

Spline-Based Finite Element Methods

A presentation in partial fulfillment of the Spacegrant scholarship requirements.

Arturo J. Fountain presenting



Spline-Based Finite Element Methods

Finite Element method

Spline theory

Numerical Methods

Finite Element Method

Approximate the true solution to a differential equation as the sum of weighted basis functions.

Formulations

Weak

Variational

Finite Elements

Finite Element Method

Mesh-based approach

E.g. triangular, hexahedral

Easily adapted to arbitrary boundaries

Time-consuming mesh generation required

Meshless approximations

E.g. Regular (square) grid

No mesh generation required

Complicated boundaries => problems

B-Splines

Univariate: (1-D)

Polynomials on bounded domain.

Recursively-defined

B-spline of order n is a weighted combination of B-splines of order $n-1$.

Symmetric

Provide optimal approximation order but...

Fail to meet FE requirements

Don't normally conform to essential BCs!

B-Splines

The uniform B-spline

$$b^n(x) = \int_{\mathfrak{R}} b^{n-1}(x) dx$$

or equivalently

$$\frac{d}{dx} b^n(x) = b^{n-1}(x) - b^{n-1}(x-1)$$

where

$$b^n(0) = 0$$

B-Splines

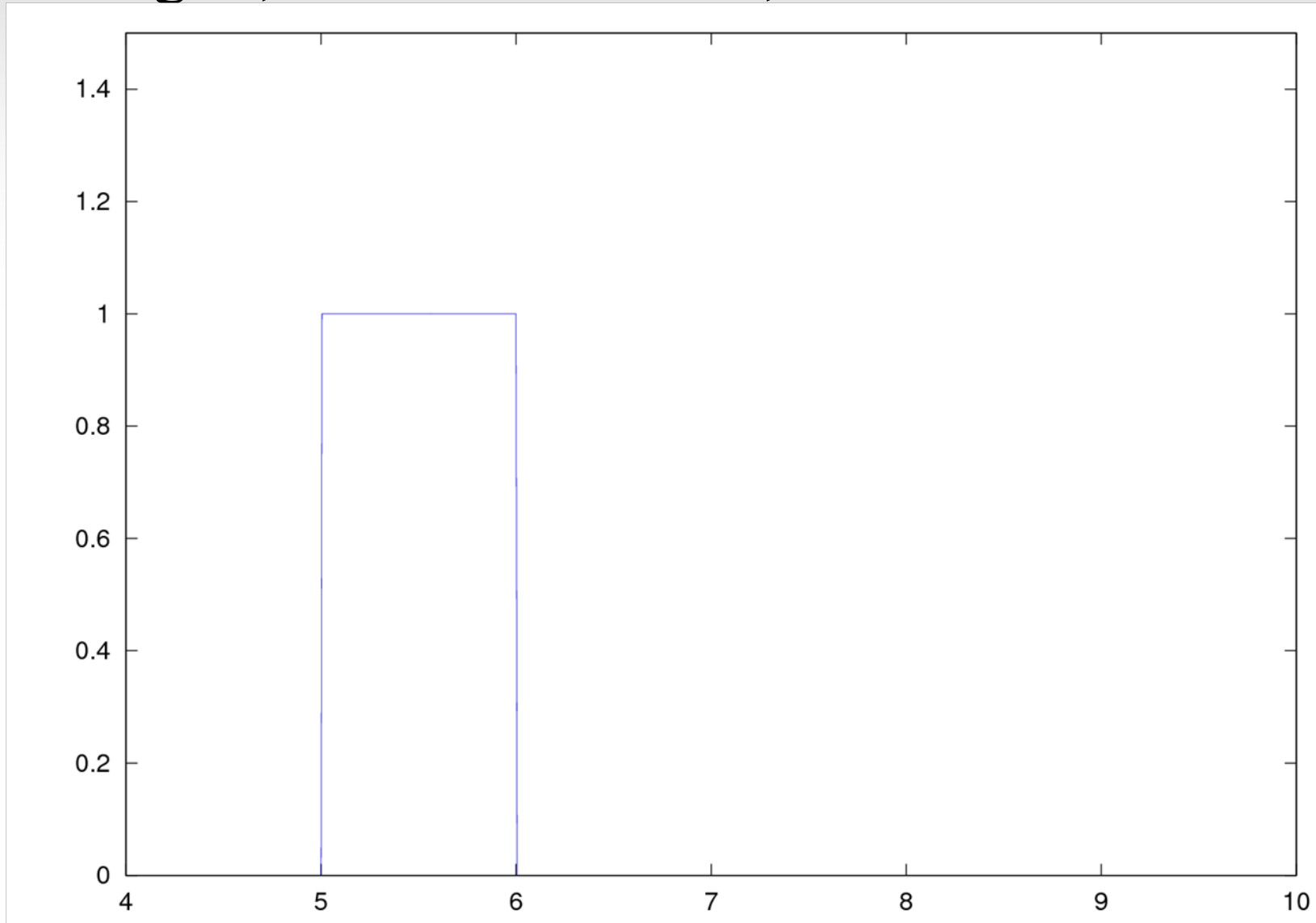
These univariate splines are nonzero only between the origin and $x = n+1$.

