DOI: 10.2312/egt.20151046

Meshless Interpolations

Meshless Interpolations for Computer Graphics, Visualization and Games:

An Informal Introduction

An Introductory Tutorial

Vaclav Skala
University of West Bohemia
Faculty of Applied Sciences, Plzen
Czech Republic.
http://www.VaclavSkala.eu

http://meshfree.zcu.cz

Meshless Interpolations

Schedule

Meshless Interpolations

Welcome and Introduction

Meshless Interpolations

Plzen (Pilsen) City

Plzen is an old city [first records of Plzen castle 976] city of culture, industry, and brewery.

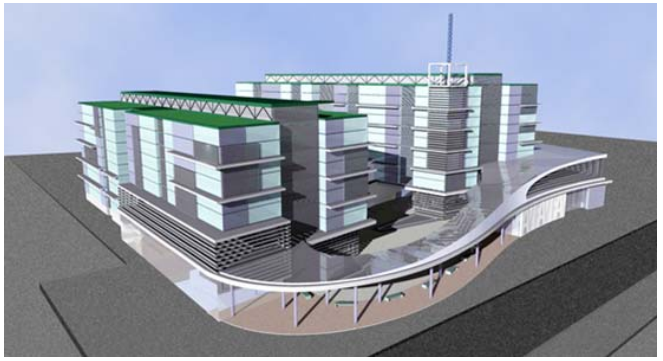
City, where today's beer fermentation process was invented that is why today's beers are called Pilsner - world wide

Meshless Interpolations

University of West Bohemia 17530 students + 987 PhD students

Faculty of Applied Sciences
Computer Science and Engineering **Mathematics** (+ Geomatics)
Physics **Cybernetics** **Mechanics** (Computational)

- Over **50%** of income from research and application projects
- NTIS project (investment of 64 mil. EUR)
- 2nd in the ranking of Czech technical / informatics faculties 2009, 2012



Meshless Interpolations

"Real science" in the XXI century

Courtesy of the Czech Film, Barrandov

Meshless Interpolations

History of Interpolation, Visualization and Geometric Modeling

Meshless Interpolations

History of interpolation

Wikimedia commons

- Interpolation is related to astronomical data processing – finding unknown values, filling gaps in tables
- Dated to Babylon, Mesopotamia– 300 BC Linear interpolation used to predict sun's position, planting crops
- Greece – 150 BC – Hipparchus of Rhodes –used "Chord function", similar $sin(x)$, for celestial bodies positions
- Chinese Liu Zhuo – interpolation formula close to Gregory-Newton's *second* order interpolation used for "Imperial Standard Calendar"

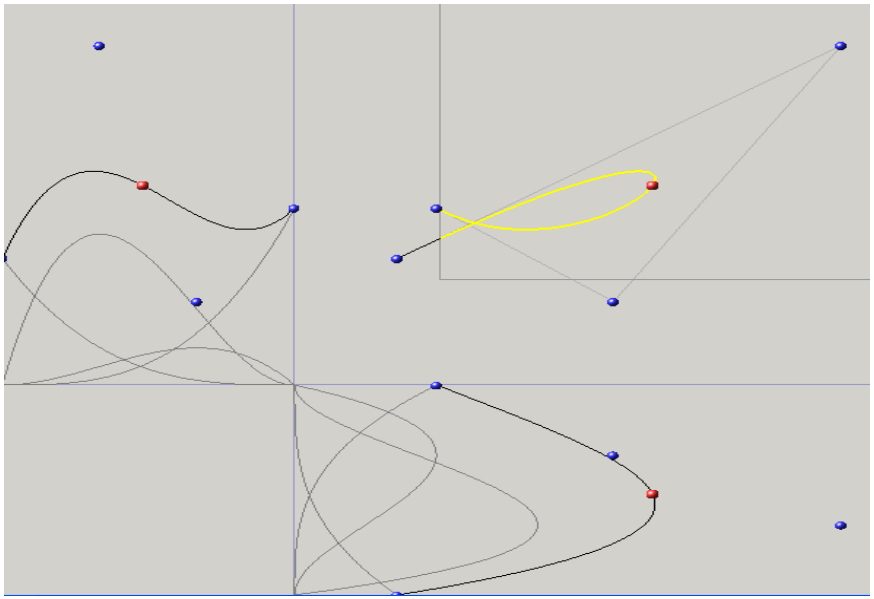
Erik Meijering: A Chronology of Interpolation – From Ancient Astronomy to Modern Signal and Image Proceesing, Proc.IEEE, Vol.90, No.3, pp.319-342, March 2002

Meshless Interpolations

Parametric non-linear interpolation

- Lagrange interpolation

- Bezier surfaces – an arbitrary degree 1962
- Third order interpolation by Catmull & Rom 1974
- Radial Basis Function (RBF) interpolation – Hardy 1971
- Splines properties demonstrated by Andrews & Hou 1978
  => B-splines
- 1999- convolution method development -> Splines



Lagrange,J.L.: "*The method of interpolation is, after logarithms, the most useful discovery in calculus*", [1792]

Refs: [Kav11], [Mei02],[Wor96]

Meshless Interpolations

History of visualization

- First notices 500-200 BC in Hindu and Egypt
- Visualization – offers a "visual" processing of complex large data sets
- Interpolation needed to fill gaps in data

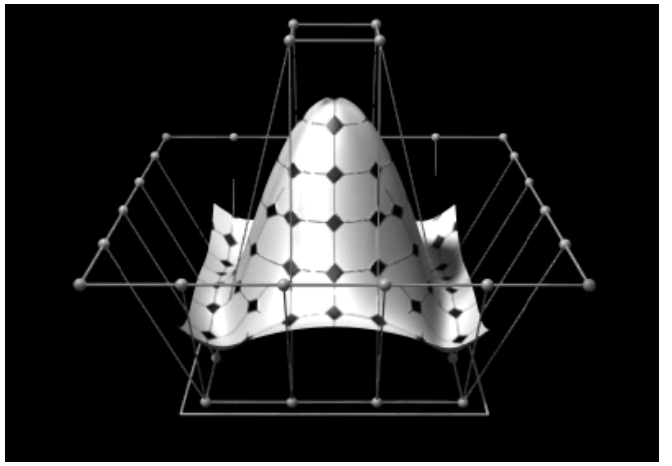
Babylon map
(Courtesy:Wikimedia commons)

Today

- visualization of complex, dynamic scalar, vector or tensor data
- 3D displays and 3D Prints
- Spatio-temporal scattered large data sets processing

Meshless Interpolations

History of geometric modeling


(Courtesy:Wikimedia commons)

- Geometric modeling = computational geometry + applied mathematics -> Shape description

- Developed especially in CAD/CAM industries – airplanes, rockets, automotive industry

- Pioneering work:
  - Ivan Sutherland – Sketchpad: A Man-machine Graphical Communications System, MIT PhD thesis, early 1960s
  - Pierre Bezier - The Mathematical Basis of the UNISURF CAD System, Butterworths (Renault)
  - Paul de Casteljau - de Casteljau algorithm (Citroen)

Today – sophisticated CAD/CAM, GIS etc. systems

BUT do we fully understand ALL of that?

Meshless Interpolations

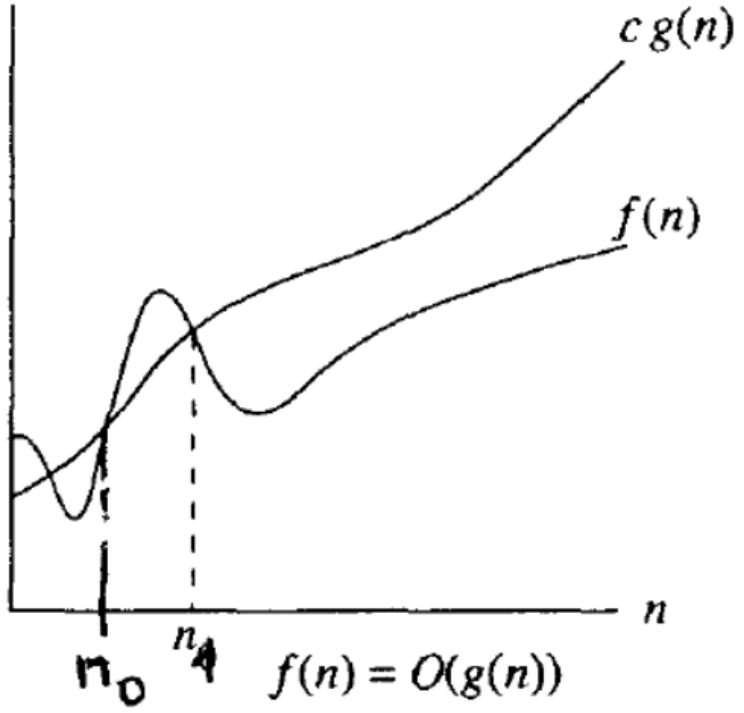
Algorithm complexity
(Computational geometry issue)

Meshless Interpolations

Algorithms are mostly evaluated by Computational Geometry (CG) terms $O(g(n))$, where n is number of primitives processed, where $n \to \infty$, which is not the real case as our algorithms will be processing $n \in \langle n_0, n_1 \rangle$, even very high.

Note that many CG approaches do not consider:

- limited speed of the data bus (data transfer cost)
- caching at the processor level
- parallelization on scalable HW
- etc.



So some algorithms can be actually faster even they have no optimal computational complexity from the CG view

Meshless Interpolations

Numerical representation and precision issues

Meshless Interpolations

Numerical systems

- Binary system is used nearly exclusively
- Octal & hexadecimal representation is used
- If we would be direct descendants of tetrapods –
  we would have a great advantage – "simple counting in octal or hexadecimal system"

	Name	Base	Digits[bits]	E min	E max
BINARY					
B 16	Half	2	10+1	−14	15
B 32	Single	2	23+1	−126	127
B 64	Double	2	52+1	−1022	1023
B 128	Quad	2	112+1	−16382	16383
DECIMAL					
D 32		10	7	−95	96
D 64		10	16	−383	384
D 128		10	34	−6143	6144

IEEE 758-2008 standard

The first tetrapods had eight fingers on each hand

Meshless Interpolations

Mathematically perfect algorithms fail due to limited numerical precision

Main issues

- stability, robustness of algorithms
- acceptable speed
- linear speedup – results depends on HW, CPU …. parameters !

Numerical stability

- limited precision of float / double
- tests A **?** B with floats

  if A = B **then** ….. **else** ….. ; **if** A = 0 **then** ….. **else** ….
  should be forbidden in programming languages

- division operation should be removed or postponed to the last moment if possible - “blue screen”, system reset, …

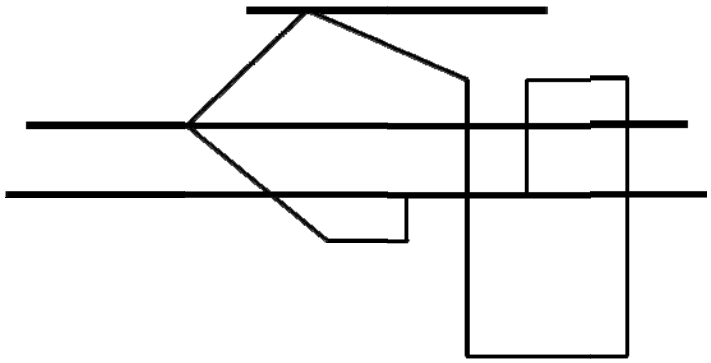
Meshless Interpolations

Typical examples of instability (non-sense tests in principal)

- intersection of 2 lines in E3
- point lies on a line in E2 or a on a plane in E3

$$Ax + By + C = 0 \quad \text{or} \quad Ax + By + Cz + D = 0$$

- k-sided polygon in E^3, $k \geq 4$
  it is not on a plane in general
- detection if a line
  - o intersects a polygon or
  - o touches a vertex or
  - o passes a vertex through it



Intersection of a line with a non-convex polygon is more complicated if is to be solved robustly; 3 value logic is to be used.

Skala,V.: Algorithms for 2D Line Clipping, in EUROGRAPHICS'89 Proceedings (Ed.W.Hansmann, F.R.A.Hopgood, W.Strasser), North Holland, ISBN 0-444-8813-5, pp.355-366, 1989

Meshless Interpolations

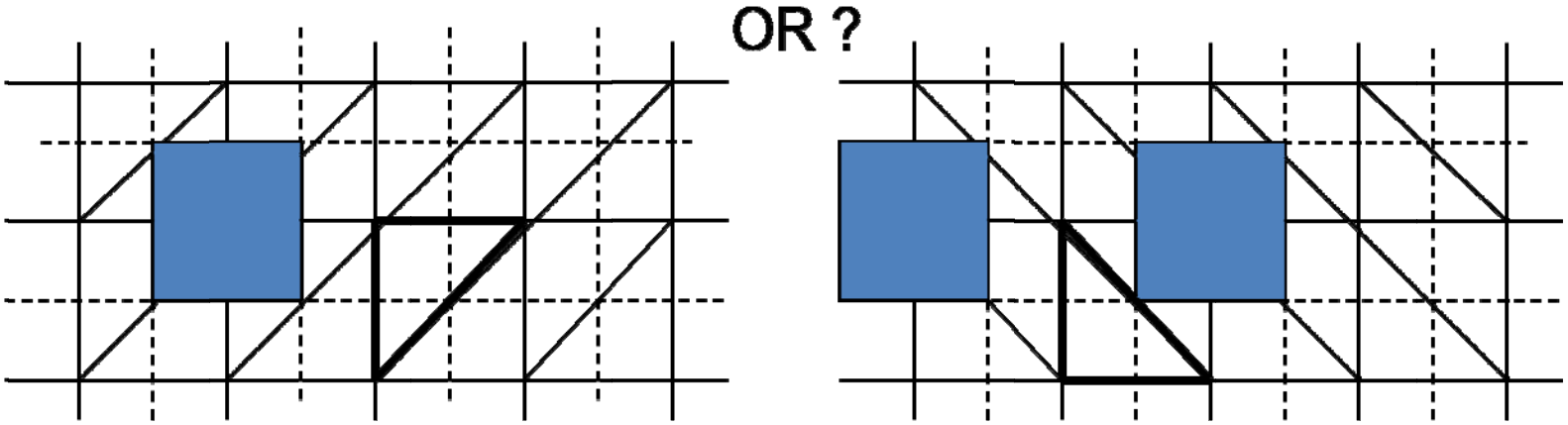
Typical problem (good for students analysis)

```
double x = -1; double p =  ….;
while ( x < +1)
{    if (x == p) Console.Out.WriteLine(” *** ”)
     x += p;
}
/*     if p = 0.1 then no output */
/*     if p = 0.25 then expected output */
```

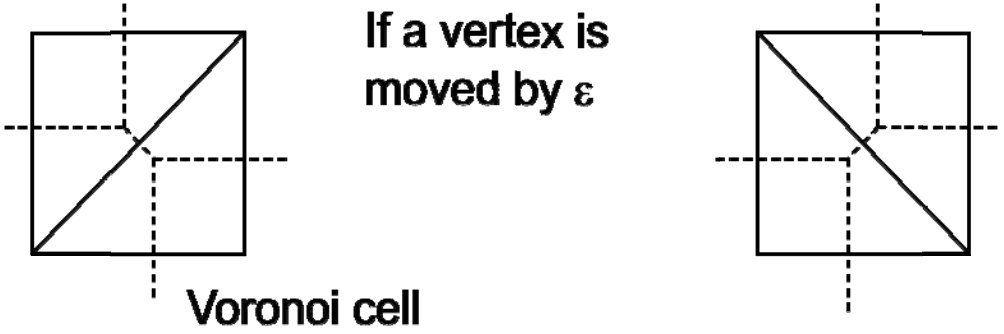
Meshless Interpolations

Delaunay triangulation & Voronoi diagram

Point inside of a circle given by three points – problems with meshing points in a regular rectangular (squared) grid.



It can be seen that the DT & VD is very sensitive to a point position change



?? ROBUSTNESS ??

Why DT is popular? it has mathematically well defined properties

Meshless Interpolations

Floating point

- Not all numbers are represented correctly
- Logarithmic arithmetic
- Continuous fractions
- Interval arithmetic
- etc.

$$\pi = \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{\ldots}}}}$$

$$\pi = [3; 7,15,1,292,1,1,1,2,1,3,1 \ldots]$$

x + y = [a + c, b + d] x = [a , b]

x - y = [a - d, b - c] y = [c , d]

x × y = [min(ac, ad, bc, bd), max(ac, ad, bc, bd)]

x / y = [min(a/c, a/d, b/c, b/d),

max(a/c, a/d, b/c, b/d)] if $y \neq 0$

Use of high precision arithmetic in SW leads to extremely slow computations.

Meshless Interpolations

Numerically NOT valid identities due to limited precision

Typically based on addition of high and small value, so the last bits of mantissa of the smaller one has got lost.

Typical examples:

$$cos^2\alpha + cos^2\beta = 1$$

$$\mathrm{x}^2 - \mathrm{y}^2 = (x - y)(x + y)$$

if $|x| \ll |y|$ or vice versa - $(x - y)(x + y)$ is to be used instead.

Be careful – compiler's optimization can change the order of computation – it mostly optimizes according to the speed not according to the precision of computation.

Meshless Interpolations

Statements like

if <float> = <float> **then ….** or **if** <float> ≠ <float> **then ….**

should not be allowed in programming languages

Quadratic equation - more reliable results

$at^2 + bt + c = 0$ usually solved as $t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

If $b^2 \gg 4ac$ then

$q = -(b + sign(b)\sqrt{b^2 - 4ac}\,)/2$ $t_1 = {}^{q}/_{a}$ $t_2 = {}^{c}/_{a}$

The discriminate should be computed with a twice precision due to x^2 and $\sqrt{x}$ operations.

