,1\} \text{ and } \mathbf{V}=\{0,0,0,1,1,1\}.$$

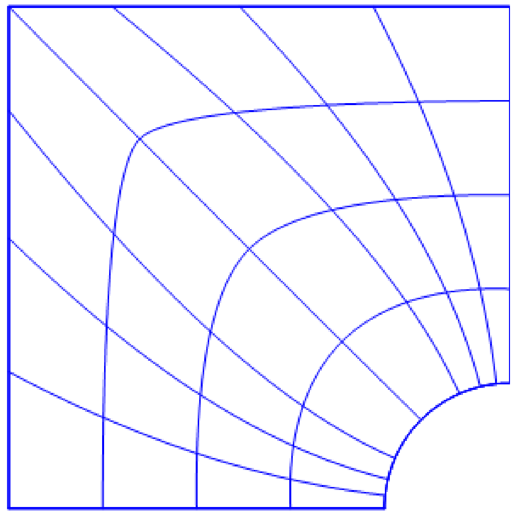


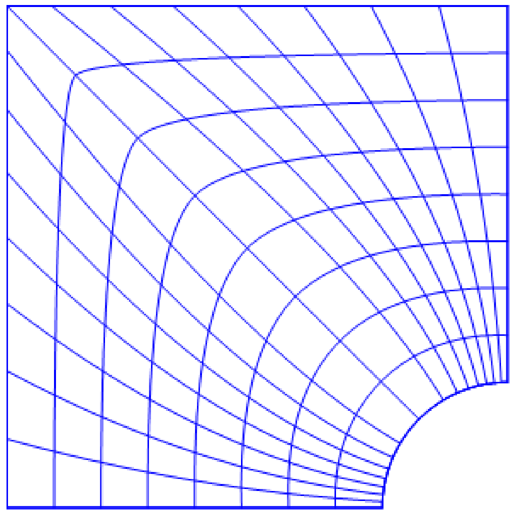


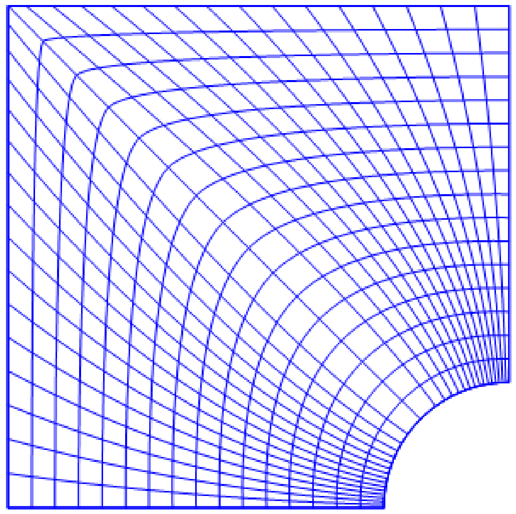
Meshing in isogeometric analysis consists on **inserting knots** to refine the normalized space (parameter space).

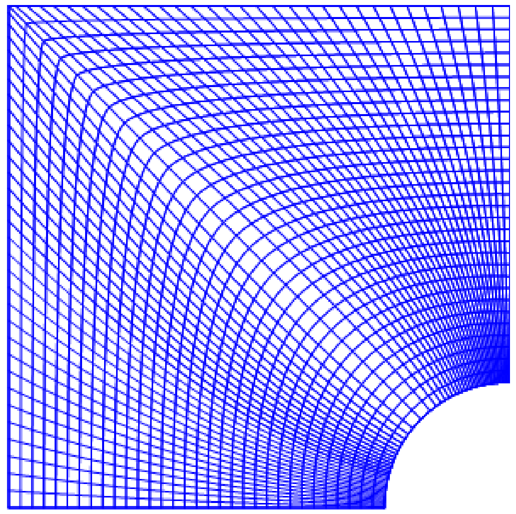

Mesh 1


Mesh 2


Mesh 3


Mesh 4


Mesh 5


Mesh 6

Knot insertion in curves

Let $\mathbf{C}(u)$ be a NURBS curve with control points $\mathbf{P}_i$ and basis functions $N_{i,p}(u)$, with $i=1,2,\ldots,n$

$$\mathbf{C}(u)=\sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i,$$

with a knot vector

$$\mathbf{U}=\{u_{0,}u_{1,}\ldots,u_m\}.$$

We want to insert a knot in the position $\bar{u}\in[u_k,u_{k+1})$, to form a new knot vector

$$\bar{\mathbf{U}}=\{u_{0,}u_{1,}\ldots,u_{k-1},u_k,\bar{u},u_{k+1},u_{k+2},\ldots,u_m\},$$

with $m+1$ knots.

The curve will have a new representation with $n+1$ control points $\mathbf{Q}_i$ and basis functions $\bar{N}_{i,p}(u)$

$$\mathbf{C}(u)=\sum_{i=0}^{n+1} \bar{N}_{i,p}(u)\mathbf{Q}_i.$$

Inserting knots consists on determining the $\mathbf{Q}_i$, $i=1,2,\ldots,n+1$. Althoug, some control points change, the shape of the NURBS is not modified.

Knot insertion algorithm

To **insert a knot** in a NURBS of p-degree it is necesary to **modify p control points**.

For a NURBS with n control points $\mathbf{P}_i$, with $i=0,1,\ldots,n$, after inserting a knot in the position $\bar{u}\in\left[u_k, u_{k+1}\right)$, **the new control points** will be $\mathbf{Q}_i$ with $i=0,1,\ldots,n+1$

$$\mathbf{Q}_i=\alpha_i \mathbf{P}_i+\left(1-\alpha_i\right)\mathbf{P}_{i-1},$$

with

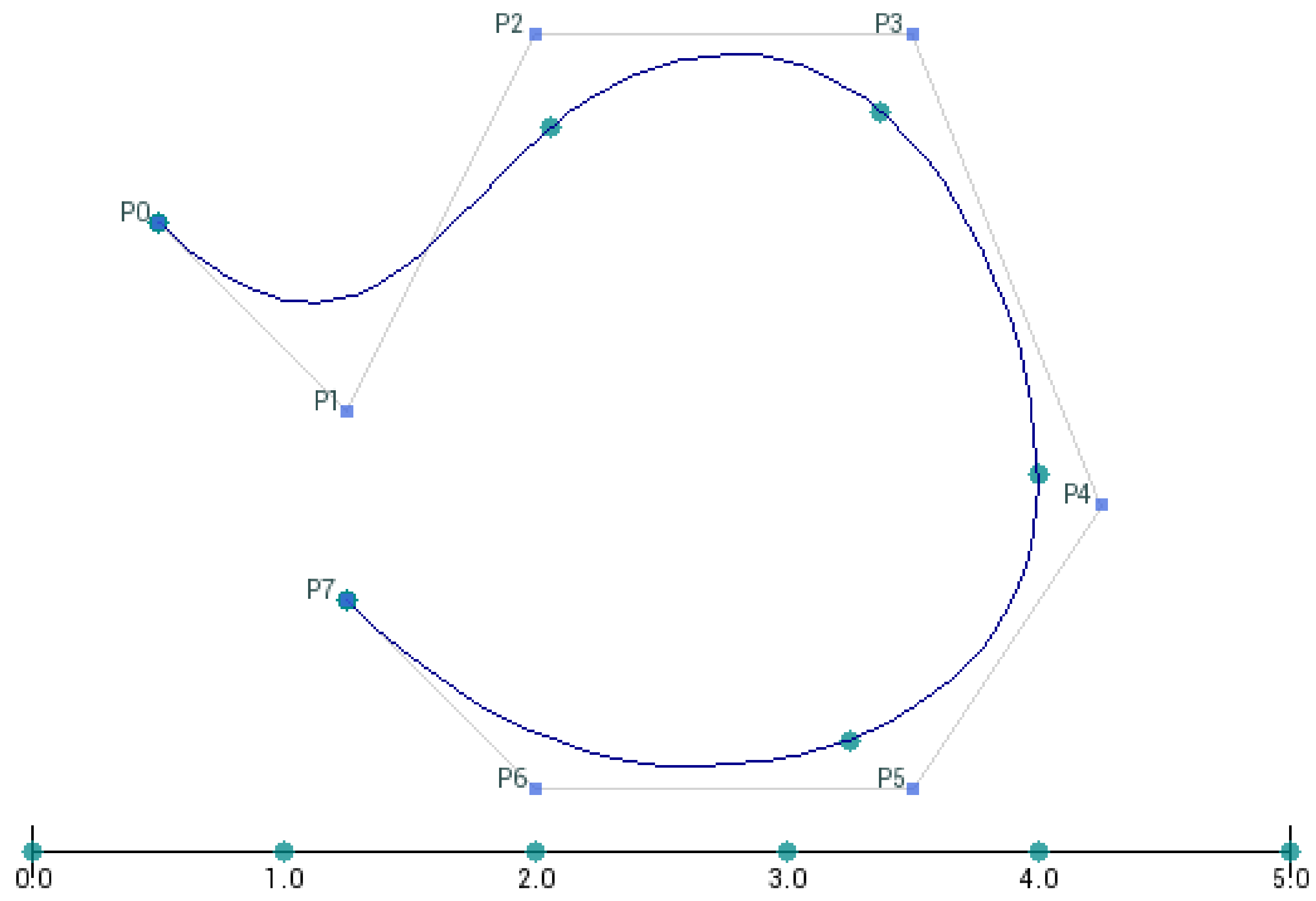
$$\alpha_i=\begin{cases} 1 & i\leq k-p \\ \dfrac{\bar{u}-u_i}{u_{i+p}-u_i} & k-p+1\leq i\leq k. \\ 0 & i\geq k+1 \end{cases}$$