To allow the splines to occupy regions away from the origin, define scaling and translating

$$b_{k,h}^n(x) = b^n\left(\frac{x}{h} - k\right)$$

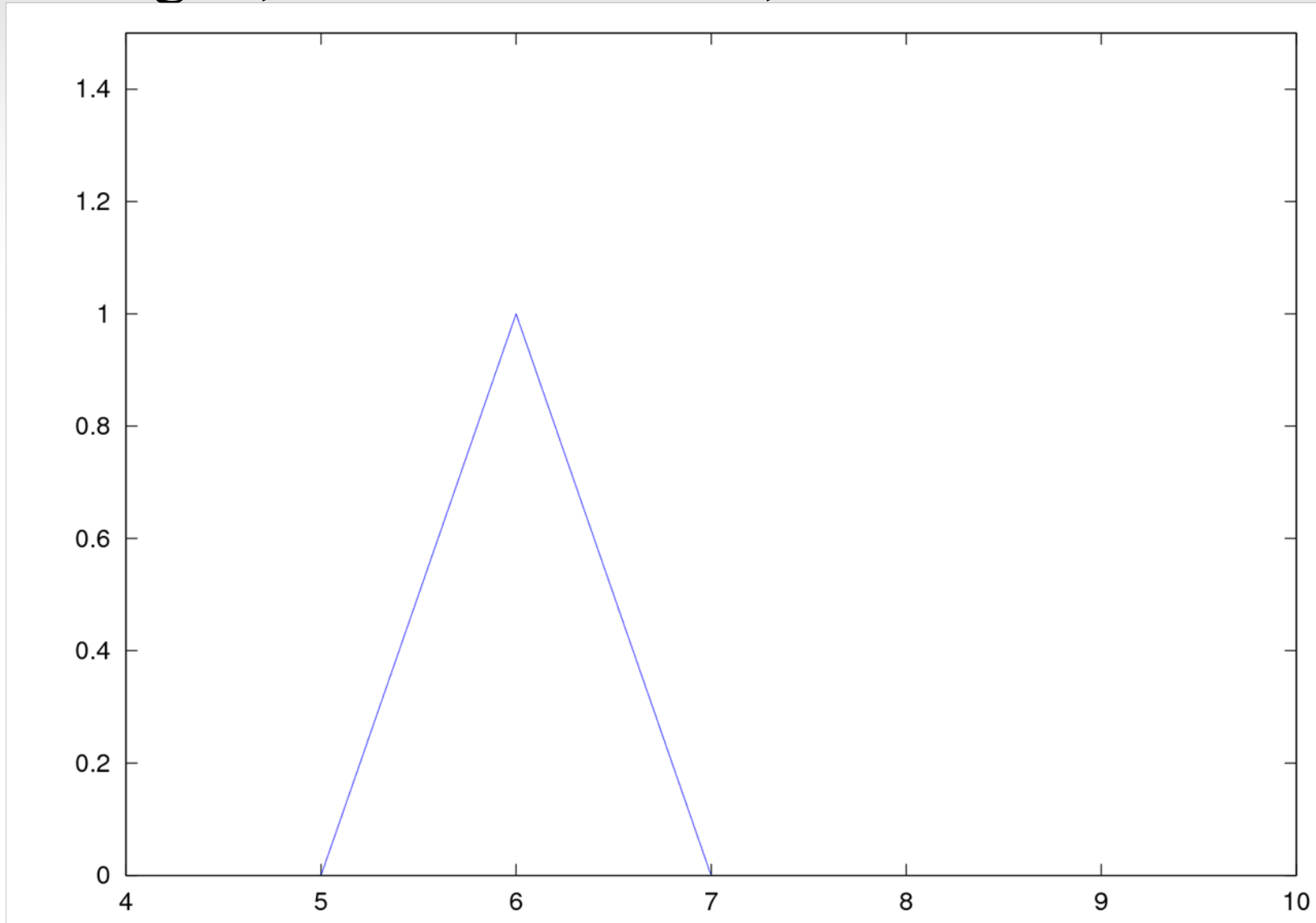