Vieta's formula $t_1 + t_2 = -{}^{b}/_{a}$ $t_1 t_2 = {}^{c}/_{a}$

Meshless Interpolations

Function value computation at $x = 77617,\ y = 33096$

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

$f = 6.33835\ 10^{29}$ single precision

$f = 1{,}1726039400532$ double precision

$f = 1{,}172603940053178631858834904520183\mathbf{8}$ extended precision

The correct result is "somewhere" in the interval of

$$[-0{,}827396059946821368141165095479816\mathbf{29005},$$
$$-0{,}827396059946821368141165095479816\mathbf{291986}]$$

Exact solution

$$f(x,y) = -2 + \frac{x}{2y} = \frac{54767}{66192}$$

Meshless Interpolations

Summation is one of often used computations

$$\sum_{i=1}^{10^3} 10^{-3} = 0.999990701675415 \qquad \sum_{i=1}^{10^4} 10^{-4} = 1.000053524971008$$

The result should be only one in both cases.

The correctness in summation is very important in power series computations.

!!!! ORDER of summation

$$\sum_{n=1}^{10^6} \frac{1}{n} = 14.3\mathbf{57357} \qquad \sum_{n=10^6}^{1} \frac{1}{n} = 14.3\mathbf{92651}$$

Meshless Interpolations

Recursion

Towers of Hanoi

```
MOVE (A, C, n);
{    MOVE (A, B, n-1);
     MOVE (A, C, 1);
     MOVE (B, C, n-1)
}  # MOVE (from, to, number) #
```

Ackermann function

$$A(m,n) = \begin{cases} n+1 & if\ m = 0 \\ A(m-1,1) & if\ M > 0\ and\ n = 0 \\ A\big(m-1, A(m,n-1)\big) & if\ m > 0\ and\ N > 0 \end{cases}$$

The value of the function grows very fast as

$$A(4,4) = 2^{2^{2^{65536}}} = 2^{2^{10^{197296}}}$$

Meshless Interpolations

Mathematical "forms" **There are several "forms":**

Implicit $F(x,y,z)=0$ or $F(\boldsymbol{x})=0$ or $\boldsymbol{F}(\boldsymbol{x})=\boldsymbol{0}$ (system of equations)

There is no orientation, e.g.

- if $F(\boldsymbol{x})=0$ is a iso-curve there is no hint how to find another point of this curve, resp. a line segment approximating the curve => tracing algorithms
- if $F(\boldsymbol{x})=0$ is a iso-surface there is no hint how to find another point of this surface => iso-surface extraction algorithms

Parametrical $\boldsymbol{x}=\boldsymbol{x}(u)$ $\boldsymbol{x}=\boldsymbol{x}(u,v)$

Points of a curve are "ORDERED" according to a parameter u, resp. u,v

Explicit $z=f(x)$ $z=f(x,y)$ [actually 2 ½ D]

For the given value x, resp. x,y we get function value z

Meshless Interpolations

Implicit form

- can be used for separation - for detection if a point is inside or outside, e.g. a half-plane or a circle etc. – reliable, but ….
- There is always a question how to compute $\boldsymbol{x}$ of $F(\boldsymbol{x}) = 0$ as there are several solutions in general, i.e. solution of an equation
- complexity of computations × precision of computation issues

Compiler optimization is **DANGEROUS** in general can change the order of operations – numerical precision

$$x^2 - y^2 = (x + y)(x - y)$$

$$\begin{vmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ C_x & C_y & C_x^2 + C_y^2 & 1 \\ D_x & D_y & D_x^2 + D_y^2 & 1 \end{vmatrix} = \begin{vmatrix} A_x - D_x & A_y - D_y & (A_x^2 - D_x^2) + (A_y^2 - D_x^2) \\ B_x - D_x & B_y - D_y & (B_x^2 - D_x^2) + (B_y^2 - D_x^2) \\ C_x - D_x & C_y - D_y & (C_x^2 - D_x^2) + (C_y^2 - D_x^2) \end{vmatrix} > 0$$

Meshless Interpolations

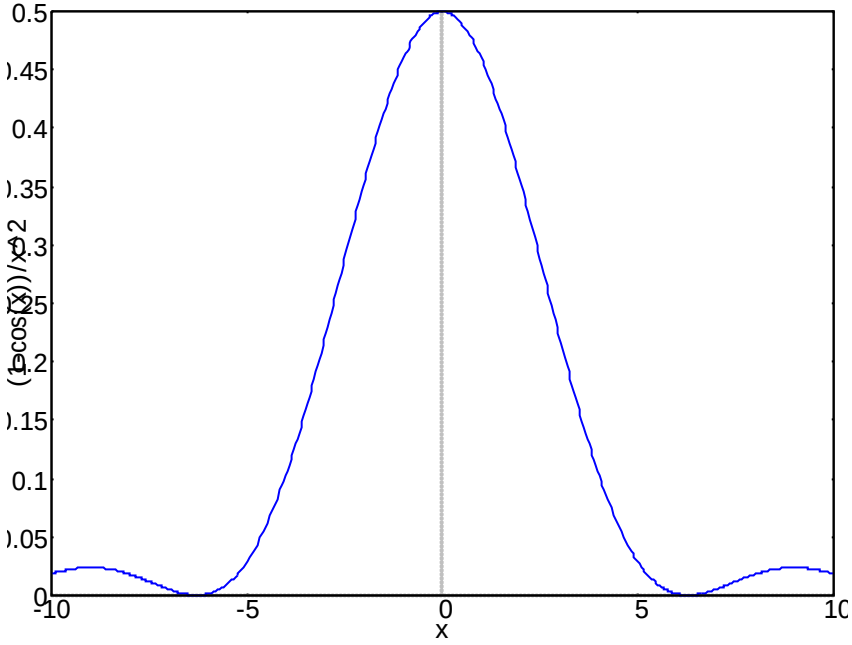
Another example

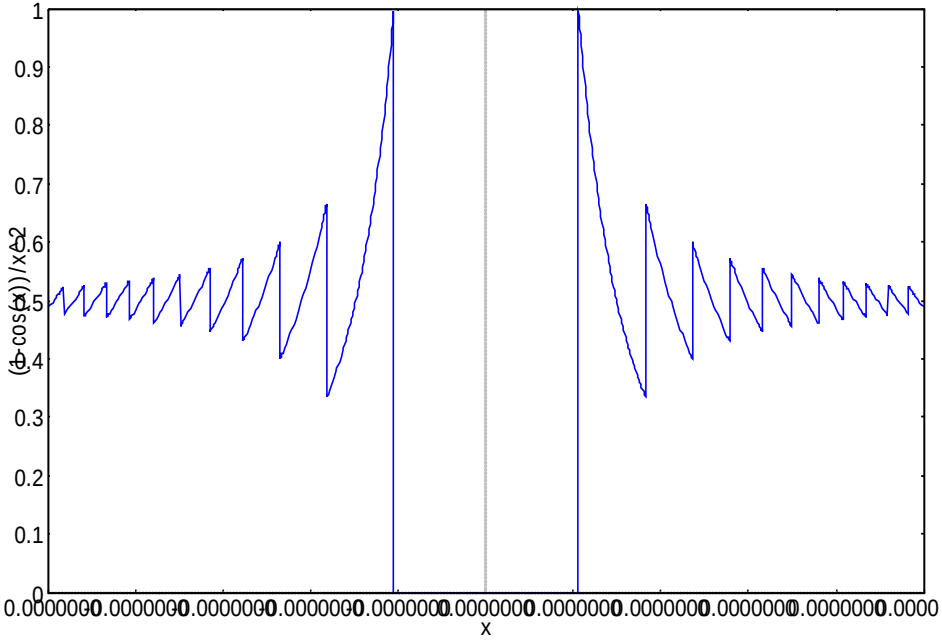
$$f(x) = \frac{1 - \cos x}{x^2}$$

$f(0) = 0{,}5$ - can be shown,
but how to compute it correctly?

Computed **values are wrong** in an interval close to zero!!!!!

In the interval $(-\varepsilon, \varepsilon)$ the function values are **ZERO** instead of 0.5!!!





$$f(x) = \frac{1 - \cos x}{x^2}$$

Interval $< -10^{-8}, 10^{-8} >$

Meshless Interpolations

Examples – what happened?

There are famous examples of numerical disasters.

When reading the original reports and followed comments and details one must be really surprised how simple errors occur and should be worried what could happen in complex problems solution.

Let us shortly explore some “traditional” cases.

Meshless Interpolations

Explosion of Ariane 5

Courtesy CNN

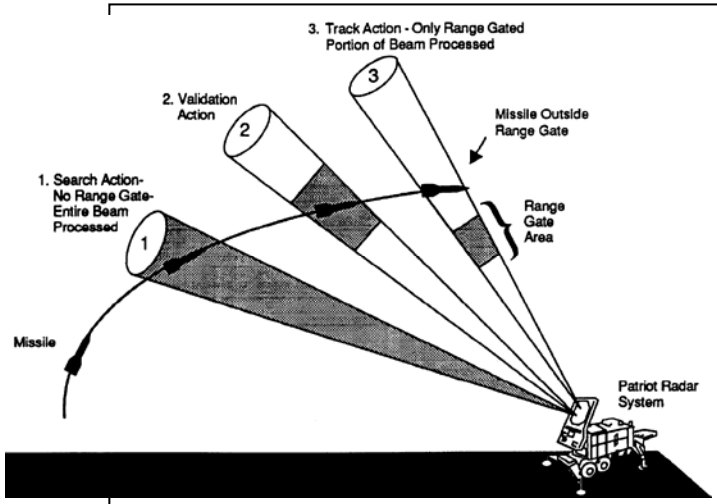
An Ariane 5 rocket was launched by the European Space Agency (ESA) on June 4, 1996. The development cost over $7 billion. The rocket exploded after lift-off in about 40 sec. Destroyed rocket and cargo were valued at $500 million. The cause of a failure was a software error in the inertial reference system. From the CNN article:

"*The internal SRI [Inertial Reference System] software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer.*"

The ***conversion from the floating point*** *to the integer* representation is very dangerous as it ***is not reported by an exception*** and stored value represents an existing number.

Meshless Interpolations

Patriot Missile Failure



Courtesy of GAO report

The system was originally designed in mid-1960 for a short and flexible operation (the system was actually running for more than 100 hours), for intercepting cruise missiles running at MACH 2 speed. But it was used to intercept the Scud missile running at MACH 5. The computation of intercepting and hitting was based on time counting with 24 bits integers with the clock of 1/10 [s] and computation in floats. The clock setting to 1/10 was a critical issue and not acceptable even for application in sport activities at that time. Unfortunately $1/10 = 1/2^4+1/2^5+1/2^8+1/2^9+1/2^{12}+....$ and therefore the error on 24 bits is about 0.000000095 and in 100 hours the error is 0.34. As the Scud flies at MACH 5, the error was actually 687[m] and the missile was out of the "range gate" area.

Meshless Interpolations

As a result of the fault assumptions, incorrect software design and irresponsible attitude of the army officials (not updated software even already available), 28 Americans were killed and over 100 other people injured in the Iraq's Scud missile attack in Dhahran, Saudi Arabia on February 25, 1991 according to the GAO report.

Meshless Interpolations

Sleipner offshore platform sinking

Another well known example is the Sleipner offshore platform sinking. The top deck is about 57 000 tons, drilling and support equipments weight about 40 000 tons and the deck provides an accommodation for about 200 people.

Courtesy of SINTEF

The Sleipner platform structure was "optimized" using finite element system and the shear stresses were underestimated nearly by 50%. It led to serious cracks in the structure and leakage that the pumps were unable to cope with. The sinking of the platform estimated cost is about $700 million.

Meshless Interpolations

We have presented some basic facts on numerical precision and examples of some disasters. Many engineering problems are somehow connected with geometry and geometrical computations with respecting physical phenomena etc.

The majority of computations are made in the Euclidean space representation and with the Cartesian coordinate system.

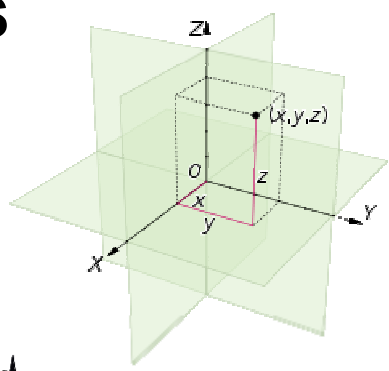
In the following we will show how

- the non-Euclidean representation, actually the projective extension of the Euclidean representation, and
- the principle of duality can be used to solve some problems in a simple, robust and elegant ways.

Meshless Interpolations

Coordinate Systems, Duality and Transformations

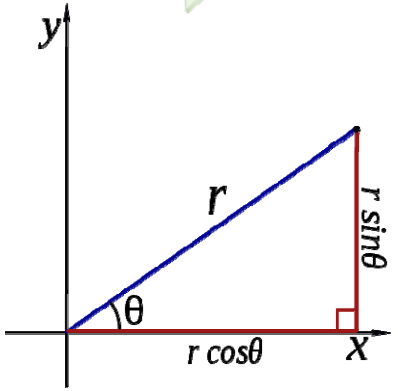
Meshless Interpolations

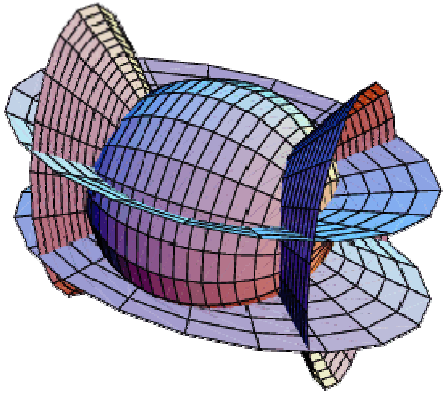


Coordinate systems:

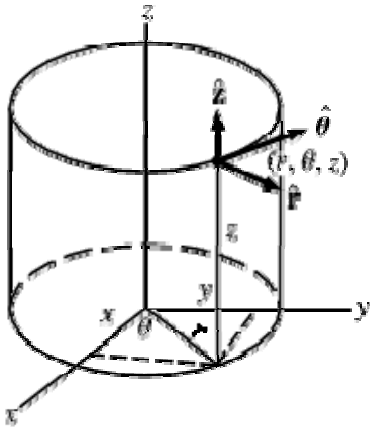
- Cartesian – left / right handed
  the right handed system is used usually
- Polar
- Spherical, Cylindrical

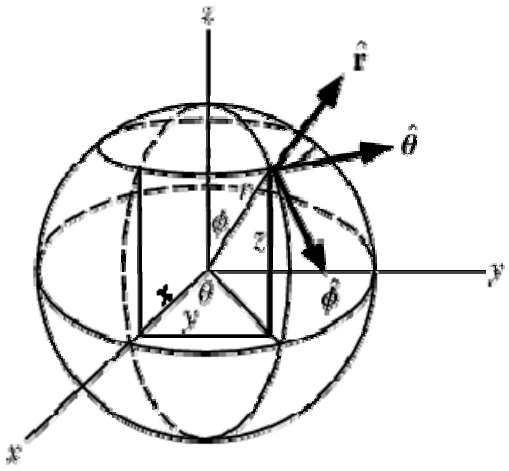
- many others, e.g. Confocal Ellipsoidal Coordinates
  http://mathworld.wolfram.com/ConfocalEllipsoidalCoordinates.html





Courtesy of http://mathworld.wolfram.com/ConfocalEllipsoidalCoordinates.html





Meshless Interpolations

Vectors and Points in Geometry

- **Vectors** – movable, no fixed position
- **Points** – no size, position fixed in the GIVEN coordinate system

- Same data structure is used for points and vectors representation in a memory
- Geometric transformations of points and lines, resp. planes are **DIFFERENT** in general

Meshless Interpolations

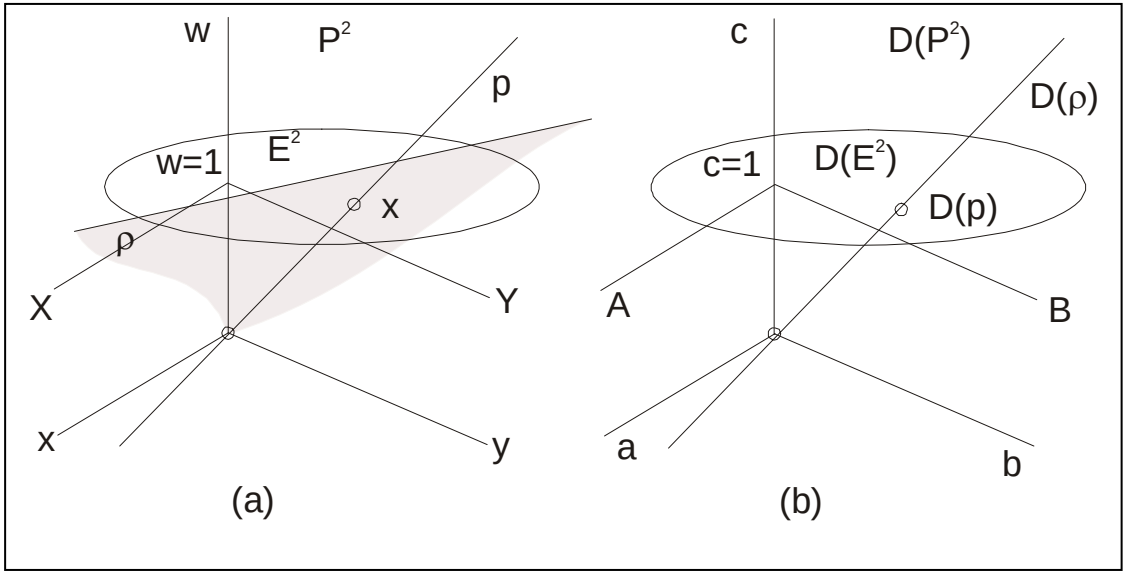
Projective Space

$\boldsymbol{X} = [X, Y]^T \qquad \boldsymbol{X} \in E^2$

$\boldsymbol{x} = [x, y: w]^T \qquad \boldsymbol{x} \in P^2$

Conversion:

$\boldsymbol{X} = [x/w, y/w]^T \qquad w \neq 0$



If $w = 0$ then $\boldsymbol{x}$ represents "an ideal point" [a point in infinity], i.e. it is a directional vector.