The value of k is determinated using the function $k \leftarrow \mathbf{FindSpan}\left(u, p, \mathbf{U}\right)$.

An **efficient algorithm** to insert knots is given in [Pieg97].

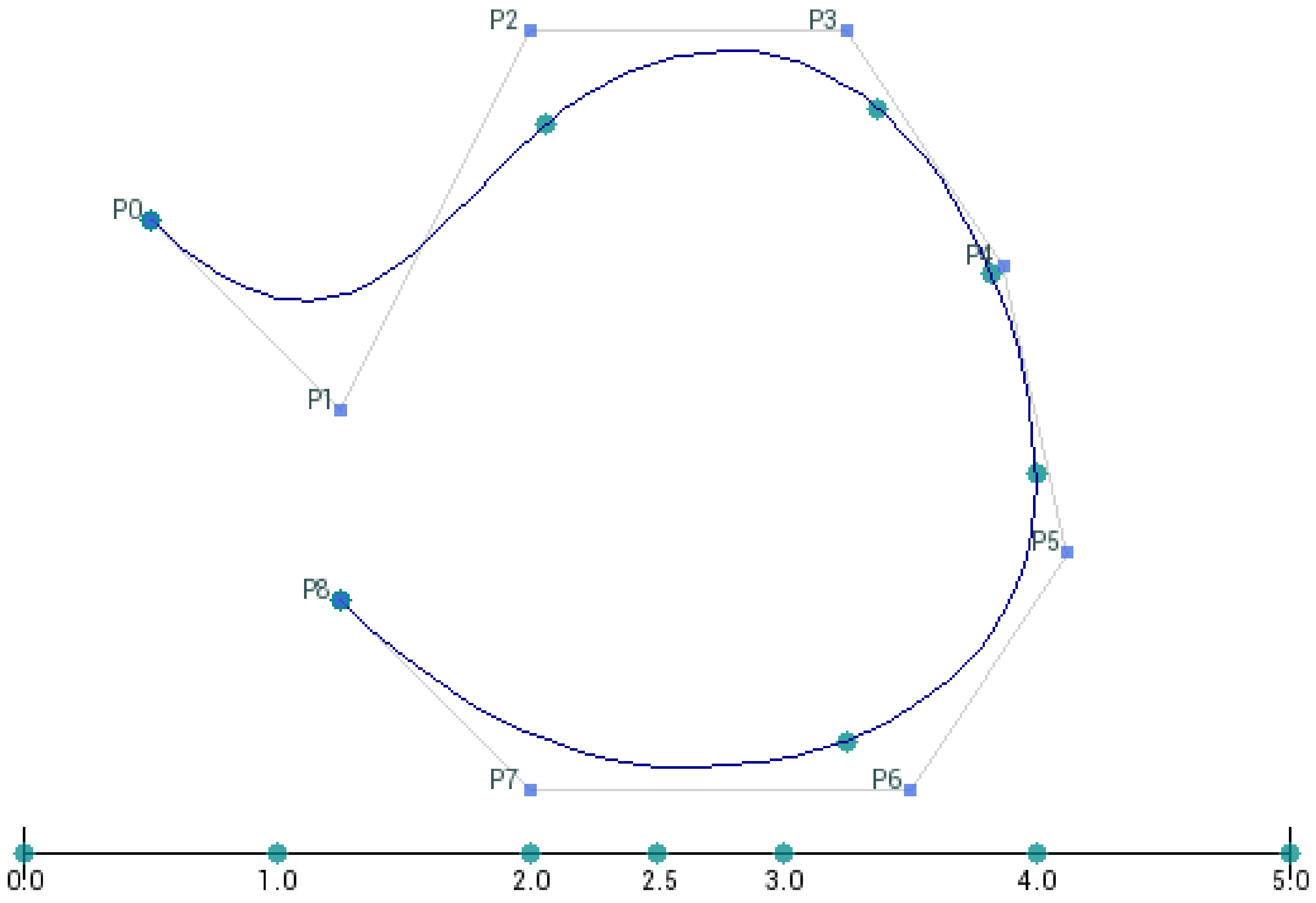
Next, an example:

For a curve with a knot vector $\mathbf{U} = \{0,0,0,0,1,2,3,4,5,5,5,5\}$, and with control points $\mathbf{P}_i$ with $i = 0,1,\ldots,7$. This curve has 5 **spans**.



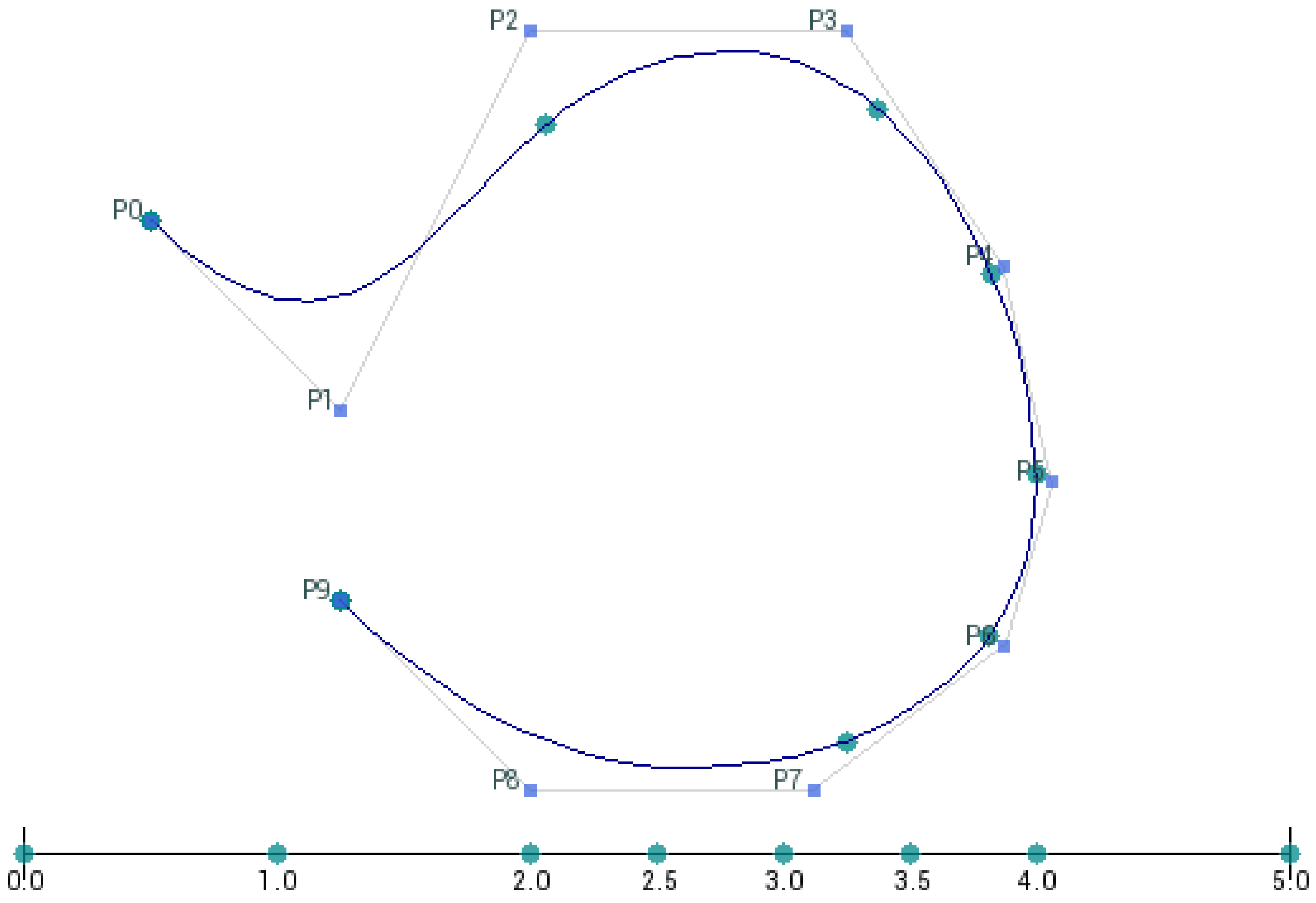
We will insert a knot at $\bar{u} = 2.5$.