B-Splines

Scaling: 1, translated 5 units, order 0



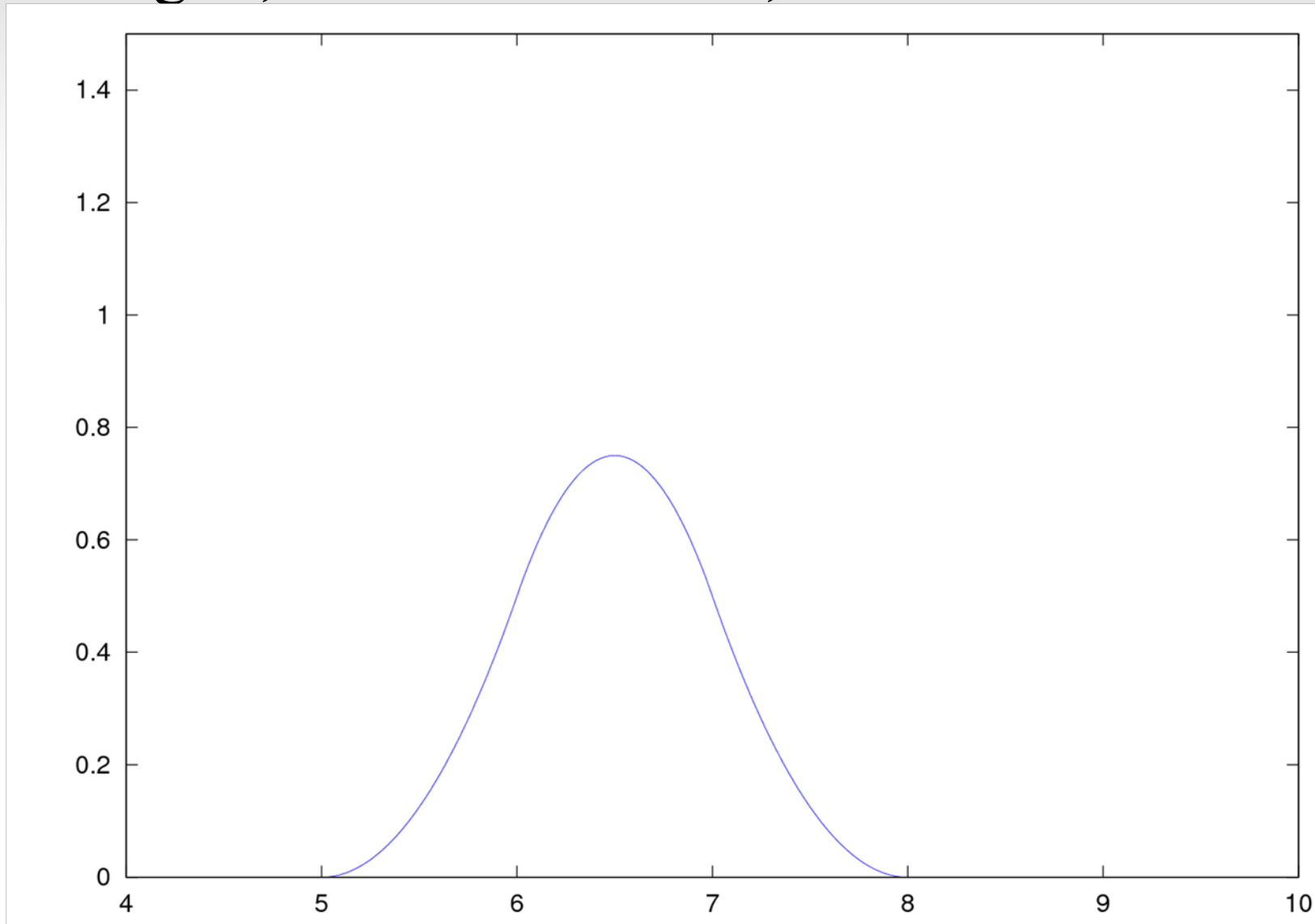
B-Splines

Scaling: 1, translated 5 units, order 1



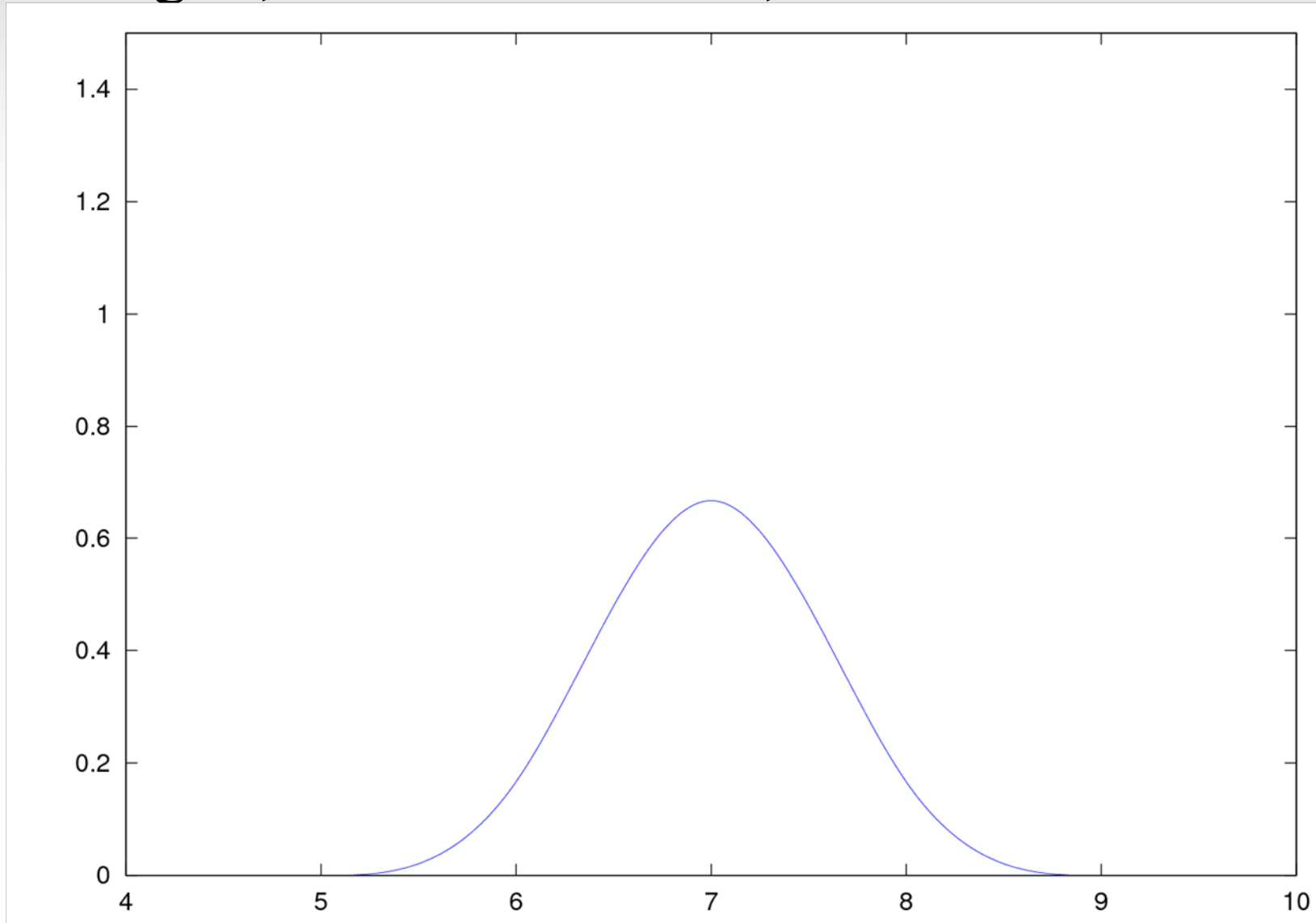