The Euclidean space E^2 is represented as a plane $w = 1$.

Equivalent "mathematical" notation often used:

$\boldsymbol{x} = [w: x, y]^T$ generally for P^n $\boldsymbol{x} = [x_0: x_1, \ldots, x_n]^T$

i.e. homogeneous coordinate is the first

Meshless Interpolations

Points and vectors

- **Vectors are "freely movable"** – not having a fixed position

$$\boldsymbol{a}_1 = [x_1, y_1 : 0]^T$$

- **Points** are **not "freely movable"** – they are fixed to the origin of the current coordinate system

$$\boldsymbol{x}_1 = [x_1, y_1 : w_1]^T \quad \text{and} \quad \boldsymbol{x}_2 = [x_2, y_2 : w_2]^T$$

usually in textbooks $w_1 = w_2 = 1$

Why ":" is used?

- x_i, y_i have physical meaning, e.g. distance in meters [m]
- w_i is just a scaling factor without a physical unit

A **vector** $\boldsymbol{A} = \boldsymbol{X}_2 - \boldsymbol{X}_1$ in the Euclidean coordinate system – **CORRECT**

$$\boldsymbol{A} = \left[A_x, A_y\right]^T = [X_2, Y_2]^T - [X_1, Y_1]^T$$

Meshless Interpolations

Horrible "construction" !DO NOT USE IT – TOTALLY WRONG!

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [x_2 - x_1, y_2 - y_1 : w_2 - w_1]^T$$

as $w_1 = w_2 = 1$

$$\boldsymbol{a} = [x_2 - x_1, y_2 - y_1 : 1 - 1]^T = [x_2 - x_1, y_2 - y_1 : 0]^T$$

What happen if $w_1 \neq w_2$ due to a numerical representation?

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [x_2 - x_1, y_2 - y_1 : w_2 - w_1]^T = [a_x, a_y : \varepsilon]^T$$

Now $\varepsilon \neq 0$

This is considered to be interpreted as a point !!!

This was presented as "How a vector" is constructed in the projective space P^k in a textbook!! **WRONG, WRONG, WRONG**

This construction has been found in SW!!

Different data types **point** and **vector** are needed with relevant operations.

Meshless Interpolations

A **Euclidean vector** $\boldsymbol{A}$ given by two points expressed in

- the Euclidean coordinates $\boldsymbol{A} = \left[\frac{w_1 x_2 - w_2 x_1}{w_1\, w_2}, \frac{w_1 y_2 - w_2 y_1}{w_1\, w_2}\right]^T$
- expressed in the homogeneous coordinates as a **vector** type

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [w_1 x_2 - w_2 x_1, w_1 y_2 - w_2 y_1 : w_1\, w_2]^T$$
$$\equiv \left[\frac{w_1 x_2 - w_2 x_1}{w_1\, w_2}, \frac{w_1 y_2 - w_2 y_1}{w_1\, w_2} : 0\right]^T$$

So we can see actually two possible representations.

We use the homogeneous coordinate to represent a denominator of a fraction – postponing the division operation

This is the **CORRECT SOLUTION**, but what is the interpretation?

Meshless Interpolations

A "difference" of coordinates of two points is a vector in the mathematical meaning BUT $w_1\ w_2$ is a "scaling" factor actually

Actually the division operation is postponed and not performed immediately. A **vector** in the projective notation

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [w_1x_2 - w_2x_1, w_1y_2 - w_2y_1 : w_1\ w_2]^T$$
$$\triangleq \left[\frac{w_1x_2 - w_2x_1}{w_1\ w_2}, \frac{w_1y_2 - w_2y_1}{w_1\ w_2} : 0\right]^T$$

where: $\triangleq$ means projectively equivalent

Thus is just "Euclidean" vector using projective representation

BUT we can define a vector in the projective space as follows

Meshless Interpolations

A vector in the **projective space** is given by coordinates x, y, w as

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [x_2 - x_1, y_2 - y_1 : w_2 - w_1]^T$$

NOTE the difference also for the homogenous coordinates

[=>Linear interpolation with a non-linear monotonic parameterization]

We have to strictly distinguish meaning of one *dimensional array* [*vector*], i.e. if we are working with:

- **points**, i.e. a data structure represent point coordinates, or
- **vectors**, i.e. a data structure represent a vector in the mathematical meaning

VECTORS x POINTS

Meshless Interpolations

Duality (linear)

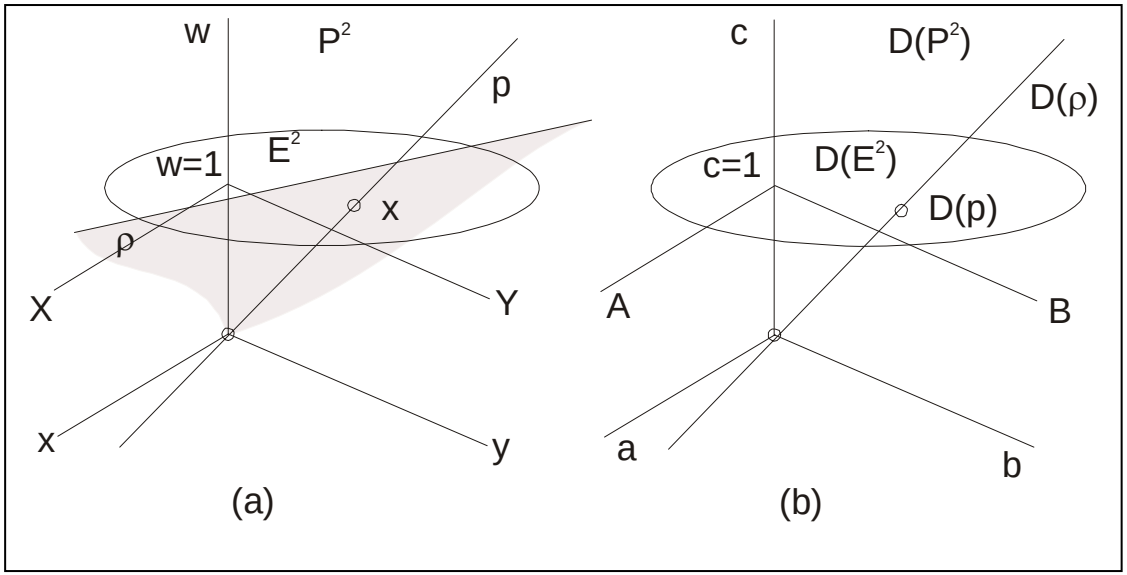
For simplicity, let us consider a line p defined as:

$$aX + bY + c = 0$$

We can multiply it by $w \neq 0$ and we get:

$$awX + bwY + cw = 0$$

$$w \neq 0$$



As $x = wX$ and $y = wY$ we can write:

$ax + by + cw = 0$ i.e. $\boldsymbol{p}^T\boldsymbol{x} = 0$

$\boldsymbol{p} = [a, b: c]^T$ $\boldsymbol{x} = [x, y: w]^T = [wX, wY: w]^T$

A line $p \in E^2$ is actually a plane ρ in the projective space P^2 passing the origin, which is excluded, i.e. the point $\boldsymbol{x} = [0,0: 0]^T$ is excluded

Meshless Interpolations

Duality

From the mathematical notation $\boldsymbol{p}^T\boldsymbol{x} = 0$

we cannot distinguish whether $\boldsymbol{p}$ is a line and $\boldsymbol{x}$ is a point or vice versa in the case of P^2. It means that

- a *point* and a *line* **are dual** in the case of P^2, and
- a *point* and a *plane* **are dual** in the case of P^3.

The principle of duality in P^2 states that:

Any theorem in E^2 remains true when we interchange the words "point" and "line", "lie on" and "pass through", "join" and "intersection", "collinear" and "concurrent" and so on.

Similarly for the E^3 case.

Once the theorem has been established, the dual theorem is obtained as described above.

This helps a lot to solve some geometrical problems.

Meshless Interpolations

Examples of dual objects and operators

	Primitive	Dual primitive
P^2	Point Line	Line Point
P^3	Point Plane	Plane Point

Operator	Dual operator
Join	Intersect
Intersect	Join

Computational sequence for a problem is the same as for the dual problem.

Meshless Interpolations

Definition

The **cross product** of the two vectors

$$\boldsymbol{x}_1 = [x_1, y_1 : w_1]^T \qquad \text{and} \qquad \boldsymbol{x}_2 = [x_2, y_2 : w_2]^T$$

is defined as:

$$\boldsymbol{x}_1 \times \boldsymbol{x}_2 = det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$$

where: $\boldsymbol{i} = [1,0:0]^T \qquad \boldsymbol{j} = [0,1:0]^T \qquad \boldsymbol{k} = [0,0:1]^T$

or as

$$\boldsymbol{x}_1 \times \boldsymbol{x}_2 = \begin{bmatrix} 0 & -w_1 & y_1 \\ w_1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ w_2 \end{bmatrix} = \boldsymbol{T}\boldsymbol{x}_2$$

Please, note that *homogeneous coordinates* are used.

Meshless Interpolations

Intersection of two lines

Let two lines p_1 and p_2 are given by

$$p_1 = [a_1, b_1 : c_1]^T \qquad \text{and} \qquad p_2 = [a_2, b_2 : c_2]^T$$

We have to solve a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$

$$a_1 x + b_1 y + c_1 = 0 \qquad\qquad a_2 x + b_2 y + c_2 = 0$$

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} -c_1 \\ -c_2 \end{bmatrix} *$$

Then well known formula is used

$$x = \frac{Det_x}{Det} = \frac{det \begin{bmatrix} q_1 & b_1 \\ q_2 & b_2 \end{bmatrix}}{det \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}} \qquad y = \frac{Det_y}{Det} = \frac{det \begin{bmatrix} a_1 & q_1 \\ a_2 & q_2 \end{bmatrix}}{det \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}}$$

Usually a sequence like $\boldsymbol{if}\ abs(\det(..)) \leq eps\ \boldsymbol{then}$ is used.

But what if Det is small? What is eps? That is wrong!

From where a programmer knows the value of ?

Meshless Interpolations

Theorem

Let two lines $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ be given. Then the coordinates of an intersection point $\boldsymbol{x}$, which is defined by those two lines, are determined as the cross product of homogeneous coefficients of those lines as

$$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2 \qquad\qquad x = [x, y: w]^T$$

Proof

We are actually looking for a solution to the following equations:

$$\boldsymbol{x}^T \boldsymbol{p}_1 = 0 \qquad\qquad \boldsymbol{x}^T \boldsymbol{p}_2 = 0$$

where: $\boldsymbol{x} = [x, y: w]^T$

Note * usually a line is in its implicit form as $ax + by = q$ instead of $ax + by + c = 0$, or in the explicit form as $= kx + q$.

Meshless Interpolations

A Line given by two points

Given two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and we want to compute a line given by those two points, i.e. we need to compute 3 values a, b, c from two values $\boldsymbol{x}_1$, $\boldsymbol{x}_2$.

⇨ One parametric set of solutions

$$ax_1 + by_1 + c = 0 \qquad\qquad ax_2 + by_2 + c = 0$$

In a matrix form

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad\qquad \boldsymbol{Ax} = \boldsymbol{0}$$

How to solve it?

Select $= 1$? **What happen if a line passing the origin**?

or $= 1$? or $b = 1$ or similarly? **NO, NO, NO!**

BUT HOW?

Meshless Interpolations

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{Ax = 0}$$

Additional condition $a + b = 1$?

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \boldsymbol{Ax = b}$$

Another approach

We know that a line is dual to a point in E^2 and vice versa.

Due to the **duality principle** in E^2:

$\boldsymbol{x = p_1 \times p_2}$ **<= duality =>** $\boldsymbol{p = x_1 \times x_2}$

$\boldsymbol{Ax = b}$ **<= why different? =>** $\boldsymbol{Ax = 0}$

Meshless Interpolations

Theorem

Let two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be given in the projective space. Then the coefficients of the $\boldsymbol{p}$ line, which is defined by those two points, are determined as the cross product of their homogeneous coordinates

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 = [a, b: c]^T$$

Proof

Let the p line be defined in homogeneous coordinates as

$$ax + by + cw = 0$$

We are actually looking for a solution to the following equations:

$$\boldsymbol{p}^T \boldsymbol{x}_1 = 0 \qquad \boldsymbol{p}^T \boldsymbol{x}_2 = 0$$

where: $\boldsymbol{p} = [a, b: c]^T$

Note that c represents a "distance" from the origin of the coordinate system.

Meshless Interpolations

It means that any point $\boldsymbol{x}$ that lies on the p line must satisfy both the equation above and the equation $\boldsymbol{p}^T\boldsymbol{x} = 0$ in other words the $\boldsymbol{p}$ vector is defined as

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 = det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$$

We can write

$$(\boldsymbol{x}_1 \times \boldsymbol{x}_2)^T \boldsymbol{x} = 0$$

$$det \begin{bmatrix} x & y & w \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0$$

Note that the **cross product** and the **dot product** are the instructions in Cg/HLSL on GPU.

Meshless Interpolations

Evaluating the determinant $det \begin{bmatrix} a & b & c \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0$

we get the line coefficients of the line p as:

$$a = det \begin{bmatrix} y_1 & w_1 \\ y_2 & w_2 \end{bmatrix} \qquad b = -det \begin{bmatrix} x_1 & w_1 \\ x_2 & w_2 \end{bmatrix} \qquad c = det \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}$$

Note:

1. A line $ax + by + c = 0$ is a one parametric set of coefficients
$$\boldsymbol{p} = [a, b : c]^T$$
From two values $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ we have to compute 3 values, coefficients a , b and c

2. For $w = 1$ we get the standard cross product formula and the cross product defines the p line, i.e. $\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$ where:
$$\boldsymbol{p} = [a, b : c]^T$$

Meshless Interpolations

DUALITY APPLICATION

In the projective space P^2 points and lines are dual. Due to duality we can directly intersection of two lines as

$$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2 = det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{bmatrix} = [x, y: w]^T$$

If the lines are parallel or close to parallel, the homogeneous coordinate $w \to 0$ and users have to take a decision – so there is no sequence in the code like $\boldsymbol{if}\ abs(\det(..)) \leq eps\ \boldsymbol{then}$ …in the procedure.

Generally computation can continue even if $w \to 0$ if projective space is used.

Meshless Interpolations

Computation in Projective Space

- Extended cross product definition
- A plane ρ is determined as a cross product of three given points

$$\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3 = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{vmatrix}$$

Due to the duality

- An intersection point $\boldsymbol{x}$ of three planes is determined as a cross product of three given planes.

$$\boldsymbol{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3 = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{vmatrix}$$

Computation of generalized cross product is equivalent to a solution of a linear system of equations => no division operation!

Using the cross product we can continue with symbolic operations which could not be made if solution of $\boldsymbol{Ax} = \boldsymbol{b}$ is used.

Meshless Interpolations

We have seen that computation of

- an intersection of two lines is given as $\boldsymbol{Ax} = \boldsymbol{b}$
- a line given by two points is given as $\boldsymbol{Ax} = \boldsymbol{0}$

Different schemes

BUT

Those problems are DUAL.

Why algorithms should be different??

Meshless Interpolations

Cross product is equivalent to a solution of both linear systems of equations, i.e.

$$\boldsymbol{Ax = b}\ \textbf{and}\ \boldsymbol{Ax = 0}\ \textbf{!}$$

No division operations!

Meshless Interpolations

DISTANCE

Geometry is strongly connected with distances and their measurement. Geometry education deals strictly with the Euclidean geometry, where the distance is measured as

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} \quad \text{, resp.} \quad d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}\ .$$

This concept is convenient for a solution of basic geometric problems, but in many cases it results into quite complicated formula.

There are severe questions of stability and robustness in many cases.

The main objection against the projective representation is that there is no metric.

Meshless Interpolations

The distance of two points can be easily computed as

$$dist = \sqrt{\xi^2 + \eta^2}/(w_1 w_2)$$

where: $\xi = w_1 x_2 - w_2 x_1$ $\eta = w_1 y_2 - w_2 y_1$

Also a distance of a point $\boldsymbol{x}_0$ from a line in E^2 can be computed as

$$dist = \frac{\boldsymbol{a}^T \boldsymbol{x}_0}{w_0 \sqrt{a^2 + b^2}}$$

where: $\boldsymbol{x_0} = [x_0, y_0 : w_0]^T$ $\boldsymbol{a} = [a, b : c]^T$

The extension to E^3/P^3 is simple and the distance of a point $\boldsymbol{x}_0$ from a plane in E^3 can be computed as

$$dist = \frac{\boldsymbol{a}^T \boldsymbol{x}_0}{w_0 \sqrt{a^2 + b^2 + c^2}}$$

where: $\boldsymbol{x_0} = [x_0, y_0, z_0 : w_0]^T$ $\boldsymbol{a} = [a, b, c : d]^T$.

Meshless Interpolations

In many cases we do not need actually a *distance*, e.g. for a decision which object is closer, and *distance*^{2} can be used instead, i.e. for the E^2 case

$$dist^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{{w_0}^2(a^2+b^2)} = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{{w_0}^2\,\boldsymbol{n}^T\boldsymbol{n}}$$

where: $\boldsymbol{a} = \lfloor a, b : c \rfloor^T = \lfloor \boldsymbol{n} : c \rfloor^T$ and the normal vector $\boldsymbol{n}$ is not normalized.