To obtain $\mathbf{U}=\{0,0,0,0,1,2,2.5,3,4,5,5,5,5\}$, with a new vector of control points $\mathbf{P}_i$ with $i=0,1,\ldots,8$. The new curve has 6 **spans**.



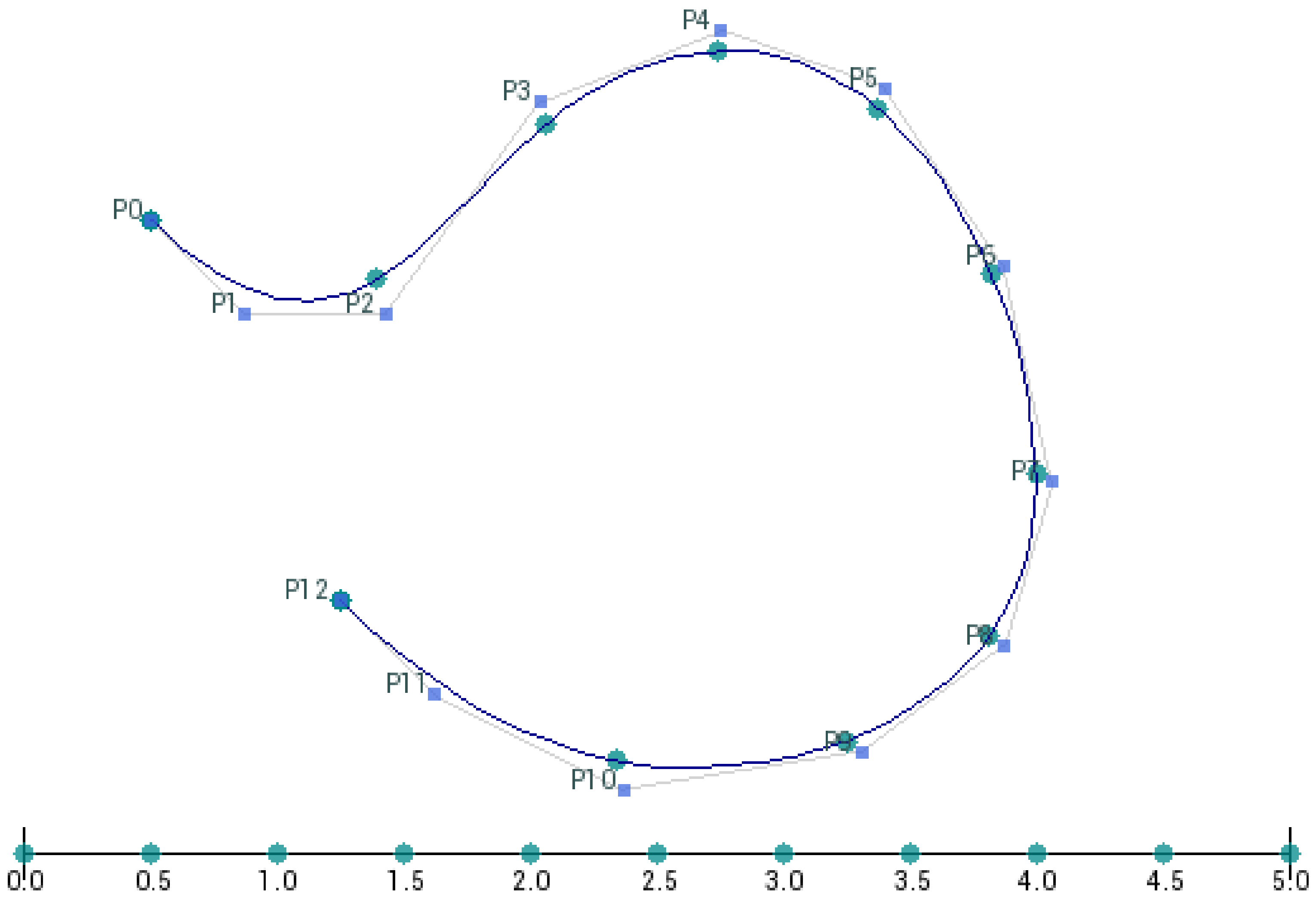
Next, we will insert a knot at $\bar{u}=3.5$.

The new knot vector is, $\mathbf{U}=\{0,0,0,0,1,2,2.5,3,3.5,4,5,5,5,5\}$. The new vector of control points $\mathbf{P}_i$ with $i=0,1,\ldots,9$. 7 spans have been formed.



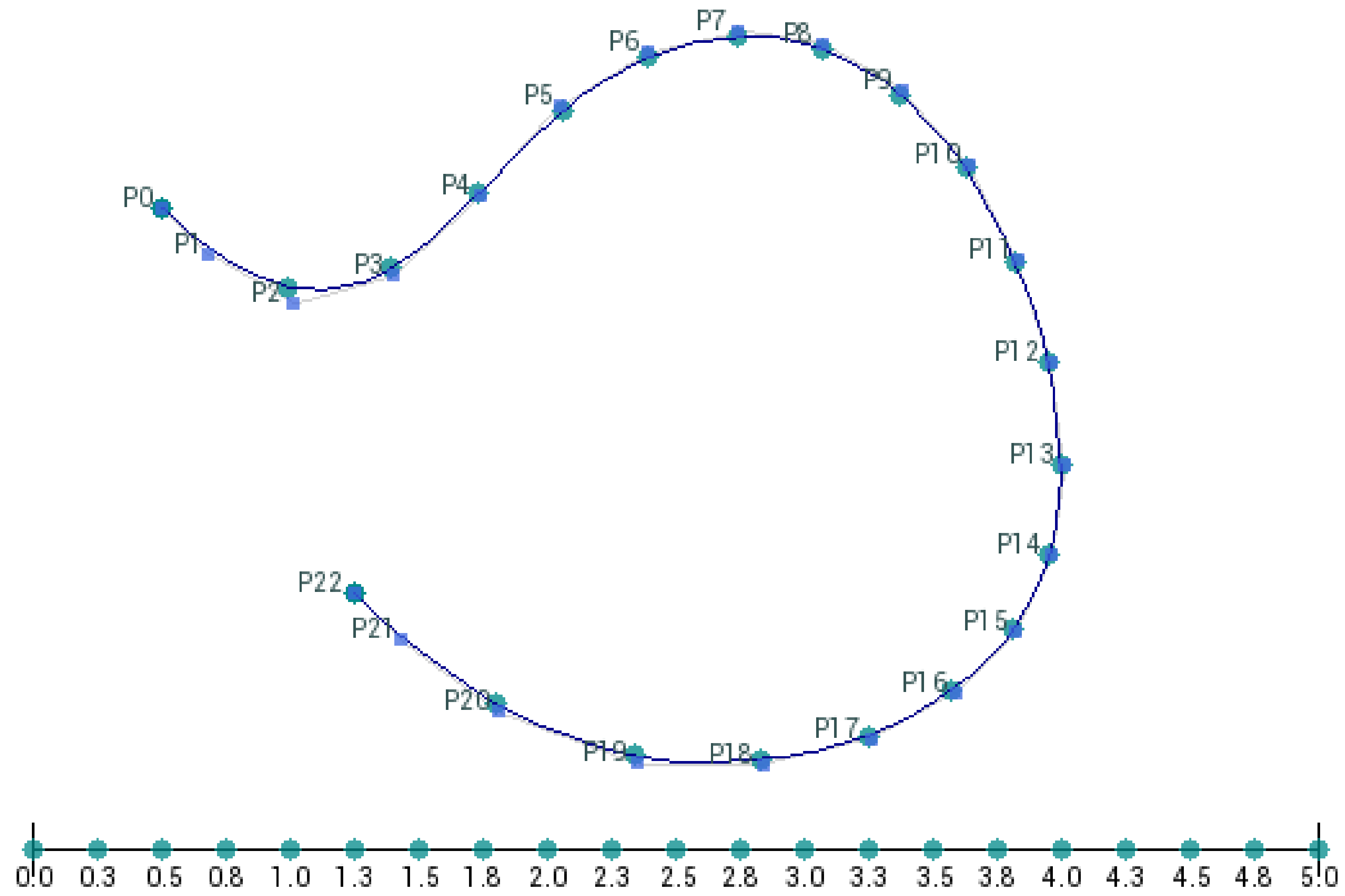
To complete the division, we will insert the knots $\{0.5,1.5,4.5\}$.