B-Splines

Scaling: 1, translated 5 units, order 2



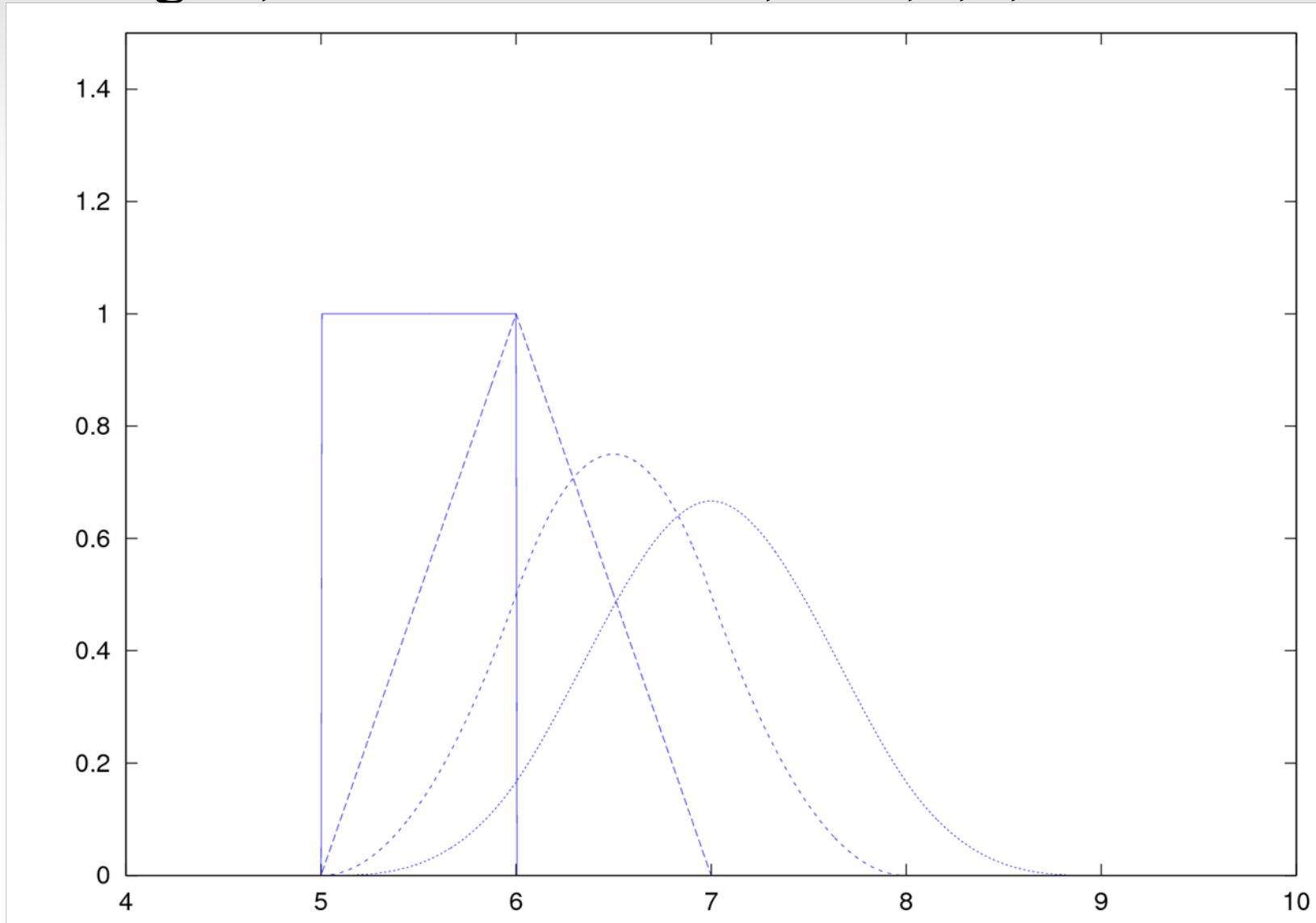
B-Splines

Scaling: 1, translated 5 units, order 3



B-Splines