If we are comparing distances of points $\boldsymbol{x}_0$ from the given line *p* we can use "*pseudo-distance*" for comparisons

$$(pseudo\_dist)^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{{w_0}^2}$$

Similarly for a plane ρ in the case of E^3

$$dist^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{{w_0}^2(a^2+b^2+c^2)} = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{{w_0}^2\,\boldsymbol{n}^T\boldsymbol{n}} \qquad \text{and} \qquad (pseudo\_dist)^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{{w_0}^2}$$

where: $\boldsymbol{a} = \lfloor a, b, c : d \rfloor^T = \lfloor \boldsymbol{n} : d \rfloor^T$

Meshless Interpolations

Transformation of lines and planes

	E^2	E^3
Dual problem	$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$ $\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2$	$\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3$ $\boldsymbol{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3$

In graphical applications position of points are changed by an interaction, i.e.

$$\boldsymbol{x}' = \boldsymbol{T}\boldsymbol{x}$$

The question is how coefficients of a line, resp. a plane are changed if the points are transformed without a need to be recomputed from the definition.

It can be proved that

$$\boldsymbol{p}' = (\boldsymbol{T}\boldsymbol{x}_1) \times (\boldsymbol{T}\boldsymbol{x}_2) = det(\boldsymbol{T})(\boldsymbol{T}^{-1})^T\boldsymbol{p} \triangleq (\boldsymbol{T}^{-1})^T\boldsymbol{p}$$

or

$$\boldsymbol{\rho}' = (\boldsymbol{T}\boldsymbol{x}_1) \times (\boldsymbol{T}\boldsymbol{x}_2) \times (\boldsymbol{T}\boldsymbol{x}_3) = det(\boldsymbol{T})(\boldsymbol{T}^{-1})^T\boldsymbol{\rho} \triangleq (\boldsymbol{T}^{-1})^T\boldsymbol{\rho}$$

Meshless Interpolations

Transformation of lines and planes

As the computation is made **in the projective space** we can write

$$\boldsymbol{p}' = (\boldsymbol{T}^{-1})^T \boldsymbol{p} = [a', b' : c']^T \qquad \text{for lines in } E^2$$

or

$$\boldsymbol{\rho}' = (\boldsymbol{T}^{-1})^T \boldsymbol{\rho} = [a', b', c' : d']^T \qquad \text{for planes in } E^3$$

THIS SIMPLIFIES COMPUTATIONS

Transformation matrices for lines, resp. for planes are **DIFFERENT** from transformations for points!

Note that a normal vector of a line is actually a co-vector, i.e. an oriented "surface".

Meshless Interpolations

Data Types, Structures and Classification

Meshless Interpolations

Data types

Data type	Dimensionality	Interval of values	Interpolation
Textual	$\rightarrow \infty$	$moh\{ASCII\}$ i.e. $\{0,\ldots,255\}^*$	NO
Geometrical	$d = 2$ or $d = 3$	$(-\infty, \infty)$	YES
Images	$d = 2$ or $d = 3$	$\{0,\ldots,255\}^*$	YES
Signals^{++}	$d = 2$ or $d = 3$	$(-\infty, \infty)$	NO
Numerical^{+}	$d = ??$	$(-\infty, \infty)$	YES

$\{\ \}^*$ usually restricted to 1,2 or 4 Bytes

$^{+}$ float, double... x *real*, *complex*, *quaternion*....

$^{++}$ signal representations – spectral representation

- Interpolation of geometrical data & image data
- Data structures

Meshless Interpolations

Data structures

- Representation of **discrete** entities, i.e. images, CT/MRI data etc.

- Representation of **continuous** entities, i.e. surface of objects

Representation of

- physical entities – scalar [CT/MRI], vector and tensor fields
  representation of physical or other phenomena
  Large data volume – [GB] – [TB]

- geometrical entities – shapes, volumes etc.

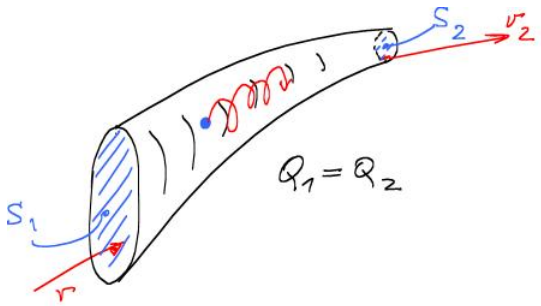
STATIC x **DYMAMIC** data

Dynamic data – **synchronous** x **asynchronous** data [sea sensors...]

Meshless Interpolations

$Q_1 = Q_2$

$S_1 v_1 = S_2 v_2$



$\{\boldsymbol{x}, \boldsymbol{h}\} = \{[x, y, z]^T, [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m]^T\}$

		Associated values $\boldsymbol{h}$	
		Static	Dynamic
Coordinates $\boldsymbol{x}$	Static	$\boldsymbol{\Omega} = \{\boldsymbol{x}, \boldsymbol{h}\}_1^n$	$\boldsymbol{\Omega} = \{\boldsymbol{x}, \boldsymbol{h}(t)\}_1^n$
	Dynamic	$\boldsymbol{\Omega} = \{\boldsymbol{x}(t), \boldsymbol{h}\}_1^n$	$\boldsymbol{\Omega} = \{\boldsymbol{x}(t), \boldsymbol{h}(t)\}_1^n$

Meshless Interpolations

Data structures - classification

- unordered – data without any inner "topological" structure
  - scattered – data scattered with or without known distribution
  - clustered – data having some geometric clustering in space
- ordered
  - un-structured – irregular triangular or tetrahedronal meshes [crash tests, mechanical properties computations....]
  - structured – typically orthogonal meshes – exact mapping of neighbours vertices, cells etc.
    - irregular – cells have different shapes - rectangles can be of different shapes
    - regular – cells have the same shape - all rectangles are same

Structures

- plain – no hierarchy etc.
- hierarchical, adaptive etc., e.g. quad tree, octree, ...,

Meshless Interpolations

In data visualization data are tessellated to cells and the given values (scalar or d-dimensional) are associated with cell's vertices.

Basic cells

- line segment
- triangle
- quadrilateral – planar or non-planar
- tetrahedron
- pyramid
- parallelpiped, hexahedron
- prism

Meshless Interpolations

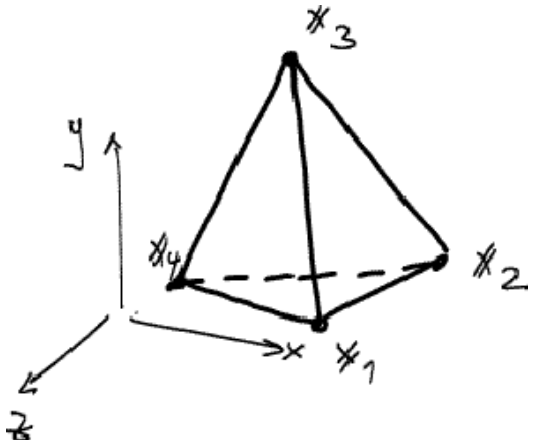
Let us consider a simple case in E – tetrahedron

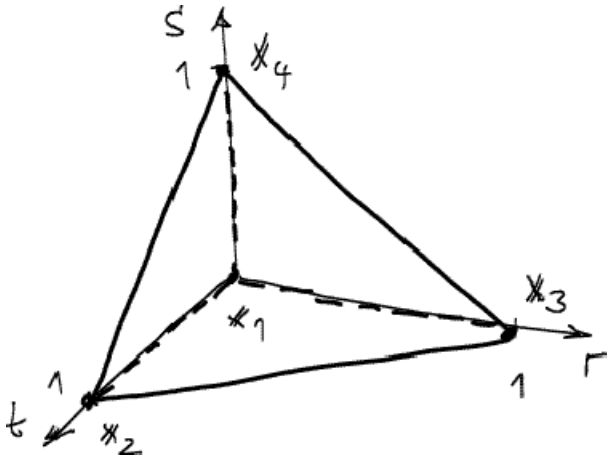
General interpolation formula: (Φ_i is an interpolation function)

$$\boldsymbol{x} = (x, y, z) = \boldsymbol{T}(r, s, t) = \sum_{i=1}^{n} \boldsymbol{x}_i \Phi_i (r, s, t) \quad \boldsymbol{x} \in \text{cell}\,(\boldsymbol{x}_i) \qquad \sum_{i=1}^{n} \Phi_i (r, s, t) = 1$$

Values are given in the vertices. Parametrization

$$\boldsymbol{x}_1 = (0,0,0) \qquad \boldsymbol{x}_2 = (1,0,0) \qquad \boldsymbol{x}_3 = (0,1,0) \qquad \boldsymbol{x}_4 = (0,0,1)$$





Meshless Interpolations

Interpolation functions

$\Phi_1 = 1 - r - s - t \qquad \Phi_2 = r \qquad \Phi_3 = s \qquad \Phi_4 = t$

Inverse transformation

$$(r, s, t) = \boldsymbol{T}^{-1}(\boldsymbol{x}) = \left(\frac{(\boldsymbol{x} - \boldsymbol{x}_1)^T (\boldsymbol{x}_2 - \boldsymbol{x}_1)}{\| \boldsymbol{x}_2 - \boldsymbol{x}_1 \|^2}, \frac{(\boldsymbol{x} - \boldsymbol{x}_1)^T (\boldsymbol{x}_3 - \boldsymbol{x}_1)}{\| \boldsymbol{x}_3 - \boldsymbol{x}_1 \|^2}, \frac{(\boldsymbol{x} - \boldsymbol{x}_1)^T (\boldsymbol{x}_4 - \boldsymbol{x}_1)}{\| \boldsymbol{x}_4 - \boldsymbol{x}_1 \|^2} \right)$$

Inverse transformation does not exist in the implicit form for all cell types -> numerical (iteration) computation -> stability

[hexadron – "deformed cube"]

How to interpolate smoothly in triangular meshes?

Meshless Interpolations

Interpolation of Ordered Data

Meshless Interpolations

Interpolation

Parametric	Explicit	Implicit
$\boldsymbol{x} = \boldsymbol{x}(u)$ $\boldsymbol{x} = \boldsymbol{x}(u,v)$	$z = f(x)$ $z = f(x,y)$	$F(x,z) = 0$ $F(x,y,z) = 0$

- Linear
  - o linear $\boldsymbol{X}(t) = \boldsymbol{X}_0 + (\boldsymbol{X}_1 - \boldsymbol{X}_0)\, t \quad t \in (-\infty, \infty)$
  - o barycentric $\boldsymbol{X}(\lambda_1, \lambda_2) = \lambda_1 \boldsymbol{X}_0 + \lambda_2 \boldsymbol{X}_1$ & $\lambda_1 + \lambda_2 = 1$
  - o spherical $slerp(\boldsymbol{X}_0, \boldsymbol{X}_1, t) = \frac{\sin[(1-t)\Omega]}{\sin\Omega}\boldsymbol{X}_0 + \frac{\sin[t\Omega]}{\sin\Omega}\boldsymbol{X}_1$
- Polynomial
  - o e.g. $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ etc. $\Rightarrow \boldsymbol{Ax} = \boldsymbol{b}$

Meshless Interpolations

Linear interpolation

$$\boldsymbol{X}(t) = \boldsymbol{X}_0 + (\boldsymbol{X}_1 - \boldsymbol{X}_0)\, t \qquad t \in (-\infty, \infty)$$

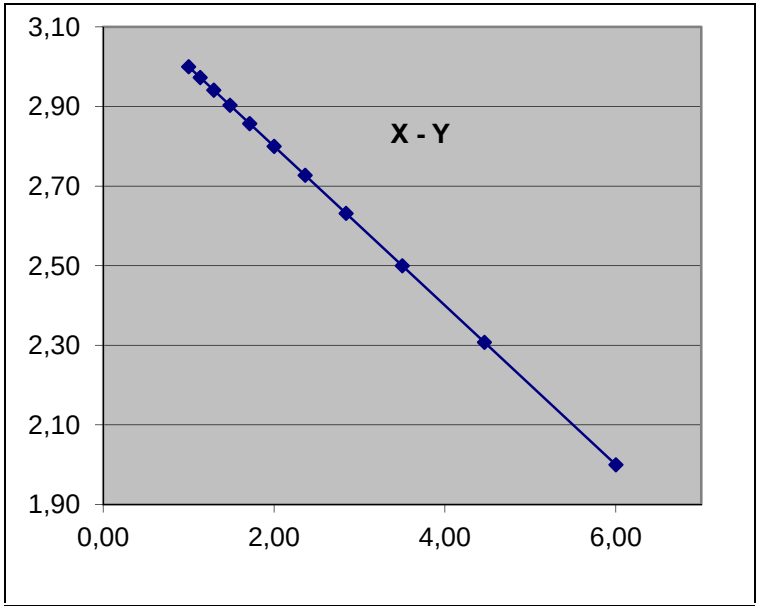
Non-linear monotonous parameterization

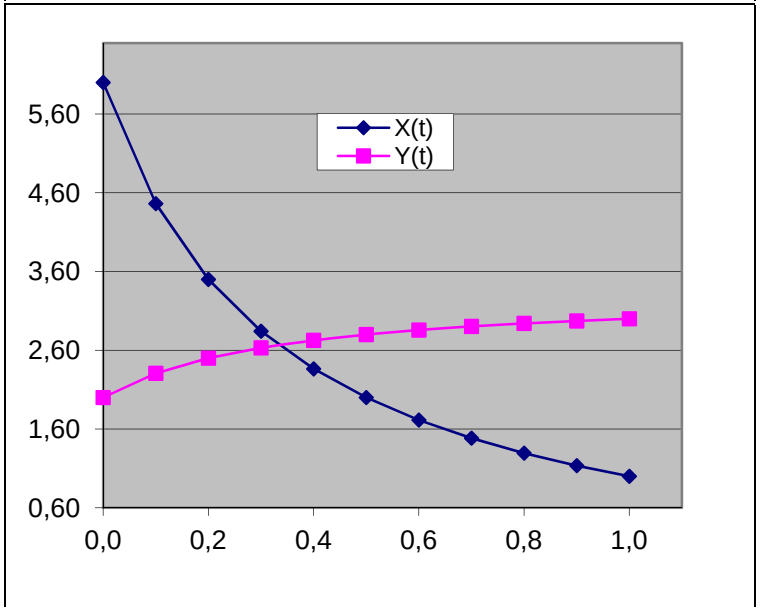
$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + (\boldsymbol{x}_1 - \boldsymbol{x}_0)\, t \qquad t \in (-\infty, \infty)$$
$$x(t) = x_0 + (x_1 - x_0)\, t$$
$$y(t) = y_0 + (y_1 - y_0)\, t$$
$$z(t) = z_0 + (z_1 - z_0)\, t$$
$$w(t) = w_0 + (w_1 - w_0)\, t$$

We can interpolate using homogeneous coordinates without "normalization"!!

- homogeneous coordinate $w \geq 0$

In many algorithms, we need "monotonous" parameterization, only





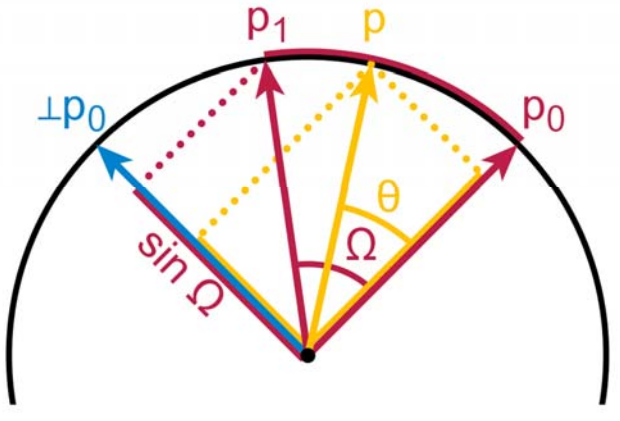
Meshless Interpolations

Spherical interpolation

$$slerp(\boldsymbol{X}_0, \boldsymbol{X}_1, t) = \frac{\sin[(1-t)\Omega]}{\sin\Omega}\boldsymbol{X}_0 + \frac{\sin[t\Omega]}{\sin\Omega}\boldsymbol{X}_1$$

Instability occurs if $\Omega \to k\pi$.