Now $\mathbf{U}=\{0,0,0,0,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5,5,5\}$, and we have 13 control points $\mathbf{P}_i$, with $i=0,1,\ldots,12$. The new curve has 10 **spans**.



We can continue inserting knots to have a finer mesh.

For this curve $\mathbf{U}=\{0,0,0,0,0.25,0.5,0.75,\ldots,4.25,4.5,4.75,5,5,5,5\}$, with a total of 23 control points $\mathbf{P}_i$, with $i=0,1,\ldots,22$. The total number of **spans** is 20.



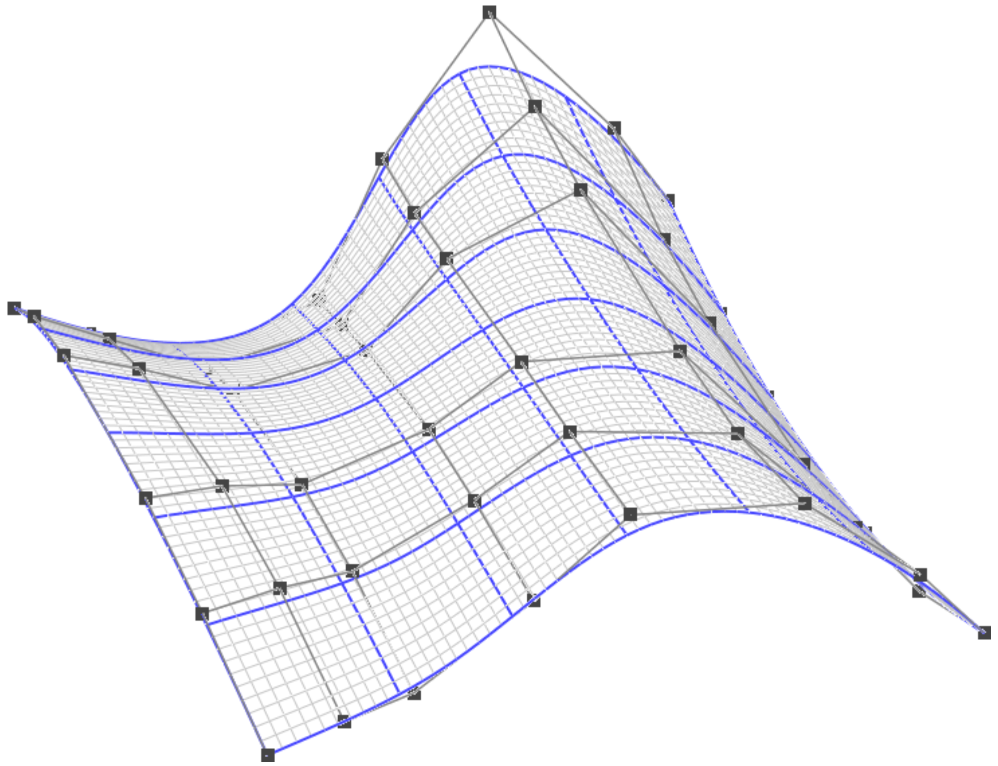
Knots insertion in surfaces (a.k.a. meshing surfaces)

A B-spline surface is defined as [Pieg97],

$$\mathbf{S}(u,v)=\sum_{i=0}^{n_1}\sum_{j=0}^{n_2} N_{i,p}(u)\,N_{j,q}(v)\,\mathbf{P}_{ij},$$

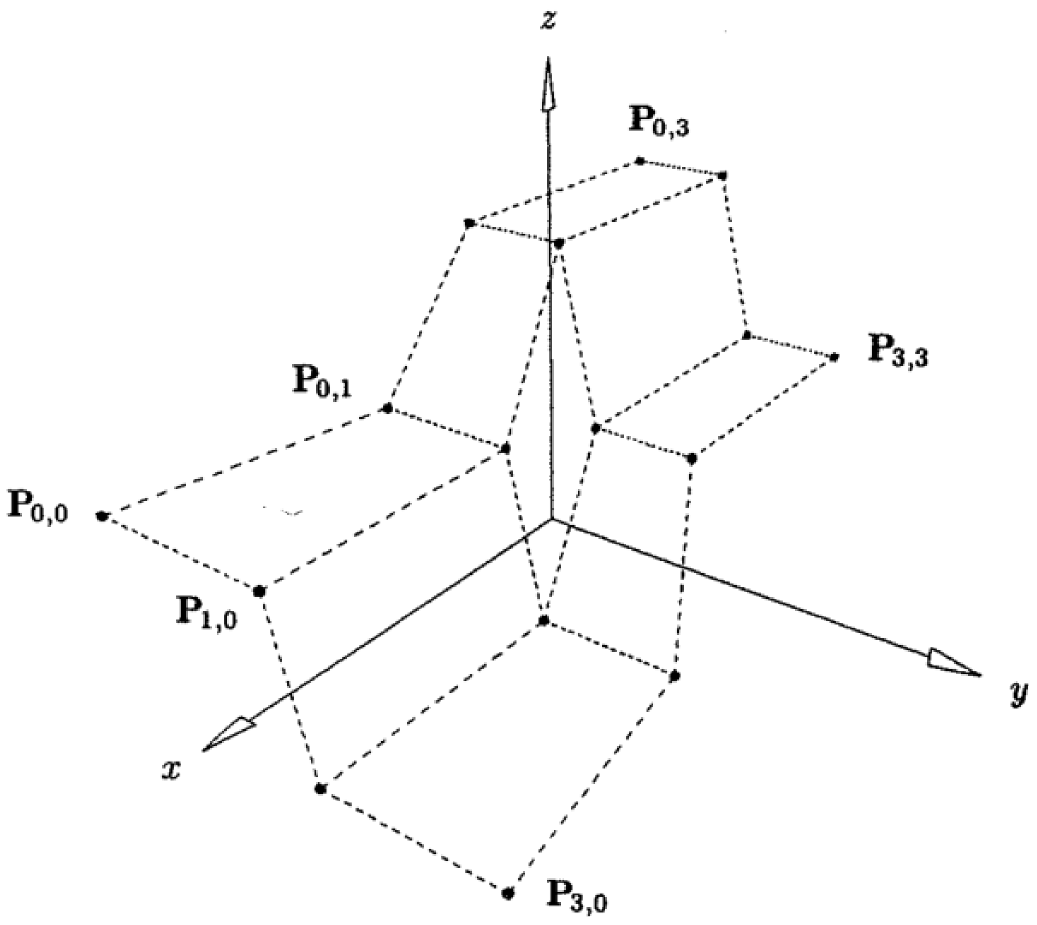
where $\mathbf{P}_{ij}$ is a **matrix of control points** (or coordinates). The basis $N_{i,p}(u)$ and $N_{j,q}(v)$ could have different degrees and knot vectors $\mathbf{U}=\{u_i\}$, $0\le i\le m_1$ and $\mathbf{V}=\{v_i\}$, $0\le i\le m_2$.

The procedure for meshing surfaces consists on inserting knots in both knot vectors.

