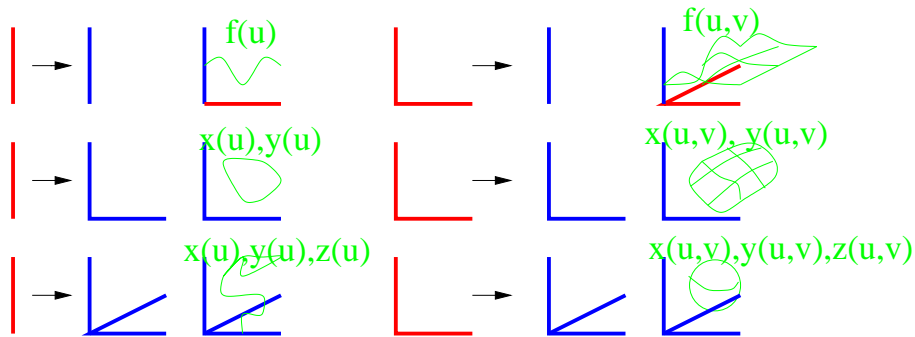


2a. Curved Geometry in 1 variable

glMap, gluNurbsCurve, splines, subdivision

Domain, Range and Maps



Graph of a function, 1-Manifolds, 2-Manifolds

scalar fields: $R^3 \mapsto R^1$
 vector fields: $R^3 \mapsto R^3$
 tensor fields: $R^3 \mapsto R^{3 \times 3}$

Polynomials and Polynomial Forms

A *polynomial* of degree d is an infinitely differentiable map such that its d th derivative is constant.

For example, Forward Differencing:

t	0	1	2	3	4	5
p	1	7	23	55	109	191
$D^1 p$	6	16	32	54	82	
$D^2 p$	10	16	22	28		
$D^3 p$	6	6	6			

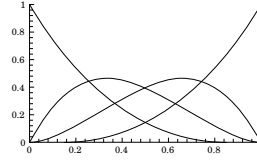
t	0	1	2	3	4	5
p	1	7	23	55	109	191
$D^1 p$	6	16	32	54	82	
$D^2 p$	10	16	22	28		
$D^3 p$	6	6	6			

OpenGL and Bezier Polynomials

A polynomial p in one variable of degree d is in *Bernstein-Bézier* form if

$$p = \sum_{i+j=d} c(i)B_{j,i},$$

$$B_{j,i}(u) = \frac{d!}{i!j!} u^i (1-u)^j.$$



Typically p is evaluated on the interval $[0, 1]$. See `10bezcurve.c`

- Define p by :


```
glMap1*(target,u_start, u_end, stride, order=degree+1, *points)
e.g. glMap1*(GL_MAP1_VERTEX_3, 0,1, 3,4,& c);
glEnable(GL_MAP1_VERTEX_3);
```

 Note: if $c(i) \in R^3$ then p maps to R^3 ; if the x, y, z components of $c(i)$ are stored sequentially then *stride* is 3.
- Evaluate all enabled 1-D maps (typically in display):

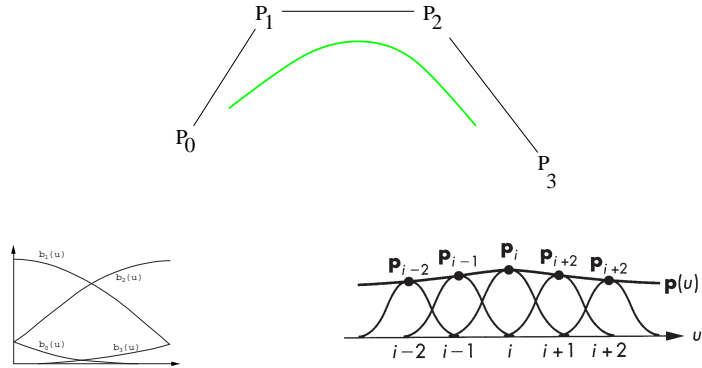

```
glEvalCoord(u)
or
glMapGrid1(n,u_start, u_end);
glEvalMesh1(GL_POINT, index_start, index_end)
```

For a detailed treatment of the BB-form: see handout

OpenGL and Splines

Non uniform rational B-spline representation.

- `gluNurbsCurve(obj, uknotcount, *uknot, ustride, &c,uorder, GL_MAP1_VERTEX)`



- `gluNewNurbsRenderer`, `gluDeleteNurbsRenderer`
- `gluNurbsProperty(obj,property,value)`
property: (GLU_DISPLAY_MODE, GLU_CULLING...)
- `gluNurbsCallback(obj,GLU_ERROR,(*fn)(errorCode))`

For a detailed treatment of the BB-form: see handout