Mathematically formula is correct;
in practice the **code is generally incorrect**! $\left[\frac{0}{0}\right]$



Courtesy of wikipedia

$$slerp(\boldsymbol{X}_0, \boldsymbol{X}_1, t) \triangleq slerp_p(\boldsymbol{X}_0, \boldsymbol{X}_1, t)$$
$$= \begin{bmatrix} \sin[(1-t)\Omega]\boldsymbol{X}_0 + \sin[t\Omega]\boldsymbol{X}_1 \\ \sin\Omega \end{bmatrix}$$
$$\equiv [\sin[(1-t)\Omega]\boldsymbol{X}_0 + \sin[t\Omega]\boldsymbol{X}_1 : \sin\Omega]^T$$

projective scalar used

homogeneous coordinate

Homogeneous coordinates

= > better numerical stability & division operation can be postponed

Meshless Interpolations

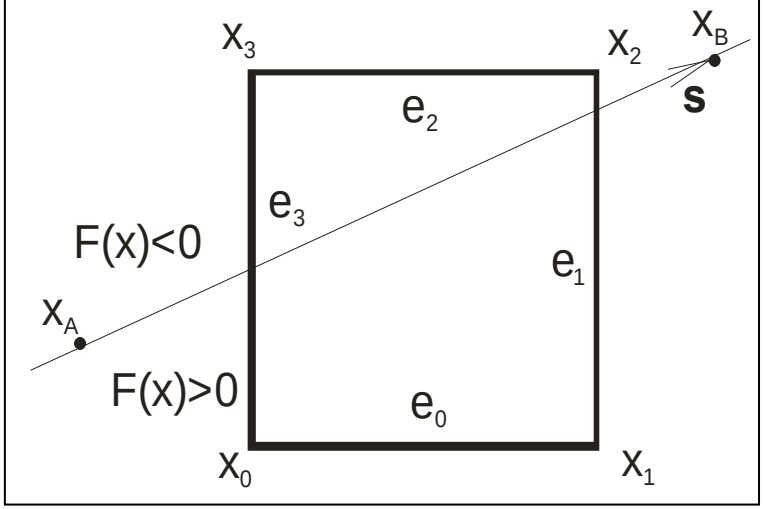
Line – Convex polygon intersection

procedure CLIP_Line ($\boldsymbol{x}_A$, $\boldsymbol{x}_B$);
/* $\boldsymbol{x}_A=[x_A,y_A:w_A]^T$ $\boldsymbol{x}_B=[x_B,y_B:w_B]^T$ */
begin /* $\boldsymbol{p}=[a,b:c]^T$ given - NO STEP 1 */
{1}$\boldsymbol{p} := \boldsymbol{x}_A \times \boldsymbol{x}_B$; /* $\boldsymbol{p}$: ax+by+c = 0 */
{2}**for** k:=0 **to** N-1 **do** /*$\boldsymbol{x}_k=[x_k,y_k,w_k]^T$*/
{3} **if** $\boldsymbol{p}^T\boldsymbol{x}_k \geq 0$ **then** c_k:=1 **else** c_k:=0;
{4}**if** **c**=[0…0]^T **or** **c**=[1…1]^T **then EXIT**;
{5}i:= TAB1[**c**]; j:= TAB2[**c**];
{6}$\boldsymbol{x}_A := \boldsymbol{p} \times \boldsymbol{e}_i$; $\boldsymbol{x}_B := \boldsymbol{p} \times \boldsymbol{e}_j$;
{7}**DRAW** ($\boldsymbol{x}_A$; $\boldsymbol{x}_B$) * $\boldsymbol{e}_i$ – *i-th edge* */
end /* CLIP_Line */
/* **c** identifies an edge intersected */

TOO COMPLEX?

Line clipping algorithms in E^2

- Cohen-Sutherland
- Liang-Barsky
- Hodgman
- Skala – modification of Clip_L for line segments



NO SIMPLE, ROBUST and FAST

- Skala,V.: A new approach to line and line segment clipping in homogeneous coordinates, The Visual Computer, SpringerVol.21, No.11, pp.905-914, 2005

Meshless Interpolations

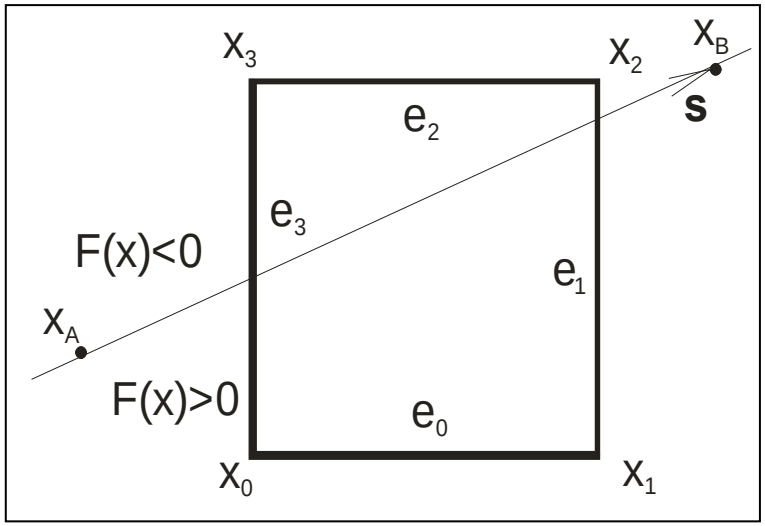
Rectangular normalized window

$\boldsymbol{x}_A := \boldsymbol{p} \times \boldsymbol{e}_i$

For the edge $y = -1$, i.e. $y + 1 = 0$

$$\boldsymbol{x}_A = [x_A, y_A : w_A]^T = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a & b & c \\ 0 & 1 & 1 \end{vmatrix} = [b - c, -a : a]$$

$$\triangleq [\frac{b-c}{a}, \frac{-a}{a} : 1]^T = [\frac{b-c}{a}, -1 : 1]^T$$



Actually expression for y_A , resp. for x_A is given by the window edge.

No multiplication or division operations

A simple modification if a line is given parametrically (in the Euclidean or projective space) as $\boldsymbol{x}(t) = \boldsymbol{x}_A + \boldsymbol{s}t$

Simple modification for non-convex polygon but it requires intersections sorting => $O(M\ lgM)$, where M is a number of intersections.

Meshless Interpolations

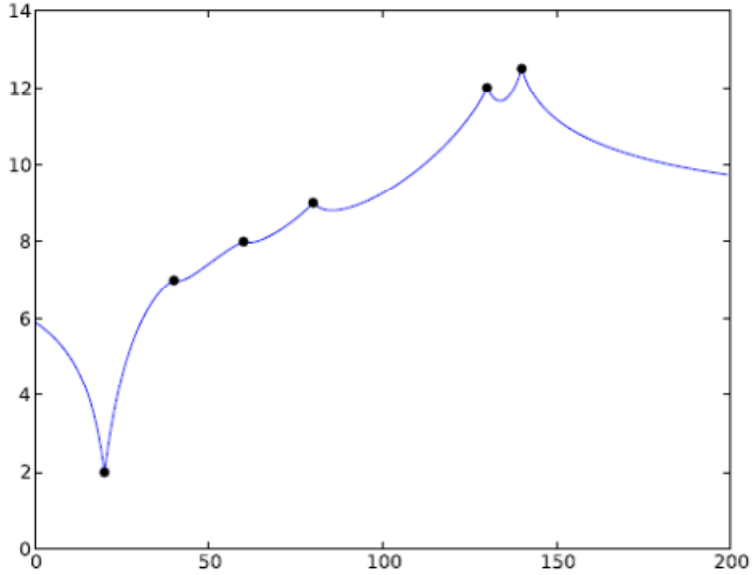
Partition of Unity

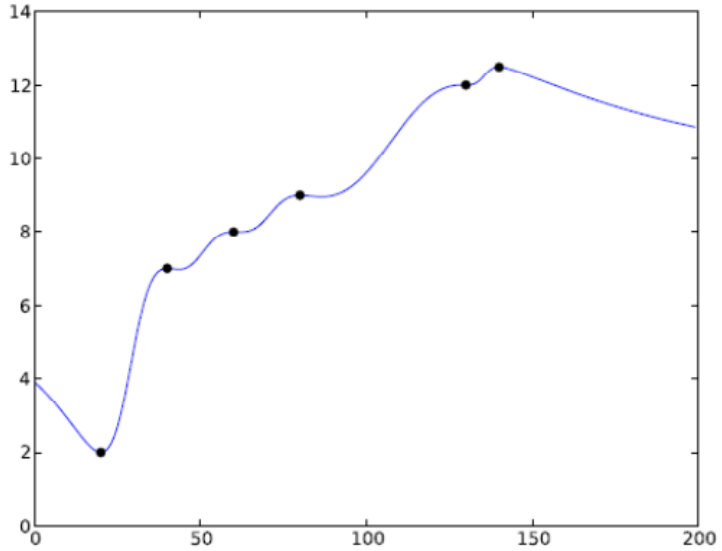
Meshless Interpolations

Shepard's interpolation

$$f(x) = \sum_{k=1}^{N} \frac{w_k(x)}{\sum_{j=1}^{N} w_j(x)} f(x_k)$$

where $w_j(x) = \|x - x_j\|^{-p}$ and $p > 0$





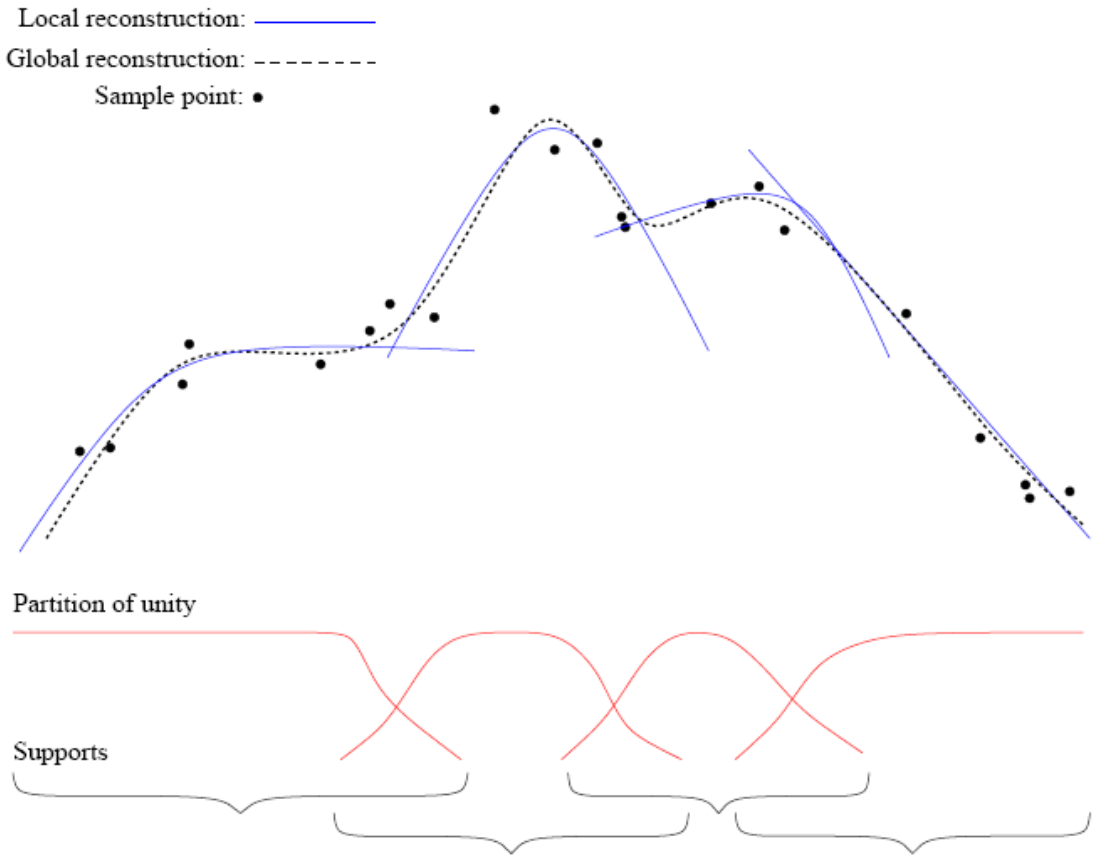
$0 < p \leq 1 \rightarrow$ PEAKS

$p > 1$ smooth, derivative in points is ZERO

Taken from: Ken Anjyo, J. P. Lewis, Frédéric Pighin Scattered data interpolation for computer graphics, SIGGRAPH 2015

Meshless Interpolations

Shepard's method - example of the Partition of Unity, which is actually a blending of local approximations with $\{\phi_k\}$ with weights so that $\sum_k \phi_k = 1$ on the data sub-domain (k-nearest neighbors etc.)



Meshless Interpolations

Interpolation of Unordered Data

Meshless [meshfree] methods

http://meshfree.zcu.cz

Meshless Interpolations

Meshless (meshfree) methods

Meshless interpolations are used for:

- scattered data interpolation, i.e. for interpolation of data sets
  - $\{h_i = f(x_1, \dots, x_d)\}_{i=1}^{N}$
- d −dimensional space in general
- time-varying data, i.e. for spatio-temporal problems in general
- scattered data in time – "not framed" data

Meshless methods:

- offer *smooth interpolation naturally* in the contrary to the methods based on tessellations.
- *do not require a tessellation step*, which has high computational complexity, e.g. Delaunay triangulation is of $O\left(n^{\left\lfloor d/2+1 \right\rfloor}\right)$

Meshless Interpolations

Meshless (or meshfree) techniques are primarily based on the idea of Radial Basis Function (RBF) interpolation [Buhm03], [Wend10], [Wrig03].

However *RBF based methods are not separable*, i.e. interpolation cannot be made along selected axis followed by another along the second axis etc., but easily extensible for the d-dimensional case, in general

Summary

RBF based techniques:

- are easily scalable to the d-dimensional space,
- do not require tessellation of the definition domain
- offers smooth interpolation naturally.
- lead to a solution of a system of linear equations [Duch77], [Hard71] with a full or sparse matrices.

Meshless Interpolations

Two different types of RBF interpolation problems:

- **explicit** ("functional") representation, i.e. $F(\boldsymbol{x}) = h$, e.g. a height map in E^2 – 2 1/2D i.e. . $F(x, y) = h$

- **implicit** (iso-lines, iso-surfaces, e.g. in surface reconstruction), i.e. $F(\boldsymbol{x}) = 0$
  However, there is a severe problem – surface extraction

where: $\boldsymbol{x}$ is a point representation generally in d-dimensional space and h is a scalar value or a vector value.

Meshless Interpolations

RBF Interpolation

The RBF interpolation is based on computing of the distance of two points in the d-dimensional space and is defined by a function:

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{x}_j\right\|\right) = \sum_{j=1}^{M} \lambda_j \, \varphi(r_j) \qquad\qquad r_j = \left\|\boldsymbol{x} - \boldsymbol{x}_j\right\|$$

It means that for the given data set $\{\langle \boldsymbol{x}_i, h_i \rangle\}_1^M$, where h_i are associated values to be interpolated and $\boldsymbol{x}_i$ are domain coordinates, we obtain a linear system of equations:

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|\right) \qquad i = 1, \ldots, M$$

where: λ_j are weights to be computed and e.g. $\varphi(r) = r^2 \lg r$ etc.

Meshless Interpolations

Due to some stability issues, usually a polynomial $P_k(\boldsymbol{x})$ of a degree *k* is added to the formula:

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|\right) + P_k(\boldsymbol{x}_i) \qquad\qquad i = 1, \dots, M$$

For a practical use, the polynomial of the 1st degree is used, i.e. linear polynomial $P_1(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + a_0$, in many applications. So the interpolation function has the form:

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|\right) + \boldsymbol{a}^T\boldsymbol{x}_i + a_0 = \sum_{j=1}^{M} \lambda_j \, \varphi_{i,j} + \boldsymbol{a}^T\boldsymbol{x}_i + a_0$$

$$h_i = f(\boldsymbol{x}_i) \qquad i = 1, \dots, M$$

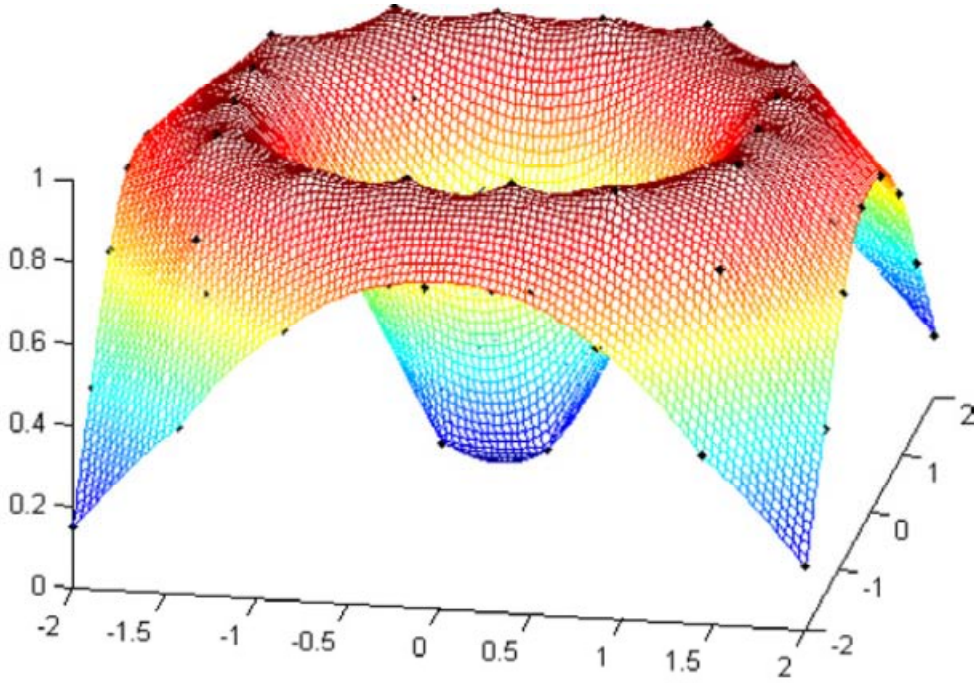
and additional conditions are applied:

$$\sum_{j=1}^{M} \lambda_i = 0 \qquad\qquad \sum_{j=1}^{M} \lambda_i \boldsymbol{x}_i = \boldsymbol{0}$$

Meshless Interpolations

It can be seen that for d-dimensional case a system of $(M + d + 1)$ linear equations has to be solved [$\boldsymbol{Ax} = \boldsymbol{b}$],
where M is a number of points in the dataset and d is the dimensionality of data.

Surface has "elastic" property => there is not a "circle" on the top we would expect



Taken from: Xin Wei, Yi-Zhong Wu, Li-Ping Chen: A new sequential optimal sampling method for radial basis functions, Mathematics and Computation, Volume 218, Issue 19, 1 June 2012, Pages 9635-9646

Meshless Interpolations

Computational complexity of solution linear system of equations is

$O(N^3)$ resp. if iterative $O(kN^2)$

but k is high.

Result is a function $f(\boldsymbol{x})$. It means that

- we can determine interpolated value at any given point **without** tessellation, i.e. without triangulation in E^2 or E^3 ! $O(N^{\left\lfloor d/2+1 \right\rfloor})$
- there is no need to deal with smoothness problems over triangulated meshes in E^2 or E^3 !
  (contour smooth interpolation on a triangular mesh)

Question

"How to decrease number of λ_i" ???