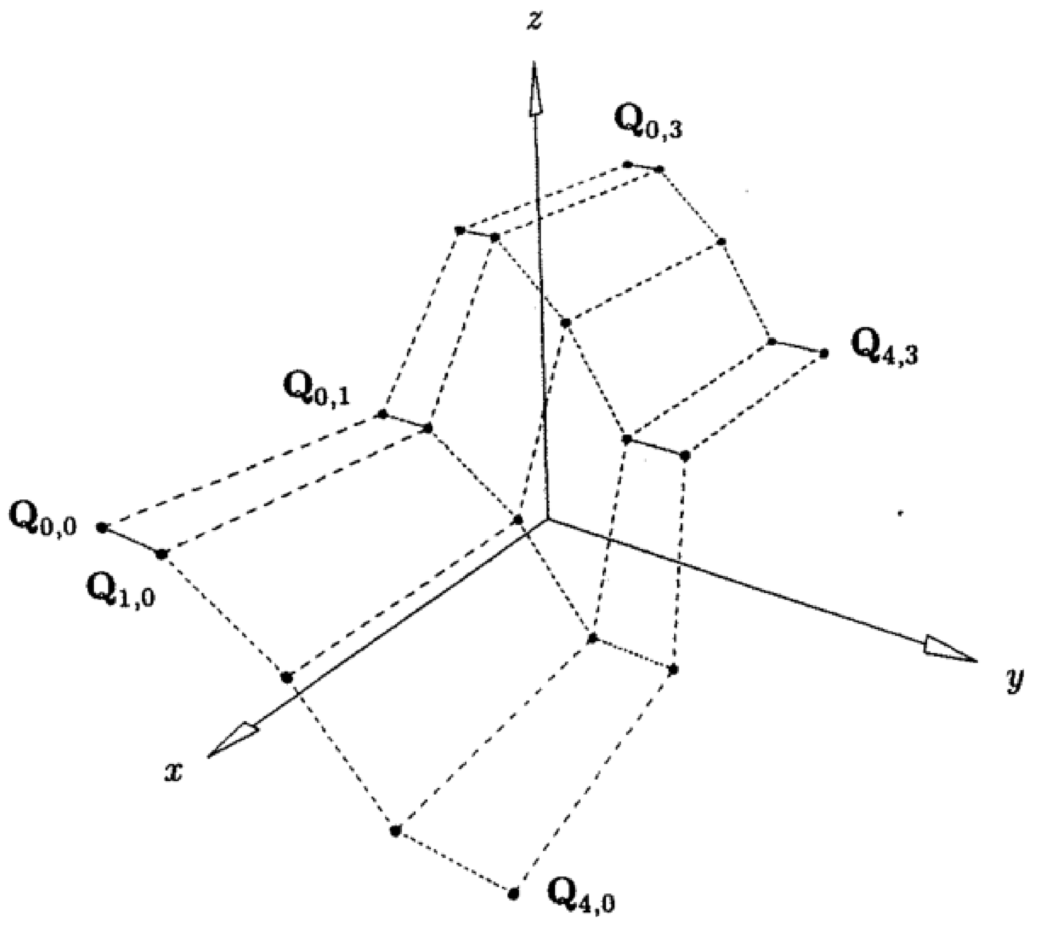


An **efficient algorithm** to insert knots is given in [Pieg97].

The following image shows the control points of a surface with knot vectors $\mathbf{U}=\{0,0,0,0,1,1,1,1\}$ and $\mathbf{V}=\{0,0,0,0.5,1,1,1\}$

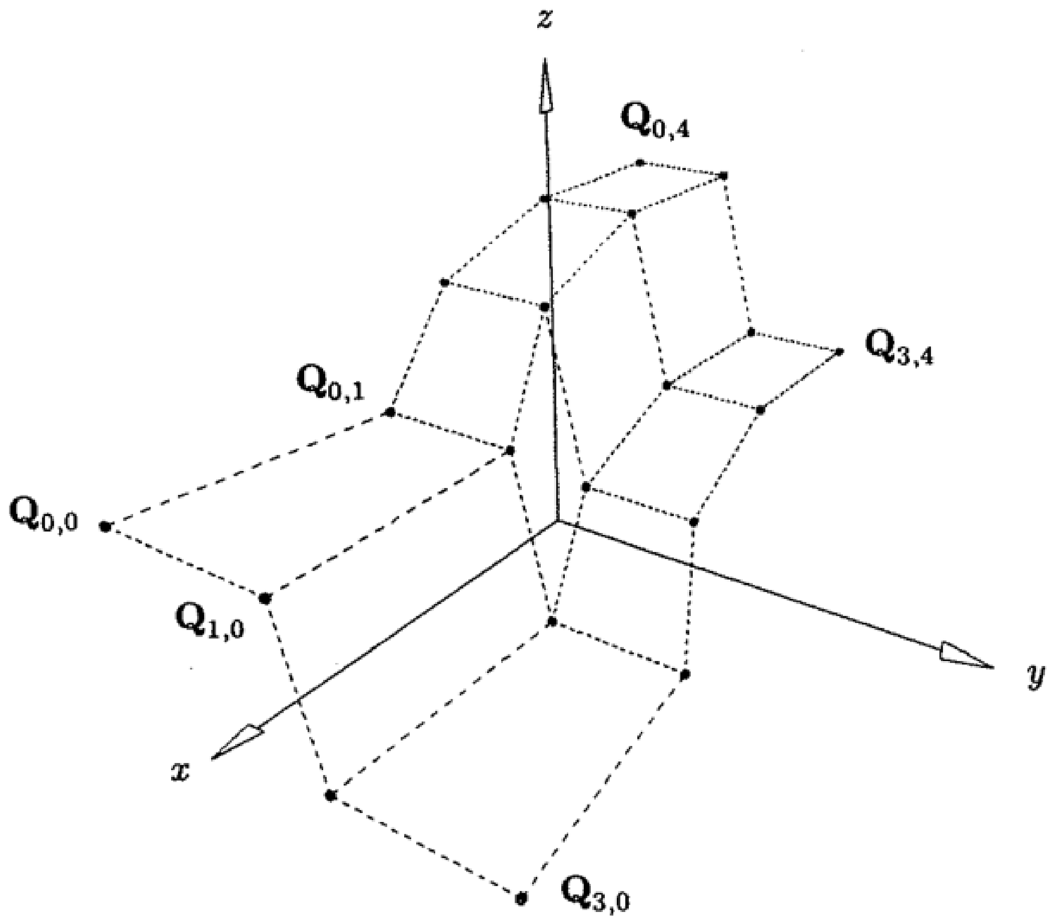


Original control points

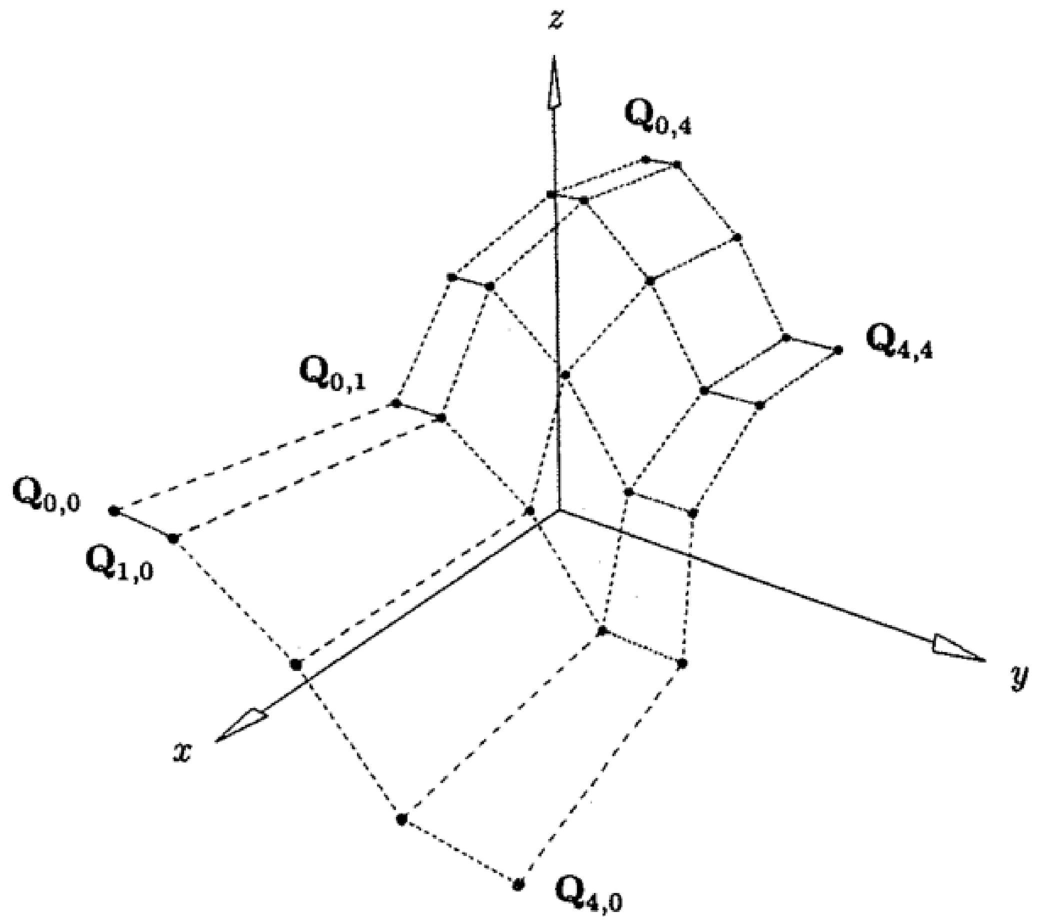


Inserting $u=0.4$

The knot insertion can be done on both knot vectors at the same time.