Scaling: 1, translated 5 units, $n=0,1,2,3$



B-Splines

Multivariate

Usually define tensor product B-splines as

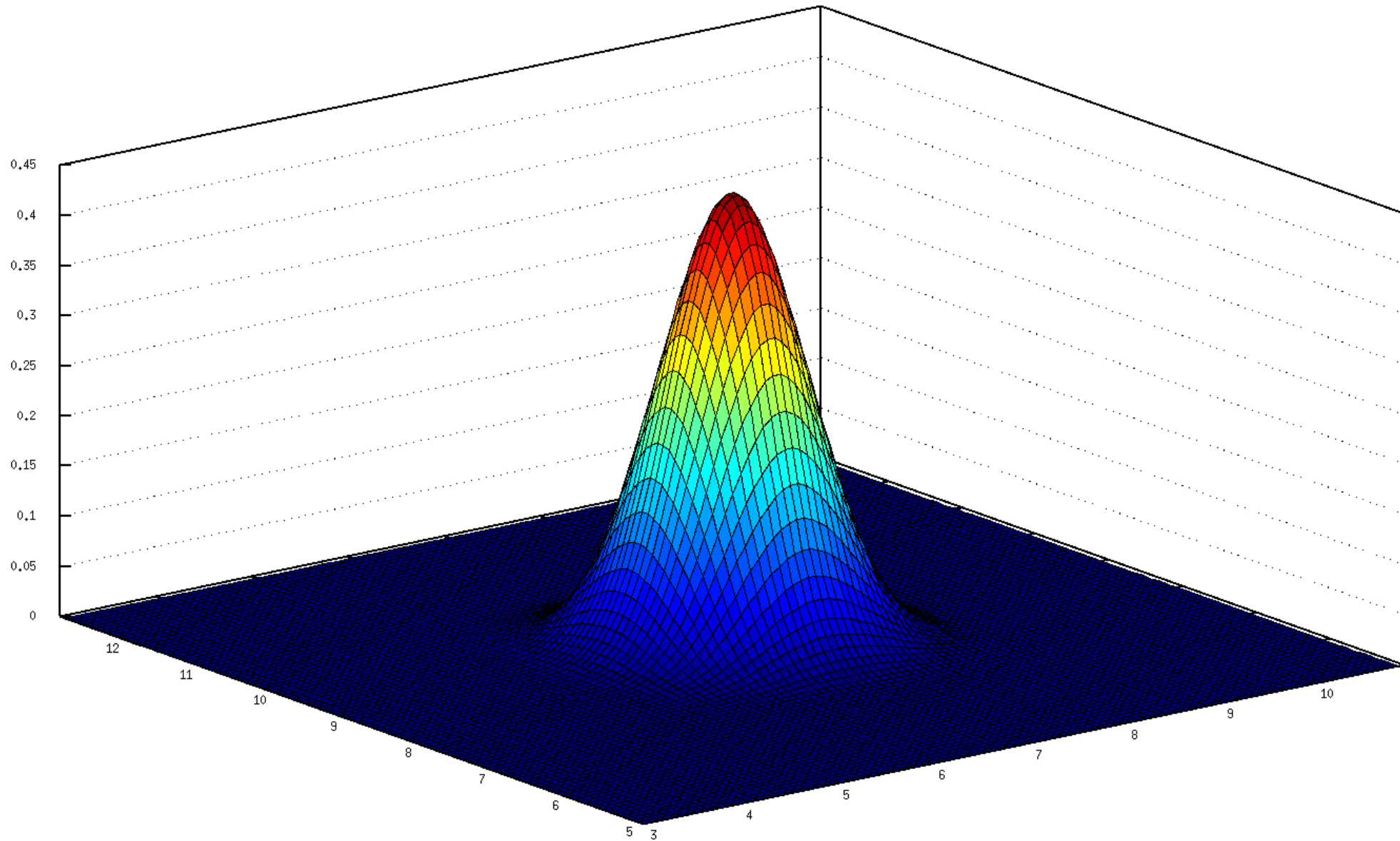
$$b_{k,h}^n(x) = \prod b_{k_v,h}^{n_v}(x_v)$$