Meshless Interpolations

For $d = 2$ vectors $\boldsymbol{x}_i$ and $\boldsymbol{a}$ are given as $\boldsymbol{x}_i = [x_i, y_i]^T$ and $\boldsymbol{a} = \left[a_x, a_y\right]^T$.
Using a matrix notation we can write for 2-dimensions:

$$\begin{bmatrix} \varphi_{1,1} & \cdots & \varphi_{1,M} & x_1 & y_1 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \varphi_{M,1} & \cdots & \varphi_{M,M} & x_M & y_M & 1 \\ x_1 & \cdots & x_M & 0 & 0 & 0 \\ y_1 & \cdots & y_M & 0 & 0 & 0 \\ 1 & \cdots & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \\ a_x \\ a_y \\ a_0 \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_M \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \qquad \boldsymbol{a}^T \boldsymbol{x_i} + a_0 = a_x\, x_i + a_y\, y_i + a_0$$

It can be seen that for the two-dimensional case and *M* points given a system of $(M + 3)$ linear equations has to be solved.

- If "**global**" functions, e.g. TPS ($\varphi(r) = r^2 lg\, r$), are used the matrix $\boldsymbol{B}$ is "full",
- if "**local**" functions (Compactly supported RBF – CSRBF) are used, the matrix $\boldsymbol{B}$ can be sparse.

Meshless Interpolations

The RBF interpolation was originally introduced by Hardy [Hard71] by introduction of multiquadric method in 1971, which he called Radial Basis Function (RBF) method. Since then many different RFB interpolation schemes have been developed with some specific properties, e.g. Duchon [Duch71] uses

$$\varphi(r) = r^2 lg\, r$$

which is called Thin-Plate Spline (TPS). A function $\varphi(r) = e^{-(\epsilon r)^2}$ was proposed by Wright [Wrig03].

Later Compactly Supported RBF (CSRBF) were introduces as

$$\varphi(r) = \begin{cases} (1-r)^q\, P(r), & 0 \le r \le 1 \\ 0, & r > 1 \end{cases},$$

where: $P(r)$ is a polynomial function and q is a parameter.

Theoretical problems with stability and solvability were solved by Ducon [Duch77].

Meshless Interpolations

Generally, there are two main groups of the RBFs:

- "global" – a typical example is TPS function
- "local" – Compactly supported RBF (CSRBF)

If the "global" functions are taken, the matrix $\boldsymbol{A}$ of the LSE is full and for large M is becoming ill conditioned and problems with convergence can be expected. On the other hand if the CSRBFs are taken, the matrix $\boldsymbol{A}$ is becoming relatively sparse, i.e. computation of the LSE will be faster, but we need to carefully select the scaling factor α and the final function might tend to be "blobby" shaped.

"Global" functions $\phi(r)$			
Thin-Plate Spline (TPS)	$r^2 \lg r$	Multiquadric (MQ)	$\sqrt{1+\varepsilon r^2}$
Gauss function	$e^{-\varepsilon r^2}$	Inverse Quadric (IQ)	$1/\sqrt{1+\varepsilon r^2}$

Table 1 Typical example of "global" functions

Meshless Interpolations

Examples of compactly supported RBF (CSRBF)

ID	CSRBF	ID	CSRBF
1	$(1-r)_+$	6	$(1-r)_+^6(35r^2+18r+3)$
2	$(1-r)_+^3(3r+1)$	7	$(1-r)_+^8$ $(32r^3+25r^2+8r+3)$
3	$(1-r)_+^5$ $(8r^2+5r+1)$	8	$(1-r)_+^3$
4	$(1-r)_+^2$	9	$(1-r)_+^3(5r+1)$
5	$(1-r)_+^4(4r+1)$	10	$(1-r)_+^7(16r^2+7r+1)$

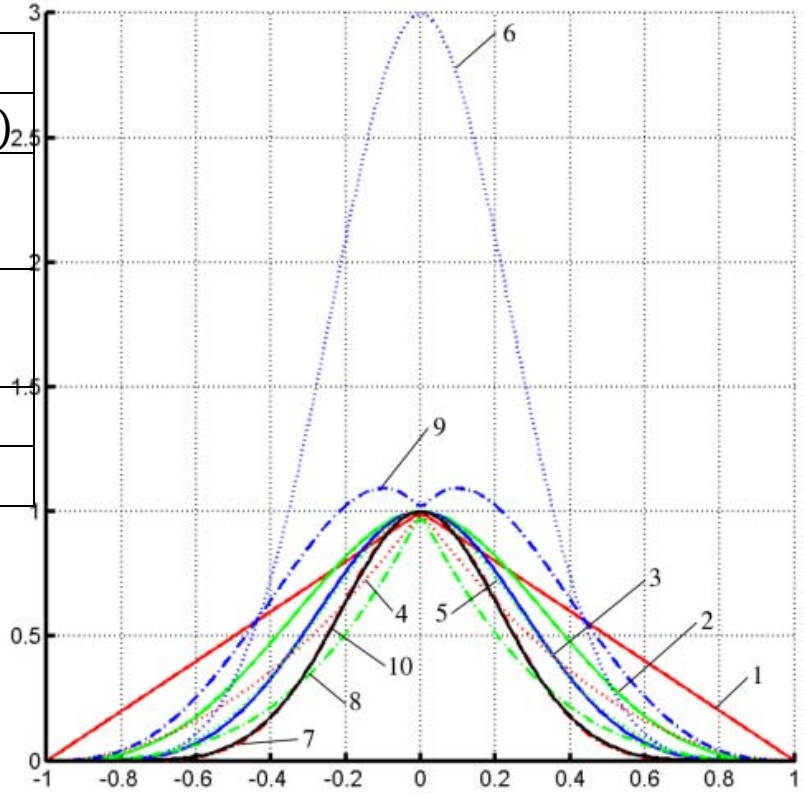


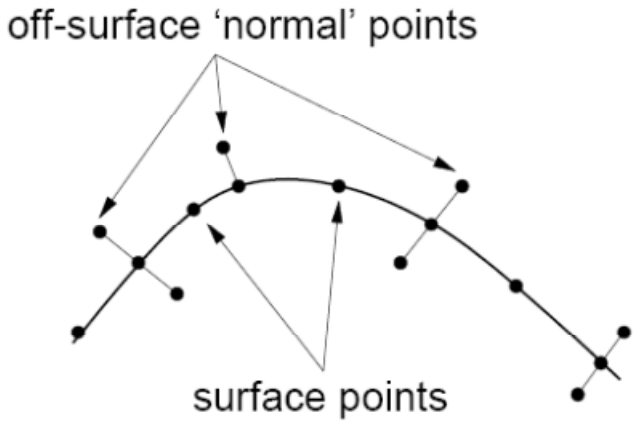
Table 2 "Local" functions - CSRBF

Tab.2 presents typical examples of CSRBFs. They are defined for the interval $\langle 0\,,\,1\,\rangle$ but for the practical use a scaling must be used, i.e. the value r is multiplied by a scaling factor α, where $0 < \alpha < 1$.

Meshless Interpolations

In the case of surface reconstruction from scattered spatial data results is an implicit function $F(\boldsymbol{x}) = 0$. This situation is a little bit more complicated, as the matrix $\boldsymbol{A}$ is generally symmetric, semi-definite or positively definite and the equation $\boldsymbol{Ax} = \boldsymbol{0}$ would have only a trivial solution $\boldsymbol{x} = \boldsymbol{0}$. In this case a surface is considered as an oriented one and additional off-set points are added expecting that a value in those points is δ. Usually additional points are given in the normal vector direction, i.e. $+\boldsymbol{n}$ and $-\boldsymbol{n}$ and matrix size is increased by factor 9, i.e. $3n \times 3n$, where n is a number of the given points [Carr01], [PanR12].

Also as number of points might be very high subdivision techniques are used [Mace11].



Meshless Interpolations

Meshless techniques are primarily based on approaches mentioned above. The resulting matrix $\boldsymbol{A}$ tends to be large and ill-conditioned. Therefore some specific numerical methods have to be taken to increase robustness of a solution, like preconditioning methods or parallel computing on GPU [Naka11] etc. Also subdivision or hierarchical methods are used to decrease sizes of computations and increase robustness [Ohta03], [Suss10].

Meshless interpolation a techniques are used in engineering problem solutions, nowadays, e.g. partial differential equations [Fass07], surface modeling [PanR11], surface reconstruction of scanned objects [Carr01], [Skal13a], reconstruction of corrupted images [Zapl09], etc.

Generally, meshless object's representation is based on specific interpolation techniques [Adam08], [Skal13b], [Skal12]. Detailed description can be found in [Buhm03], [Fass07] and [Wrig03].

Meshless Interpolations

Spatio-temporal data are usually considered as "framed" or "synchronized" in time. The first difficulty is distance computing as distance of two points $\boldsymbol{x}_1 = (x_1, y_1, z_1, t_1)$ and $\boldsymbol{x}_2 = (x_2, y_2, z_2, t_2)$ is usually taken as

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 + \gamma^2 (t_2 - t_1)^2}$$

where $\gamma = 1$ dimensionless. It is incorrect, as we are putting difference in [m] and in [s]. Therefore γ must be of [m/s].

As the scattered spatio-temporal data are naturally scattered in time as well, i.e. they are not "framed", meshless methods enable to solve spatio-temporal not "framed" interpolation, manipulation and representation in a consistent way.

Meshless Interpolations

Approximation – Least Square Error

Meshless Interpolations

In many applications approximation instead of interpolation is needed, typically sampled data approximation with Least Square Error (LSE).

It is necessary to distinguish cases of the approximation

- explicit, i.e. $y = f(x)$ in E^2 or $y = f(\boldsymbol{x})$ in E^d $\quad \boldsymbol{x} = \left[{}^1x, \ldots, {}^{d-1}x \right]^T$
- implicit, i.e. $F(x, y) = 0$ in E^2 or $F(\boldsymbol{x}) = 0$ in E^d

Explicit case - Regression

- linear, i.e. $y = a + bx$, resp. $y = a_0 + a_1 \, {}^1x + a_2 \, {}^2x + \cdots + a_r \, {}^rx$
- quadratic, i.e. $y = a + bx + cx^2$
- polynomial, i.e. $y = a_0 + a_1 x + a_2 x^2 + \cdots + a_r x^r$, e.g. Lagrange interpolation etc.
- hyperbolic, i.e. $y = a + b/x$

Implicit case, i.e. $F(\boldsymbol{x}) = 0$ case, the Orthogonal (Total) Least Square Error should be used.

Meshless Interpolations

Explicit case - Given $\{x_i, y_i\}\ i = 1, \ldots, n$, looking for polynomial $y = f(\boldsymbol{x})$

$$y_i = a_0 + a_1\ {}^1x_i + a_2\ {}^2x_i + \cdots + a_r\ {}^rx_i \qquad\qquad i = 1, \ldots, n, n > r$$

this lead to over determined system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$

Difference between data a interpolation

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax} \quad (\textbf{vector!})$$

The ERROR E is defined as

$$E = \|\boldsymbol{r}\|^2 = \boldsymbol{r}^T\boldsymbol{r} = \|\boldsymbol{b} - \boldsymbol{Ax}\|^2 = (\boldsymbol{b} - \boldsymbol{Ax})^T(\boldsymbol{b} - \boldsymbol{Ax})$$

Minimization

$$\frac{\partial E}{\partial a_k} = \frac{\partial}{\partial a_k}[\boldsymbol{b}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{Ax} - (\boldsymbol{Ax})^T\boldsymbol{b} + \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{Ax}] = 0 \qquad\qquad \forall k = 0, \ldots, r$$

If $\|\boldsymbol{r}\| < 0.05$ then $E < 0.0025$ => converges fast $\to 0$

Meshless Interpolations

Usual solution

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \qquad\qquad \boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$$

!!! => instability – eigenvalues (if $\boldsymbol{A}$ would be regular) $\lambda_k \rightarrow {\lambda_k}^2$ or **SVD** is used.

A similar approach is taken for other "explicit" LSE

However for large datasets, the problem of numerical instability remains.

Note that LSE method is not of the coordinate system origin independent – father points have higher weight, so changing more the space orientation of the approximating function, e.g. of a plane in the case of linear interpolation.

Meshless Interpolations

Orthogonal (Total) Least Square Errors

$$F(x,y)=0 \quad \text{, resp. } F(\boldsymbol{x})=0 \qquad \boldsymbol{x}=\left[\,{}^{1}x,\ldots,{}^{d-1}x\right]^{T}$$

If $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}=\boldsymbol{A}^T\boldsymbol{b}$ would be used we actually get $\boldsymbol{Q}\boldsymbol{x}=\boldsymbol{0}$ as $\boldsymbol{b}=\boldsymbol{0}$

Typical example

Given points $\{x_i,y_i\}$, resp. $\{x_i,y_i,z_i\}$ $i=1,\ldots,n$ and we are looking for a line $ax+by+c=0$ or a plane $ax+by+cz+d=0$ fitting the data with a minimal distance (orthogonal) error, i.e. minimizing the distance (orthogonal) of all the data from a plane.

More complex solution –> eigenvalues and eigenvectors have to be computed in general case.

A simple solution in E^2 is available ->Graphics Gems II

Meshless Interpolations

Meshless approximation of un-ordered multidimensional data sets

Meshless Interpolations

Approximation of Un-ordered Data and Least Square Error

Real life data are noisy. Approximation is to be used instead of interpolation and Tikhonov regularization known in statistics can be applied to RBF.

Then

$$\begin{bmatrix} \boldsymbol{B} + q\boldsymbol{I} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}$$

is used, where $\boldsymbol{I}$ is identical matrix and q is a parameter. If q value is high, function gets smoother, if q is small, the function is closer to pure interpolation.

However the size of the matrix remains and it is large.

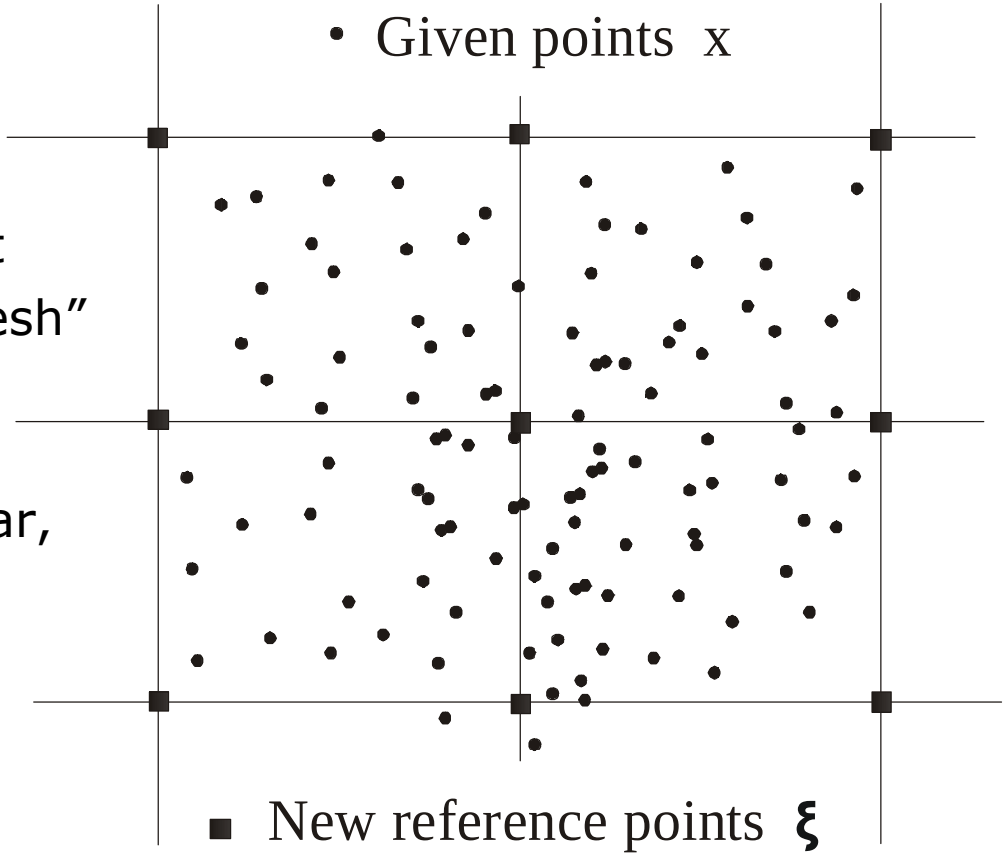
Meshless Interpolations

Let us consider RBF again in the form

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\right\|\right) + \boldsymbol{a}^T \boldsymbol{x}_i + a_0$$

$h_i = f(\boldsymbol{x}_i) \qquad i = 1, \ldots, N$

where: $\boldsymbol{\xi}_j$ are not given points, but points in a pre-defined "virtual mesh" as only coordinates are needed (there is no tessellation needed). This "virtual mesh" can be irregular, orthogonal, regular, adaptive etc.