Inserting $v = 0.7$

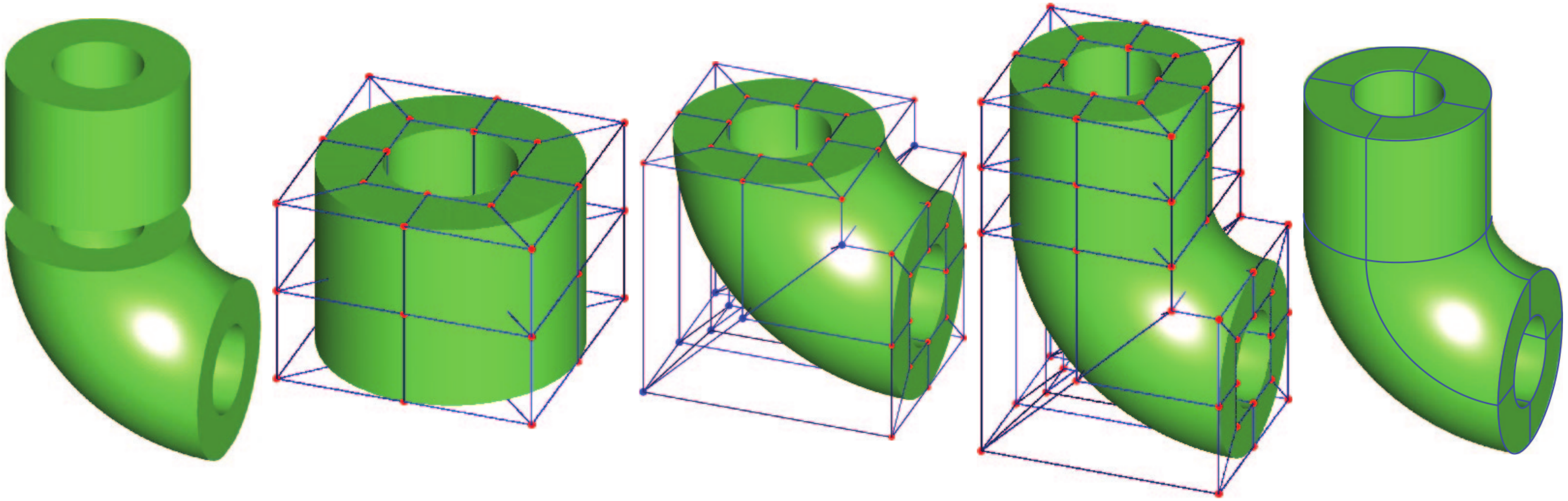


Inserting $u = 0.4$ and $v = 0.7$

Multiple patches (sub-domains)

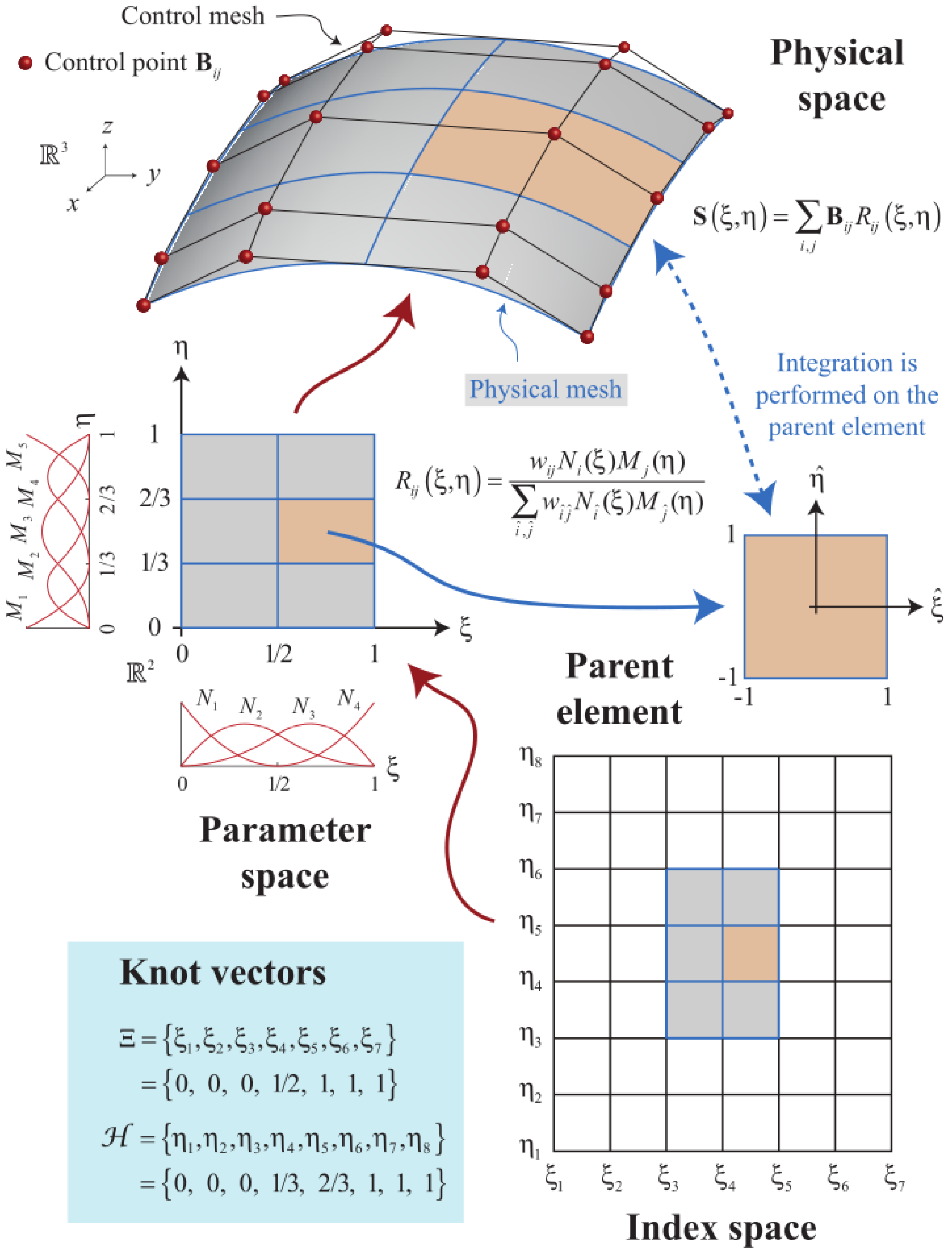
A NURBS object can not have a very complex shape, to handle domains with complex shapes it is necessary to divide the domain in patches of NURBS objects.

For example, a pipe, can be divided in two:



On the join of the two sub-domains the numeration of nodes must be unique. For this example, full domain has 8 elements.

Isogeometric analisys



Control mesh
Control point $\mathbf{B}_{ij}$
$\mathbb{R}^3$
Physical space
$\mathbf{S}(\xi,\eta)=\sum_{i,j}\mathbf{B}_{ij}R_{ij}(\xi,\eta)$
Physical mesh
Integration is performed on the parent element
$R_{ij}(\xi,\eta)=\frac{w_{ij}N_i(\xi)M_j(\eta)}{\sum_{i,j}w_{ij}N_i(\xi)M_j(\eta)}$
$\mathbb{R}^2$
Parameter space
Parent element
Knot vectors
$\Xi=\{\xi_1,\xi_2,\xi_3,\xi_4,\xi_5,\xi_6,\xi_7\}=\{0,\ 0,\ 0,\ 1/2,\ 1,\ 1,\ 1\}$
$\mathcal{H}=\{\eta_1,\eta_2,\eta_3,\eta_4,\eta_5,\eta_6,\eta_7,\eta_8\}=\{0,\ 0,\ 0,\ 1/3,\ 2/3,\ 1,\ 1,\ 1\}$
Index space

NURBS shape are defined by **control points**, these do not need to be in the domain.

The simulation program takes this set of control points (control mesh) to construct the domain or **physical space**.

The **normalized mapping** of this physical space is called **parameter space**.