In terms of the standard rectangular components

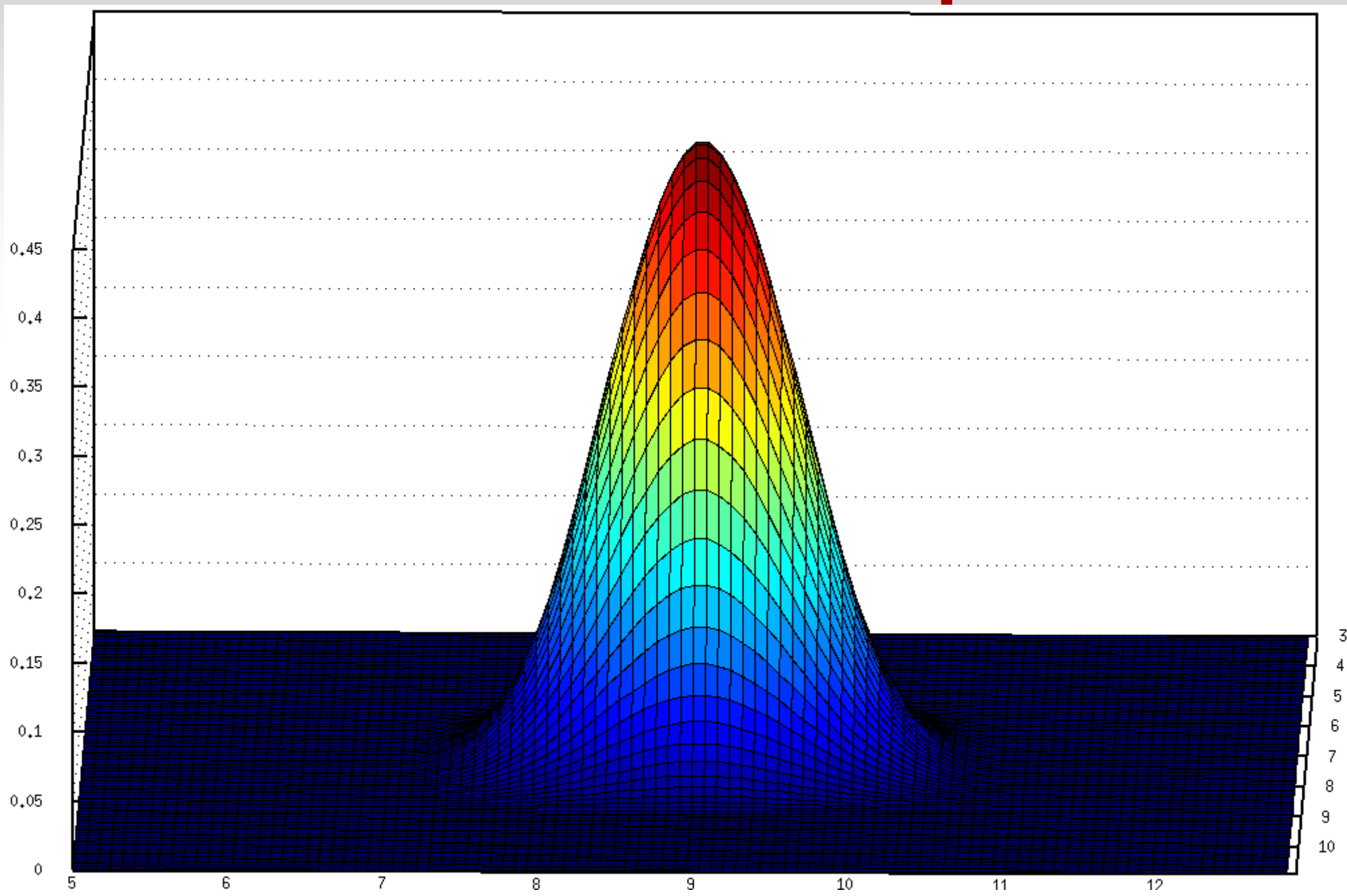
$$x_1 = x, x_2 = y, x_3 = z$$

Normally use a single, common order n .