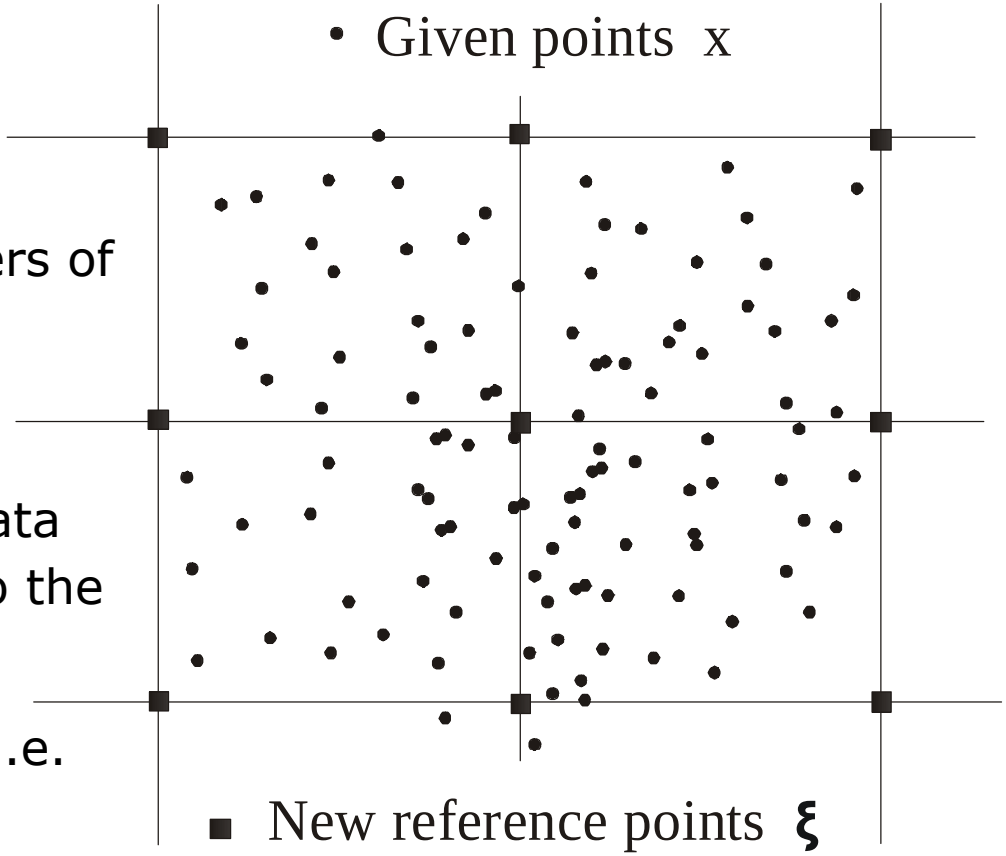
Meshless Interpolations

For simplicity, let us consider 2-dimensional squared (orthogonal) mesh in the following example.

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\right\|\right)$$

$$i = 1, \ldots, N$$

and the $\boldsymbol{\xi}_j$ coordinates are the corners of this virtual mesh.



It means that the given scattered data will be actually "re-sampled", e.g. to the squared mesh.

The question is how to decrease M, i.e. number of λ values and decrease computational cost significantly.

Meshless Interpolations

In many applications the given data sets are heavily over sampled, or for the fast previews, e.g. for the WEB applications, we can afford to "down sample" the given data set. Therefore the question is how to reduce the resulting size of LSE.

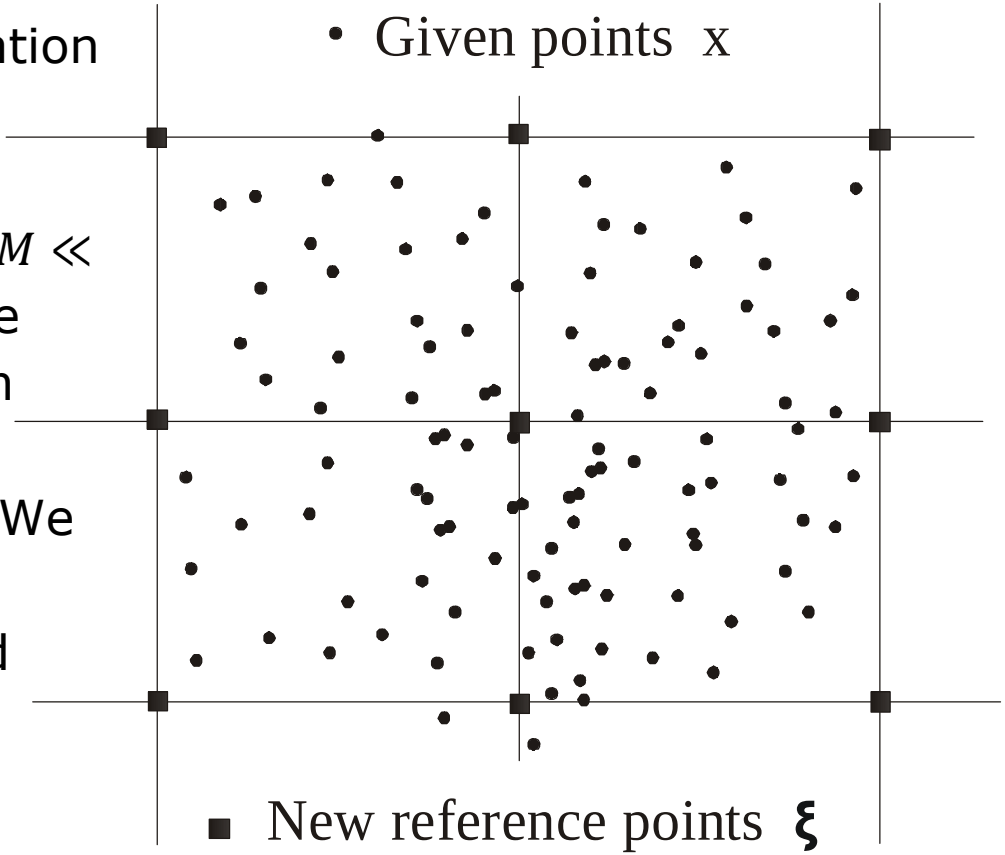
Of course there is a possibility to decrease number of λ's, e.g. by picking the smallest one and remove relevant point and recomputed all.

But note that the solution of a system of linear equations is of $O(N^3)$ in general and usually ill condition systems is solved.

The above presented approach of specifying position of $\boldsymbol{\xi}_j$ points gives us a chance to set "virtual points" at place of our interest with a high density, while in the rest of the space their distribution can be sparse.

Meshless Interpolations

Let us consider that for the visualization purposes we want to represent the final potential field in d-dimensional space by M values instead of N and $M \ll N$. The reason is very simple as if we need to compute the function $f(\boldsymbol{x})$ in many points, the formula above needs to be evaluated many times. We can expect that the number of evaluation Q can be easily requested at $10^2\ N$ of points (new points) used for visualization.



If we consider that $Q \geq 10^2\ N$ and $N \geq 10^2\ M$ then **the speed up factor in RBF function evaluation can be easily about 10^4 !**

This formulation leads to a solution of a linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ where number of rows $N \gg M$, number of unknown $[\lambda_1, \ldots, \lambda_M]^T$.

Meshless Interpolations

This approach reduces the size of the linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ significantly and can be solved by the Least Square Method (LSM) as $\boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{A}^T\boldsymbol{b}$ or Singular Value Decomposition (SVD) can be used.

$$\begin{bmatrix} \varphi_{1,1} & \cdots & \varphi_{1,M} \\ \vdots & \ddots & \vdots \\ \varphi_{i,1} & \cdot\cdot & \varphi_{i,M} \\ \vdots & \ddots & \vdots \\ \varphi_{N,1} & \cdots & \varphi_{N,M} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ \vdots \\ \vdots \\ h_N \end{bmatrix} \qquad \boldsymbol{Ax} = \boldsymbol{b}$$

Meshless Interpolations

The high dimensional scattered data can be approximated by RBF approximation efficiently with a high flexibility as it is possible to add additional points of an area of interest to the virtual mesh and increase precision if needed.

It means that a user can add some points to already given virtual mesh and represent easily some details if requested.

Note that a mesh is not generated!

Meshless Interpolations

However, there are other possibilities, how to decrease computational cost.

If the Compactly Supported RBF (CSRBF) approach is used:

- the matrix $\boldsymbol{A}$ is a sparse matrix
- if data set preprocessed and space subdivision technique is applied, then only the data in a cell and its neighboring cells are used for computation of $\boldsymbol{\lambda}$, which makes the computation much more faster and also stability is increased.

Meshless Interpolations

More general approach

Let us assume again

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|\right) \qquad i = 1, \ldots, N \qquad A\boldsymbol{\lambda} = \boldsymbol{f}$$

where $M \leq N$

We want to determine $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_M]^T$ minimizing quadratic form

$$\frac{1}{2}\boldsymbol{\lambda}^T \boldsymbol{Q} \boldsymbol{\lambda}$$

with a linear constrains $\boldsymbol{A\lambda} - \boldsymbol{f} = \boldsymbol{0}$, where $\boldsymbol{Q}$ is positive and symmetric matrix.

This can be solved using Lagrange multipliers $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^T$, i.e. minimizing

$$\frac{1}{2}\boldsymbol{\lambda}^T \boldsymbol{Q} \boldsymbol{\lambda} - \boldsymbol{\xi}^T (\boldsymbol{A\lambda} - \boldsymbol{f})$$

i.e $\boldsymbol{\lambda}$ =? and $\boldsymbol{\xi}$ =?

Meshless Interpolations

So we are getting as the matrix $\boldsymbol{Q}$ is positive

$$\frac{\partial}{\partial \boldsymbol{\lambda}}\left(\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{Q}\boldsymbol{\lambda} - \boldsymbol{\xi}^T(\boldsymbol{A}\boldsymbol{\lambda} - \boldsymbol{f})\right) = \boldsymbol{Q}\boldsymbol{\lambda} - \boldsymbol{A}^T\boldsymbol{\xi} = \boldsymbol{0}$$

$$\frac{\partial}{\partial \boldsymbol{\xi}}\left(\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{Q}\boldsymbol{\lambda} - \boldsymbol{\xi}^T(\boldsymbol{A}\boldsymbol{\lambda} - \boldsymbol{f})\right) = \boldsymbol{A}^T\boldsymbol{\lambda} - \boldsymbol{f} = \boldsymbol{0}$$

in more compact matrix form

$$\begin{bmatrix} \boldsymbol{Q} & -\boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\xi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{f} \end{bmatrix}$$

As $\boldsymbol{Q}$ is positive definite, block in matrix operations can be applied and we get:

$$\boldsymbol{\lambda} = \boldsymbol{Q}^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{f} \qquad\qquad \boldsymbol{\xi} = (\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{f}$$

If $\boldsymbol{A} = \boldsymbol{A}^T$ and invertible, computation can be simplified.

Meshless Interpolations

Experimental data generation

Meshless Interpolations

Random distribution of points is usually used

Halton points – better distribution

Any non-negative integer n, $0 \leq a_i < p$, $\exists k > 0,\ p$ – prime

$$n = \sum_{i=0}^{k} a_i p^i$$

Function

$$h_p(n) = \sum_{i=0}^{k} \frac{a_i}{p^{i+1}}$$

maps to the interval $\langle 0, 1)$

Sequence generated

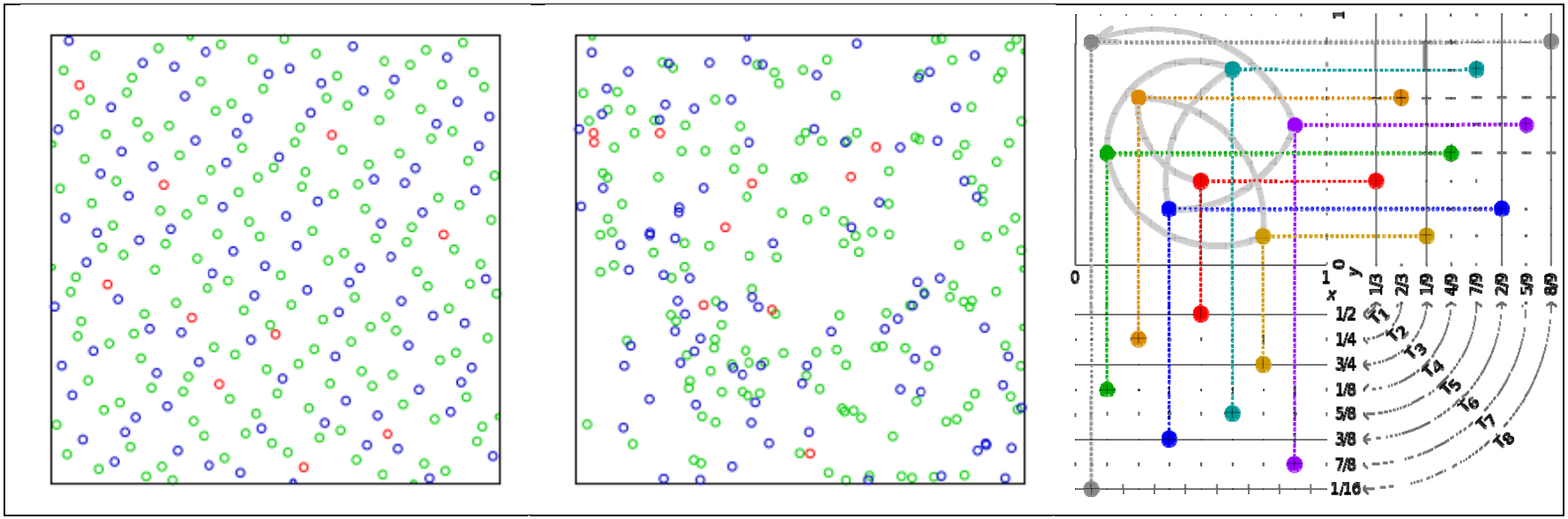
$$h_{p,N} = \{h_p(n) : n = 0,1,2, \ldots, N\}$$

Example:

$$h_3(10) = {}^1/_3 + {}^1/_{3^3} = {}^{10}/_{27}$$

$$h_{3,10} = \left\{0, {}^1/_3, {}^2/_3, {}^1/_9, {}^4/_9, {}^7/_9, {}^2/_9, {}^5/_9, {}^8/_9, {}^1/_{27}, {}^{10}/_{27}, \right\}$$

Meshless Interpolations



Code see - http://en.wikipedia.org/wiki/Halton_sequence

Useful testing functions for multi-dimensional experiments

$$f_d(\boldsymbol{x}) = 4^d \prod_{k=1}^{d} x_k(1-x_k) \qquad \text{where } \boldsymbol{x} = [x_1, \dots, x_d]^T \in [0,1]^d$$

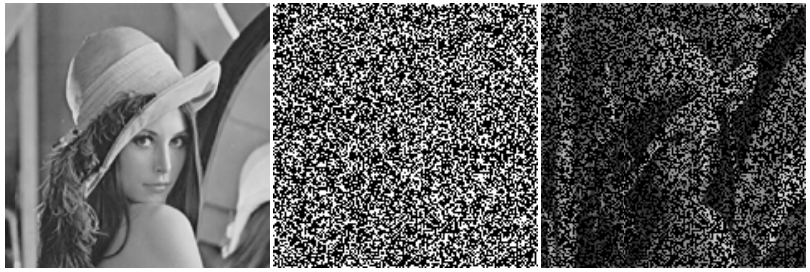
$$\text{sinc}(\boldsymbol{x}) = \prod_{k=1}^{d} \frac{\sin \pi x_k}{\pi x_k} \qquad \text{where } \boldsymbol{x} = [x_1, \dots, x_d]^T \in R^d$$

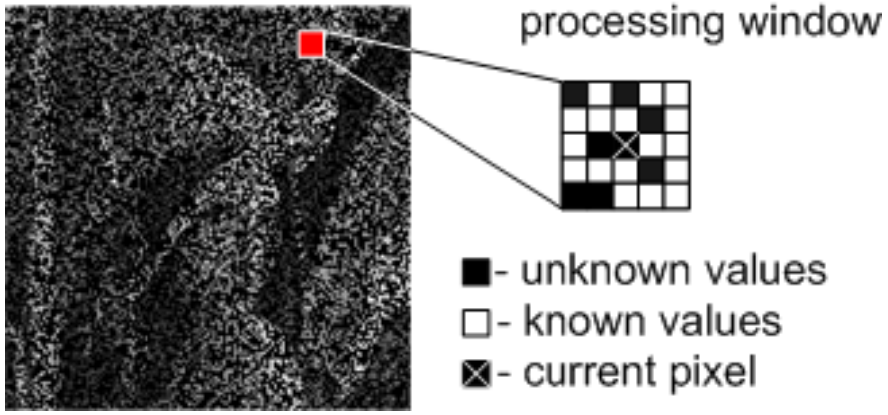
Meshless Interpolations

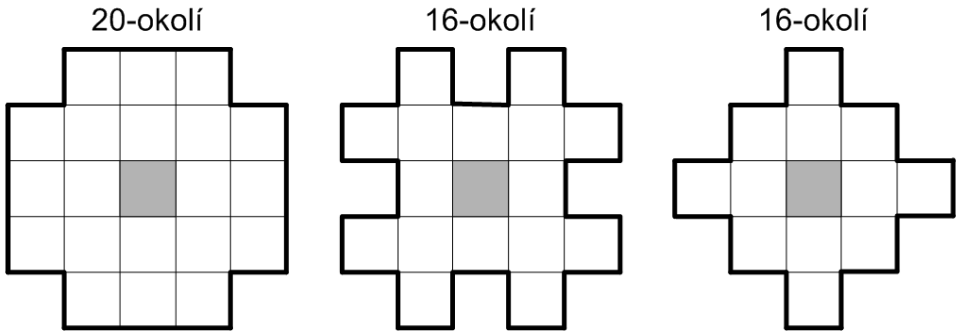
Meshless interpolation and approximation – examples