This parameter space is discretized using a **structural mesh**, mapping this mesh backwards to the physical space creates the **physical mesh** of the domain.

The root idea behind isogeometric analysis is that the basis used to exactly model the geometry will also serve as the basis for the solution space of the numerical method.

Control variables

Let $\mathbf{C}(\xi)$ be a NURBS curve in 2D with $n+1$ control points $\mathbf{P}_i$ and basis functions $N_{i,p}(\xi)$, with $i=0,1,2,\ldots,n$

$$\mathbf{C}(\xi)=\sum_{i=0}^{n} N_{i,p}(\xi)\mathbf{P}_i, \text{ or } \begin{pmatrix} x(\xi) \\ y(\xi) \end{pmatrix}=\sum_{i=0}^{n} N_{i,p}(\xi)\begin{pmatrix} (\mathbf{P}_i)_x \\ (\mathbf{P}_i)_y \end{pmatrix}$$

with a knot vector $\boldsymbol{\Xi}=\{\xi_0, \xi_1, \ldots, \xi_m\}$.

We can build other functions over the patch in similar fashion, to map from the normalized space to the physical space

$$\hat{u}(\xi)=\sum_{i=0}^{n} N_{i,p}(\xi)d_i,$$

the coefficients d_i are called **control variables**.

"As with control points, the **non-interpolatory nature of the basis** prevents strictly interpreting the control variables as we can do with nodal values in FEA." [Cott09]

Lets remember that for any u, a maximum of $p+1$ basis functions $N_{i,p}(u)$ are different to zero.

An example, the Poisson equation in a 2D surface

We have two knot vectors,

$$\mathbf{\Xi}=\{\xi_0,\xi_1,\dots,\xi_{m_1}\},\ \mathbf{H}=\{\eta_0,\eta_1,\dots,\eta_{m_2}\},$$

with a matrix of control points $\mathbf{P}_{ij}$ of size $n_1\times n_2$. The basis $N_{i,p}(u)$ and $N_{j,q}(v)$ have degrees p and q respectively.

The functions of change of coordinates are the same functions that define the surface

$$\mathbf{S}(\xi,\eta)=\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}N_{i,p}(\xi)N_{j,q}(\eta)\mathbf{P}_{ij},\text{ or }\begin{pmatrix}x(\xi,\eta)\\ y(\xi,\eta)\end{pmatrix}=\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}N_{i,p}(\xi)N_{j,q}(\eta)\begin{pmatrix}(\mathbf{P}_i)_x\\ (\mathbf{P}_i)_y\end{pmatrix}$$

where is a **matrix of control points** (or coordinates).

We can define a **matrix of control variables** $\{\phi_{ij}\}$.

$$\phi(\xi,\eta)=\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}N_{i,p}(\xi)N_{j,q}(\eta)\phi_{ij}$$

Lets **remember again** that for any u, a maximum of $p+1$ basis functions $N_{i,p}(u)$ are different to zero. This means that the resulting system of equations will be sparse!!

We want to solve

$$f(\phi(x,y))=\frac{\partial}{\partial x}\left(k\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial\phi}{\partial y}\right)-S=0,$$

in a domain Ω with certain boundary conditions. Lets also define the flux as

$$\boldsymbol{q}=k\begin{pmatrix}\frac{\partial\phi}{\partial x}\\ \frac{\partial\phi}{\partial y}\end{pmatrix}.$$

By using a weak formulation, using a weight function W, we want to solve

$$\int_\Omega W\, f(\phi(x,y))\,dx\,dy=0,$$

this is equivalent to

$$\int_\Omega W\left[\frac{\partial}{\partial x}\left(k\frac{\partial\varphi}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial\varphi}{\partial y}\right)-S\right]d\Omega+\underbrace{W[\varphi-\bar{\varphi}]+W[q-\bar{q}]}_{\text{Boundary conditions}}=0.$$

Integrating by parts and using the definition of flux, we have

$$W\,q_x\big|_\omega-\int_\Omega\frac{\partial W}{\partial x}k\frac{\partial\phi}{\partial x}d\Omega+W\,q_y\big|_\omega-\int_\Omega\frac{\partial W}{\partial y}k\frac{\partial\phi}{\partial y}d\Omega-\int_\Omega W\,S\,d\Omega+W[\phi-\bar{\phi}]+W[q-\bar{q}]=0.$$

We will use the NURBS basis as weight functions.

For convenience, lets define

$$N_{ij} \overset{\text{def}}{=} N_{i,p}(\xi) N_{j,q}(\eta).$$

Therefore

$$\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\left(N_{ij} q_x\Big|_{\omega} - \int_{\Omega} \frac{\partial N_{ij}}{\partial x} k \frac{\partial \phi}{\partial x} d\Omega + N_{ij} q_y\Big|_{\omega} - \int_{\Omega} \frac{\partial N_{ij}}{\partial y} k \frac{\partial \phi}{\partial y} d\Omega - \int_{\Omega} N_{ij} S \, d\Omega + N_{ij}[\phi - \bar{\phi}] + N_{ij}[q - \bar{q}] \right) = 0.$$

For each element,

$$\sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\left[k \int_{\Omega^e} \frac{\partial N_{ij}}{\partial x} \frac{\partial \phi(x,y)}{\partial x} d\Omega^e + k \int_{\Omega^e} \frac{\partial N_{ij}}{\partial y} \frac{\partial \phi(x,y)}{\partial y} d\Omega^e \right] = \sum_{i=0}^{n_1}\sum_{j=0}^{n_2}\left[\int_{\Omega} N_{ij} S(x,y) \, d\Omega \right],$$

the integration will be done using gaussian quadrature (the description is below).

The Jacobian

Using

$$N_{ij} \overset{\text{def}}{=} N_{i,p}(\xi)\, N_{j,q}(\eta).$$

The change of variables is

$$\begin{pmatrix} \dfrac{\partial N_{ij}}{\partial \xi} \\ \dfrac{\partial N_{ij}}{\partial \eta} \end{pmatrix} = \underbrace{\begin{pmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{pmatrix}}_{\mathbf{J}^{\mathrm{e}}} \begin{pmatrix} \dfrac{\partial N_{ij}}{\partial x} \\ \dfrac{\partial N_{ij}}{\partial y} \end{pmatrix}.$$

If $\det \mathbf{J}^{\mathrm{e}} \neq 0$, then the inverse operation is given by

$$\begin{pmatrix} \dfrac{\partial N_{ij}}{\partial x} \\ \dfrac{\partial N_{ij}}{\partial y} \end{pmatrix} = \left(\mathbf{J}^{\mathrm{e}}\right)^{-1} \begin{pmatrix} \dfrac{\partial N_{ij}}{\partial \rho} \\ \dfrac{\partial N_{ij}}{\partial \eta} \end{pmatrix}.$$

Basis function derivatives

For a given polynomial order p and knot vector U, the derivative of the i-th basis function is given by

$$\frac{d}{du} N_{i,p}(u) = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u).$$

For higher derivatives [Cott09]

$$\frac{d^k}{du^k} N_{i,p}(u) = \frac{p}{u_{i+p} - u_i} \left(\frac{d^{k-1}}{du^{k-1}} N_{i,p-1}(u) \right) - \frac{p}{u_{i+p+1} - u_{i+1}} \left(\frac{d^{k-1}}{du^{k-1}} N_{i+1,p-1}(u) \right).$$

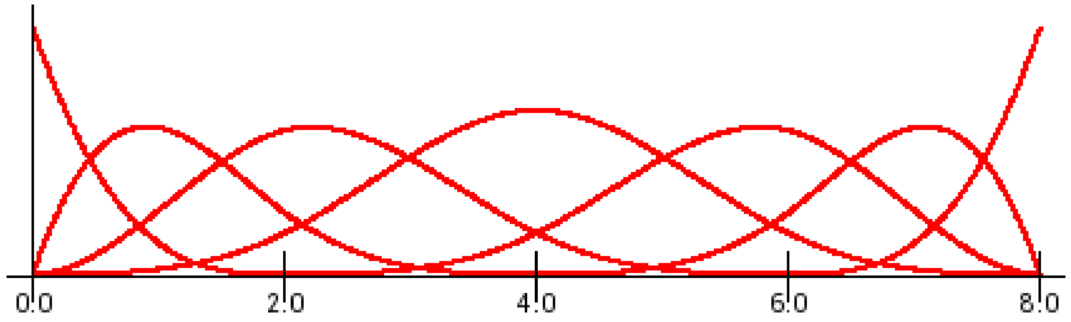
An **efficient algorithm** to calculate all the non-vanishing base functions derivatives is given in [Pieg97].