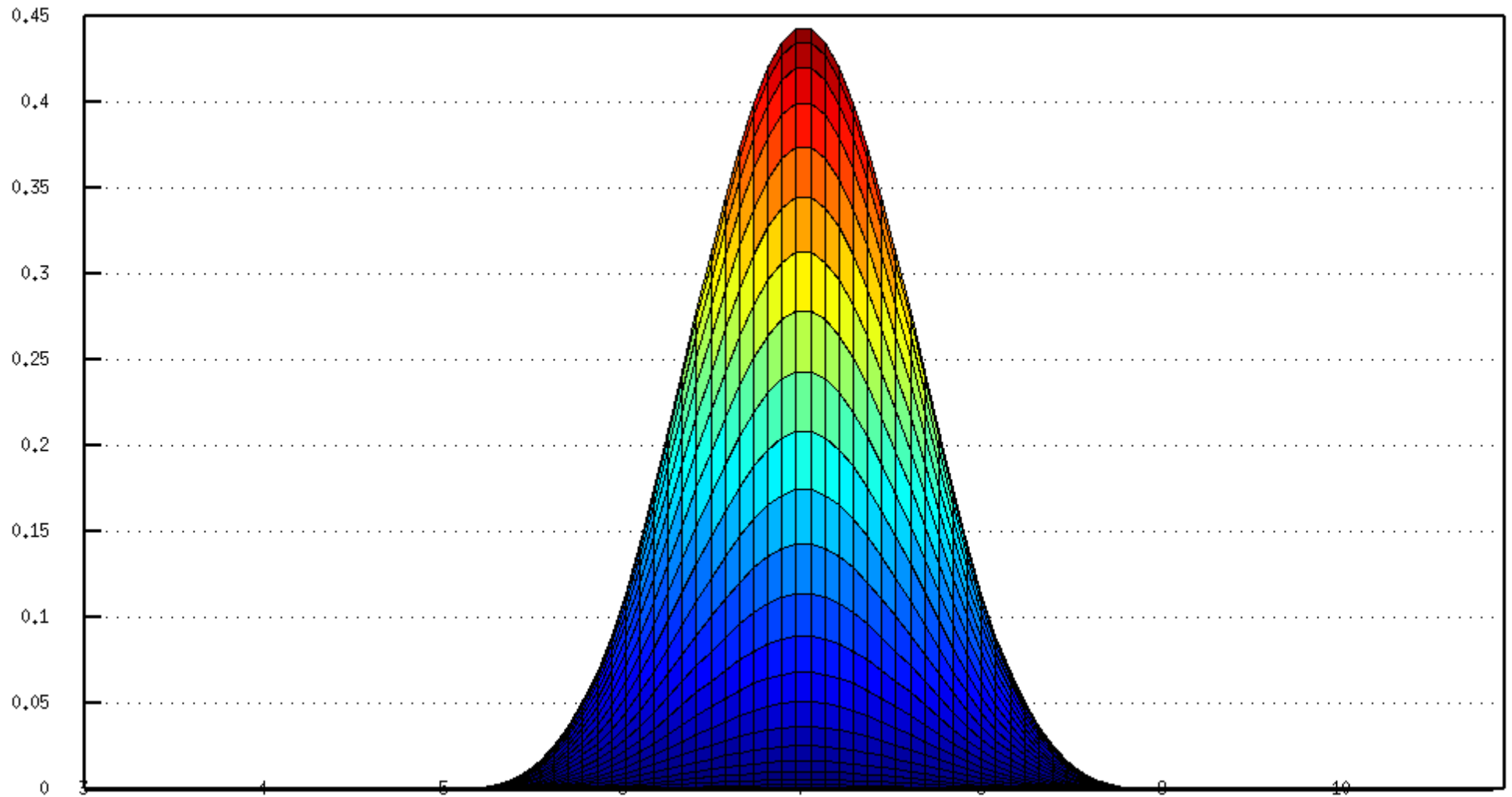
Tensor Product B-Splines



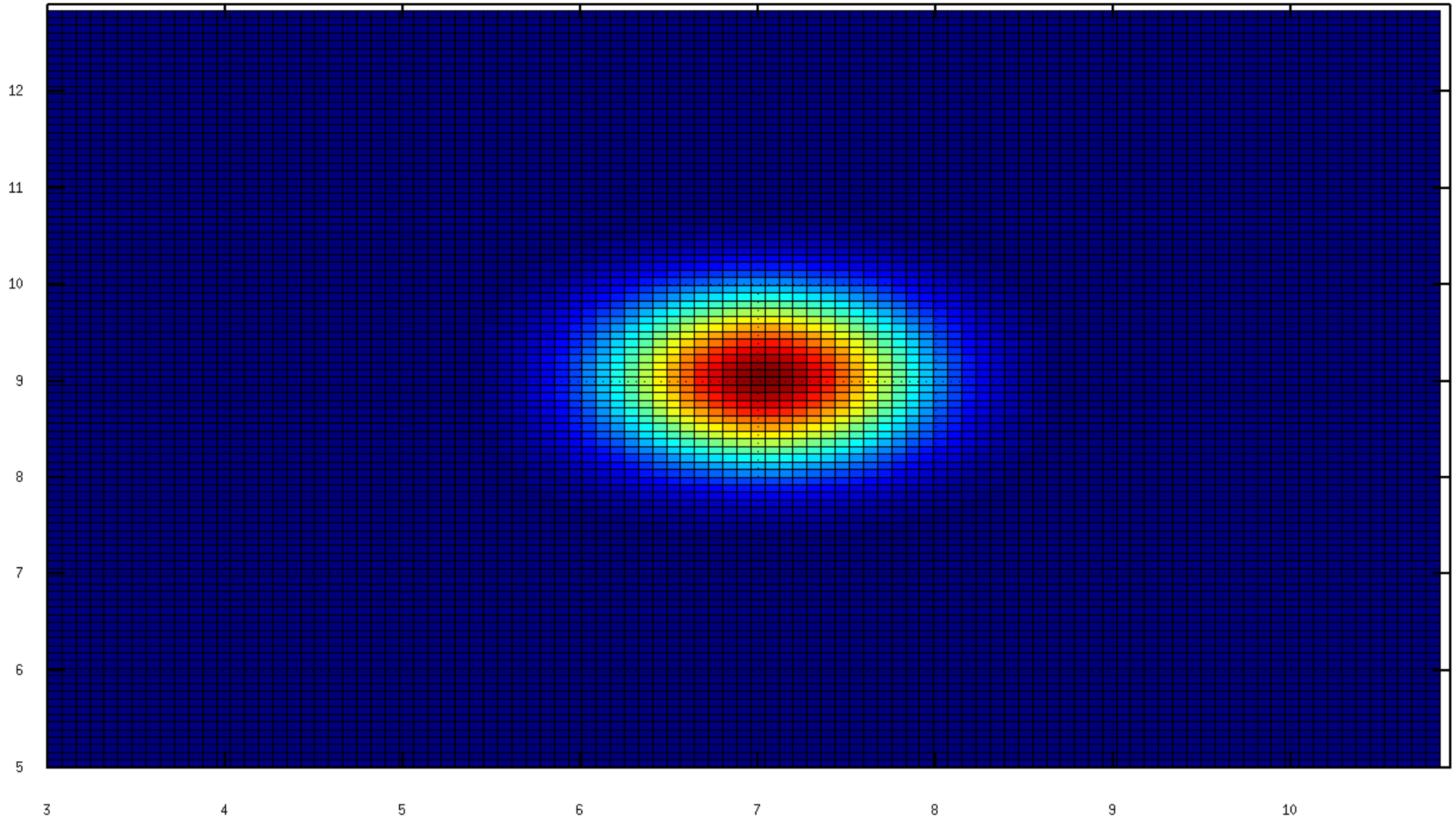
Tensor Product B-Splines



Tensor Product B-Splines



Tensor Product B-Splines



Tensor Product B-Splines

$$b_{k,h}^n(x) = \prod b_{k_v, h}^{n_v}(x_v)$$

Product-form yields simplification of mathematics.

E.g:

$$g_{k,l} = \int \nabla(b_{k,h}^n) \nabla(b_{l,h}^n)$$
$$= \sum_{v=1}^m d_{k_v - l_v}^n \prod_{u \neq v} s_{k_\mu - l_\mu}^n$$

where

$$s_{k-l}^n = hb^{2n+1} (n+1+l-l)$$

$$d_{k-l}^n = \frac{2s_{k-l}^{n-1} - s_{k-l-1}^{n-1} - s_{k-l+1}^{n-1}}{h}$$

Tensor Product B-Splines

Calculus reduced to arithmetic?

Not without penalties

Only for splines entirely contained in domain

Need to consider B-splines which are...

Inside the domain

Outside of the domain?

Intersecting the boundary!