For a NURBS of p-degree we will have C^p-continuity on $N_{i,p}(u)$ for any $u \in (u_{0,} u_m)$ if there is not multiplicity in u.

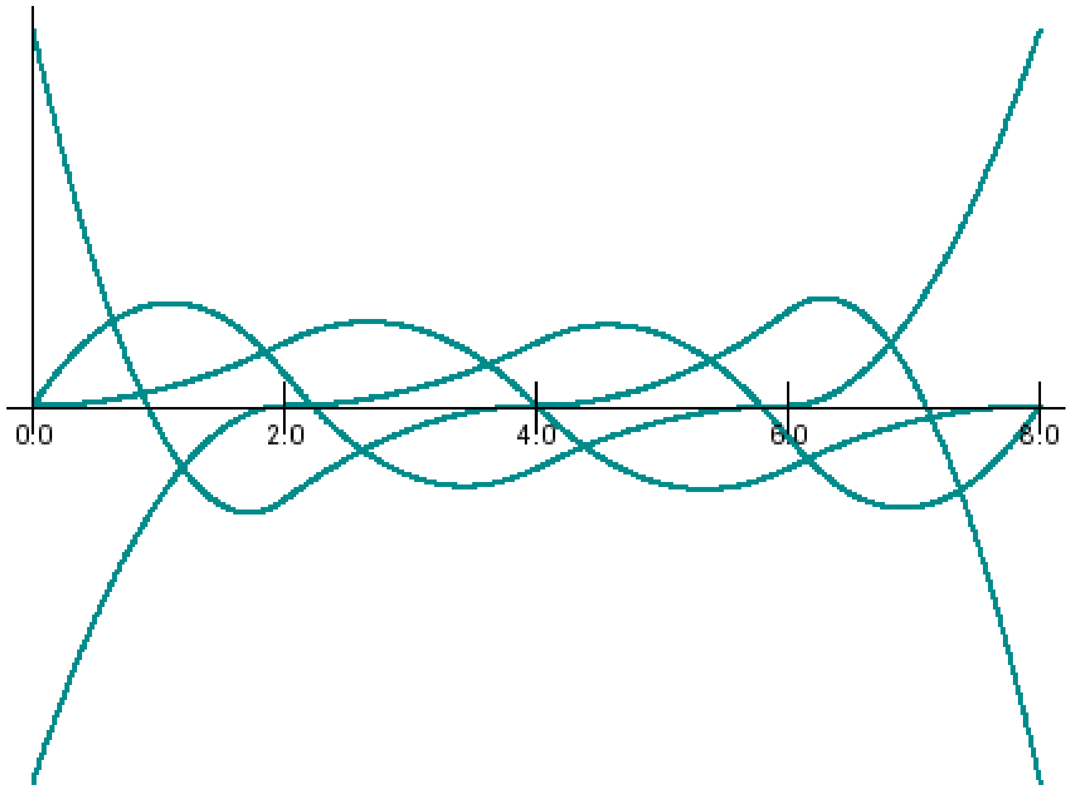
If there is multiplicity of knots of r, then we will have $C^{p-(r-1)}$-continuity on $N_{i,p}(u)$.

Example

For a knot vector $\mathbf{U}=\{0,0,0,0,2,4,6,8,8,8,8\}$, and degree 3, basis functions $N_{i,3}(u)$ are:

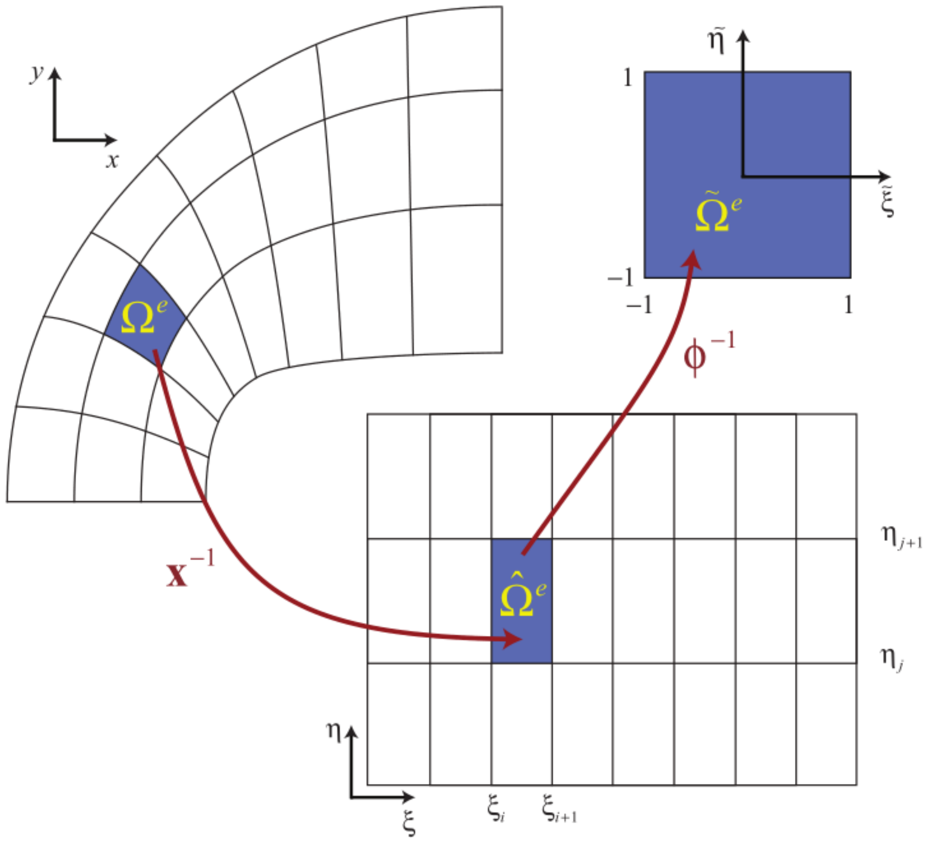


Their derivatives are: $\dfrac{d}{du}N_{i,3}(u)$:



Integration over elements

Integration is done using Gaussian quadrature in each element in the parameter space.



The evaluation of the shape functions $N_{i,p}(\xi)N_{j,q}(\eta)$ should be done in $\hat{\Omega}^e$.

We can calculate $(\xi,\eta)\in\hat{\Omega}^e$ from $(\tilde{\xi},\tilde{\eta})\in\tilde{\Omega}^e$ as [Cott99 p99]:

$$\xi=\xi_i+(\tilde{\xi}+1)\frac{(\xi_{i+1}-\xi_i)}{2},$$

$$\eta=\eta_i+(\tilde{\eta}+1)\frac{(\eta_{i+1}-\eta_i)}{2}.$$

Comparison of FEA and IGA

Finite Element Analysis
Nodal points
Nodal variables
Mesh
Lagrange basis functions
Basis interpolate nodal points and variables
h-refinement
p-refinement
Approximate geometry
Sub-domains

Isogeometric Analysis
Control points
Control variables
Knots
NURBS basis functions
Basis **does not** interpolate control points and variables
Knot insertion
Order elevation
Exact geometry
Patches

Summary

An analysis framework based on NURBS consists of [Hugh05]:

- A **mesh** for a NURBS patch is defined by the **product of knot vectors**. For example, in 3D, a mesh is given by $\mathbf{U} \times \mathbf{V} \times \mathbf{W}$.
- **Knot spans** subdivide the domain into “elements”.
- The **support** of each basis functions consists of a small number of “elements”.
- The **control points** associated with the basis functions define the geometry.
- The fields in question (displacement, velocity, temperature, etc.) are represented in terms of the same **basis functions** as the geometry. The coefficients of the basis functions are the degrees-of-freedom, or control variables.
- Mesh refinement strategies are developed from a combination of **knot insertion** and **order elevation** techniques.
- Arrays constructed from isoparametric NURBS patches can be assembled into **global arrays** in the same way as finite elements.
- The easiest way to set **Dirichlet** boundary conditions is to apply them to the control variables. In the case of homogeneous Dirichlet conditions, this results in exact, pointwise satisfaction. In the case of inhomogeneous Dirichlet conditions, the boundary values must be approximated by functions lying within the NURBS space.

References

[Cott09] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs. Isogeometric Analysis. Toward Integration of CAD and FEA. John Wiley & Sons. 2009.

[Pieg97] L. Piegl, W. Tiller. The NURBS Book. Second Edition. Springer. 1997.