Classification of Splines

B-Spline Classification

Consider grid cells: $Q = (l + [0, 1]^m)h$ and a domain D bounded by surface ∂D

Define cells as

Interior, $Q \subseteq \bar{D}$

Boundary, *Interior of Q intersects ∂D*

Exterior, $Q \cap D = \emptyset$

B-Spline Classification

Using the partitioned cells and the relevant B-splines

$$b_k, k \in K$$

distinguish between

Inner B-splines $b_i, i \in I$

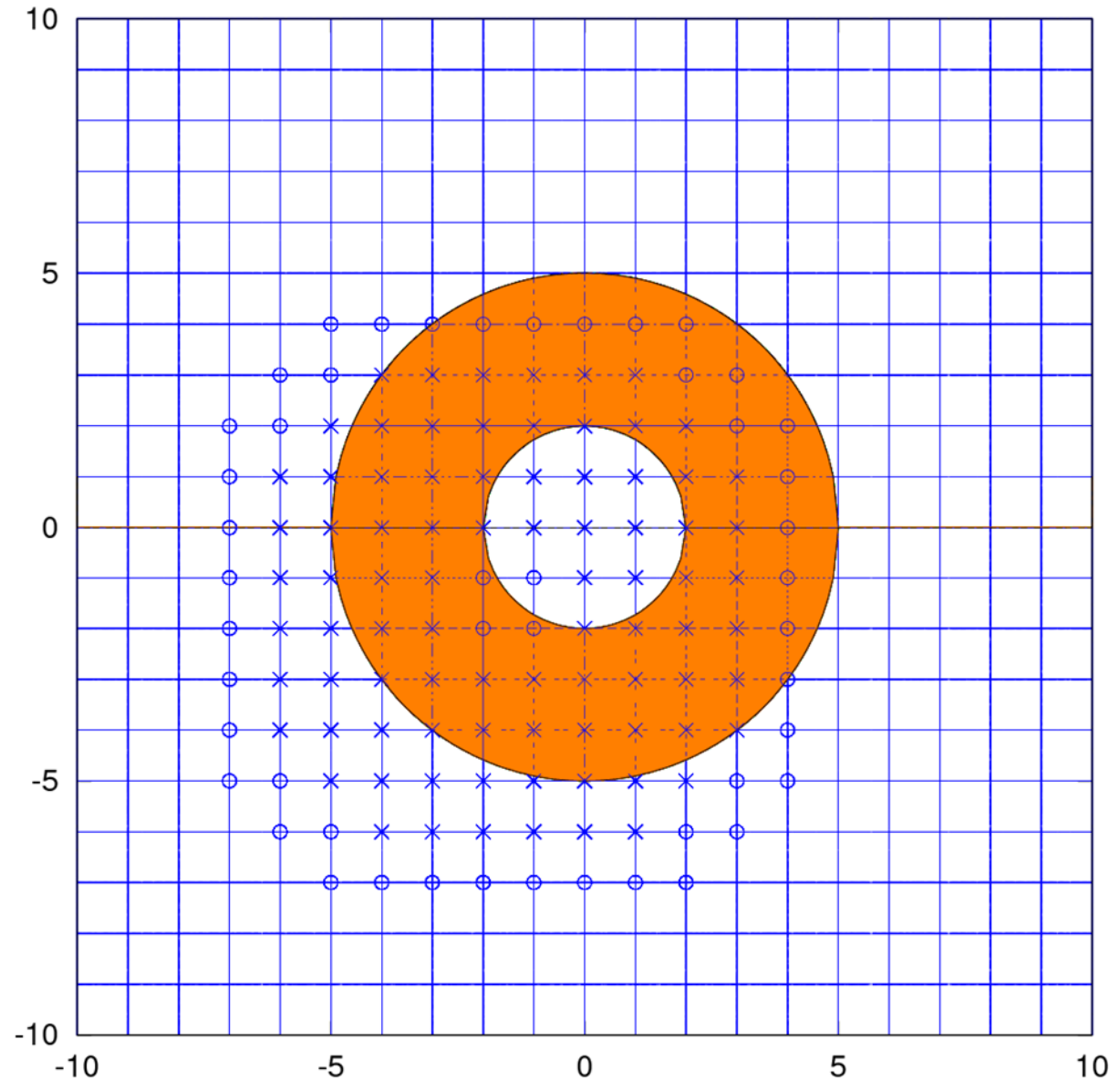
At least one interior cell in their support

Outer B-splines $b_j, j \in J = K \setminus I$

Support contains only boundary and exterior cells.

Coaxial Classification

Biquadratic spline space



WEB splines

B-splines are then

Extended

Linear combinations of outer, inner splines

Weighted

Multiplied by a weighting function which is

>0 inside the domain

<0 outside the domain

$=0$ on domain boundary

System matrix assembly

Build matrices

Ritz-Galerkin matrix G , source F

Extension matrix \tilde{E}

Linear algebra gives system ${}^eG^eU = {}^eF$

$${}^eG = \tilde{E}G\tilde{E}^t \quad {}^eF = \tilde{E}F$$

Solution coefficients of this system are multiplied with basis functions to yield approximate solution.

Progress

Code development

C libraries for...

Bspline classification, ~1800 lines

Numerical integration, ~500 lines

WEB-spline evaluation < 300 lines!

Expect ~2000 lines of code to use libraries to build the system matrix

Matrix assembly + supporting functions, 1000 lines

Progress

Background work mostly completed

Literature survey

Practical basis largely understood

Significant work toward analyzing realistic models.

Unresolved Issues

Matrix assembly

Weighting function concerns

Currently using analytic weighting functions

Complicated geometries make this impractical

R-Function Method?

Brute-force classification

Will benefits outweigh drawbacks?

Remaining Work

Resolve current issues

Parallelize code

Apply methodology to realistic data

VHP

Questions?

References

1. K. Hollig: Finite Element Methods with B-splines, SIAM, Philadelphia, 2003.
2. C. de Boor: A Practical Guide to Splines, Springer-Verlag, New York, 1978.