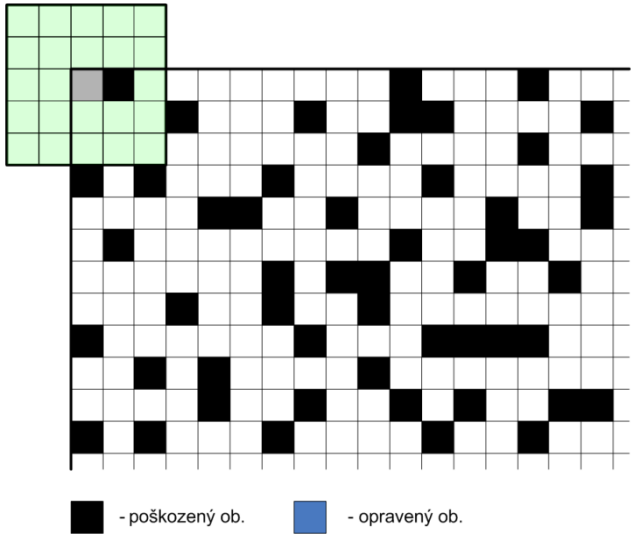
Meshless Interpolations

Image reconstruction





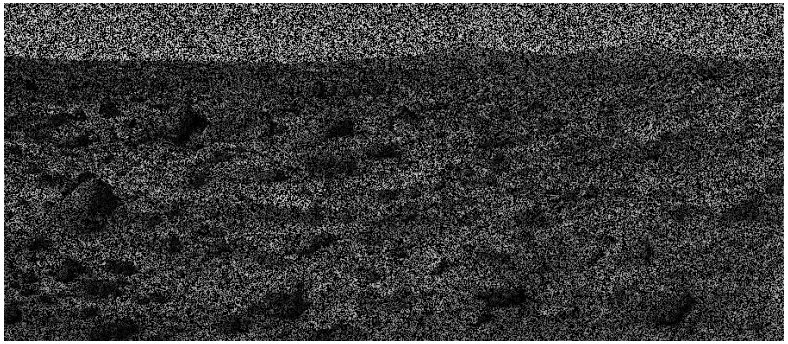




Surface reconstruction

Meshless Interpolations

Image reconstruction



Original - 60% corrupted pixels

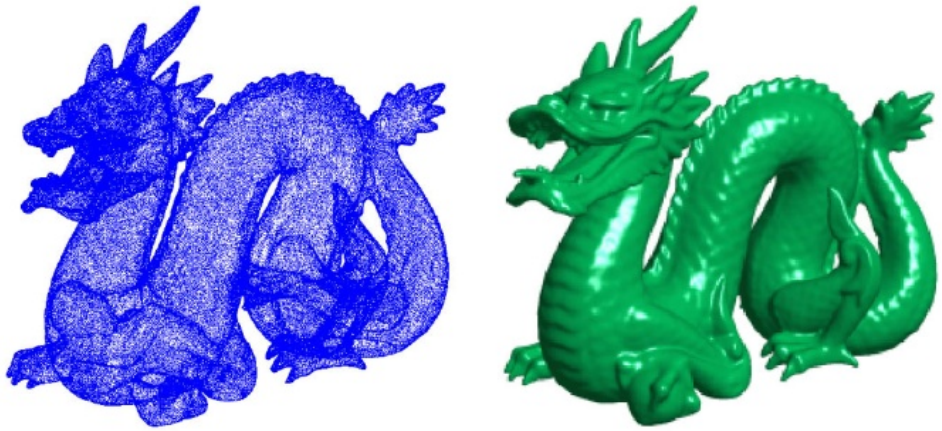
Reconstructed image

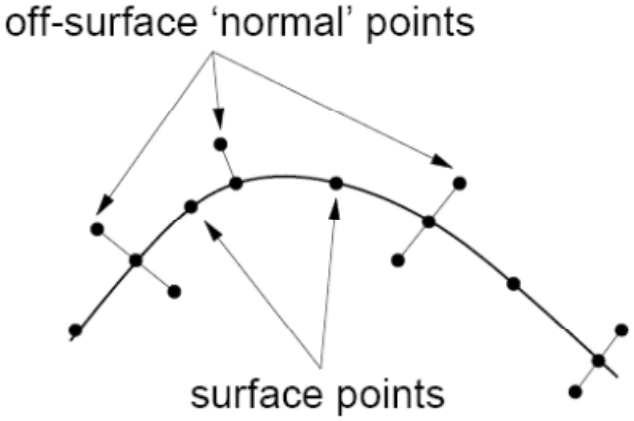
Original image [Bertalmio2000]

Reconstructed [Uhlir&Skala2006]

Meshless Interpolations

Surface reconstruction from scanned data





Surface reconstruction (438 000 points) [Carr et al. 2001]
Implicitly defined problems:

$$F(x, y, z) = 0$$

Problems:

- It leads to $\boldsymbol{Ax} = \boldsymbol{0}$ - trivial solution, only
- additional points with "orientation" have to be artificially included to get $\boldsymbol{Ax} = \boldsymbol{b}$

Meshless Interpolations

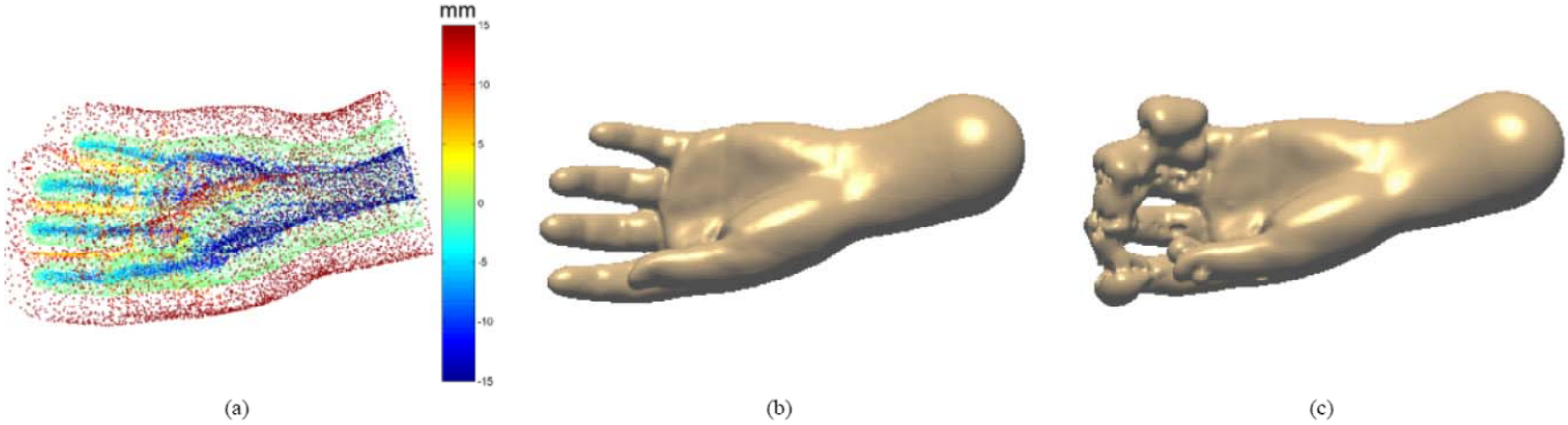


Figure 3: Reconstruction of a hand from a cloud of points with and without validation of normal lengths.

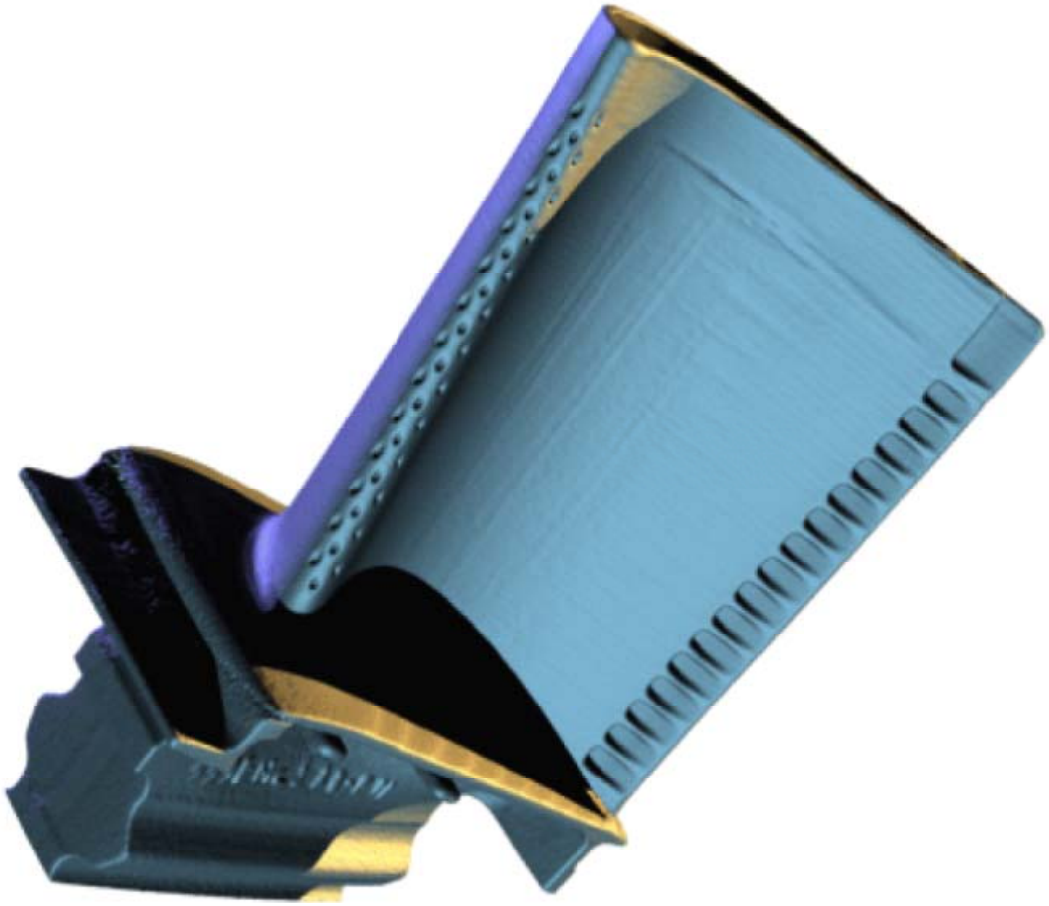
Surface reconstruction (438 000 points) [Carr et al. 2001]

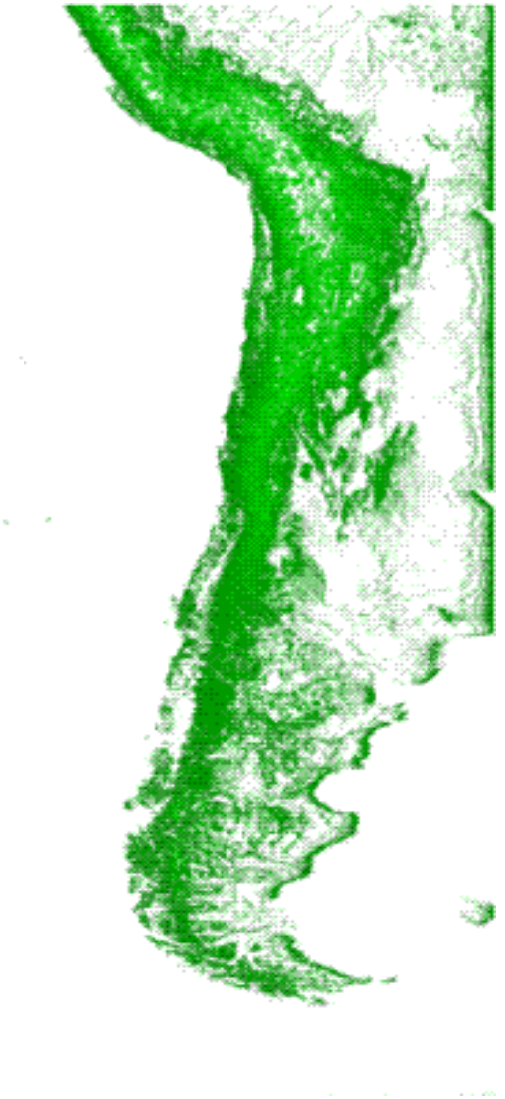
There is a severe problem

- how to set the "offset" points and how to place them
- what is the minimum of the "offset" points, as the computational cost grows significantly

Meshless Interpolations

Non-trivial problems





Turbine blade with 594 000 points

Meshless Interpolations

Visualization & Meshless representation

Meshless methods offer natively

- smoothness of the physical phenomena, i.e. $F(\boldsymbol{x}) = h$, resp. $F(\boldsymbol{x}) = \boldsymbol{h}$ generally in d-dimensional space
- analytical form for derivatives , e.g. $\varphi(\|\boldsymbol{x}\|) = \varphi\left(\sqrt{x^2 + y^2}\right)$

$$\frac{\partial \varphi(\|\boldsymbol{x}\|)}{\partial x} = \frac{\partial}{\partial r}\varphi(r)\frac{\partial}{\partial x}r(x,y) = \frac{\partial}{\partial r}\varphi(r)\frac{x}{\sqrt{x^2+y^2}}$$

$$= \frac{x}{r}\frac{\partial}{\partial r}\varphi(r) \qquad \frac{\partial \varphi(\|\boldsymbol{x}\|)}{\partial x} = \frac{y}{r}\frac{\partial}{\partial r}\varphi(r)$$

$$\frac{\partial}{\partial x}f(\boldsymbol{x}_i) = \sum_{j=1}^{N}\lambda_j \frac{\partial}{\partial x}\varphi\left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right) = \sum_{j=1}^{N}\lambda_j \frac{x}{r_{ij}}\frac{\partial}{\partial x}\varphi\left(r_{ij}\right)$$

where $r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$

Similarly for the case of approximation $r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|$

where $\boldsymbol{\xi}_j$ are virtual points given by a user

Meshless Interpolations

Visualization & Meshless representation

In data visualization no "high precision" is needed for the visual assessment of the behavior of the physical phenomena

- reduction of weights λ_i => user controlled =>approximation
- hierarchical approach

In the case of precision required

- Progressive RBF interpolation – point insertion or point removal of computational complexity $O(N^3)$ to $O(N^2)$ - using block matrix operations

J. Süßmuth, Q. Meyer andG. Greiner: Surface Reconstruction Based on Hierarchical Floating Radial Basis Functions, Computer Graphics Forum, Vol.29, No.6, pp. 1854–1864, 2010

Skala,V: Progressive RBF Interpolation, 7th Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, Afrigraph 2010, pp.17-20, ACM, 2010

Meshless Interpolations

Summary

Meshless methods are:

- Progressively developing methods in many fields ranging from computational sciences, e.g. partial differential equations, solving economical problems, visualization, computer graphics etc.
- Offering unique properties
  - natural smoothness
  - applicability in d −dimensional problems
- Parallelization – as matrix-vector operations are used, relevant specialized libraries available on CPUs and GPUs
  GPUML (GPU fro Machine Learning open SW), GMRES, etc.
- If CSRBF are used, simple use for wide range of data sets
  space subdivision and parallelization techniques can be applied

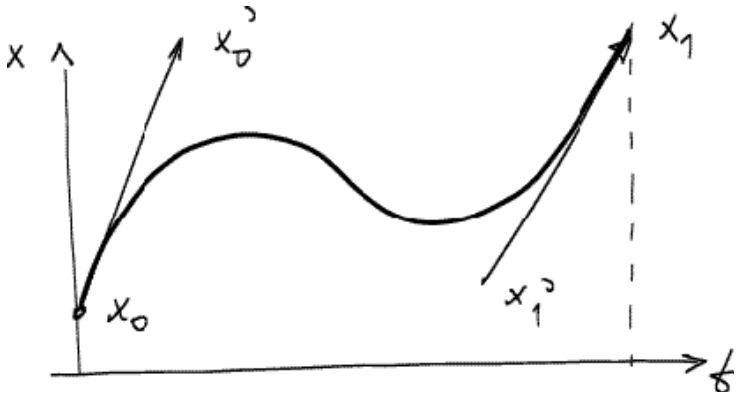
Meshless Interpolations

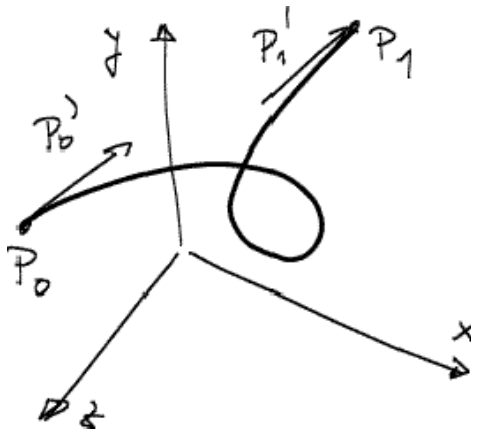
Parametric curves

Meshless Interpolations

Hermite curve

Cubic curve given by two end-points and two tangent vectors





$$x(t) = at^3 + bt^2 + ct + d \qquad x'(t) = 3at^2 + 2bt + c$$

Substituting $t = 0$ and $t = 1$ we get 4 equations for unknown values, i.e. a, b, c, d

$$x(0) = d \qquad x'(0) = c$$
$$x(1) = a + b + c + d \qquad x'(1) = 3a + 2b + c$$

=>solve $\boldsymbol{Ax} = \boldsymbol{b}$

Meshless Interpolations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix}$$

Solution

$$a = x(0)$$

$$b = x'(0)$$

$$c = -3x(0) + 3x(1) - 2x'(0) - x'(1)$$

$$d = -2x(0) - 2x(1) + x'(0) + x'(1)$$

and we can write for the x coordinate

$$x(t) = \boldsymbol{x}^T \boldsymbol{M}_H \boldsymbol{t} \qquad \boldsymbol{x} = [x(0), x(1), x'(0), x'(1)]^T \qquad \boldsymbol{t} = [t^3, t^2, t, 1]^T$$

$$\boldsymbol{M}_H = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Meshless Interpolations

Usually a notation

$$\boldsymbol{P}(t) = \boldsymbol{P}^T \boldsymbol{M}_H \boldsymbol{t} \qquad \boldsymbol{P} = [P_0, P_1, P_0', P_1']^T \qquad \boldsymbol{t} = [t^3, t^2, t, 1]^T$$

is used to express that x,y and z coordinated are to be taken.

$\boldsymbol{M}_H$ is a matrix of the Hermite form and **blending functions** are

$$\boldsymbol{g}_H(t) = [g_1(t), g_2(t), g_3(t), g_4(t)]^T = \boldsymbol{M}_H\ [t^3, t^2, t, 1]^T \quad t \in < 0,1 >$$

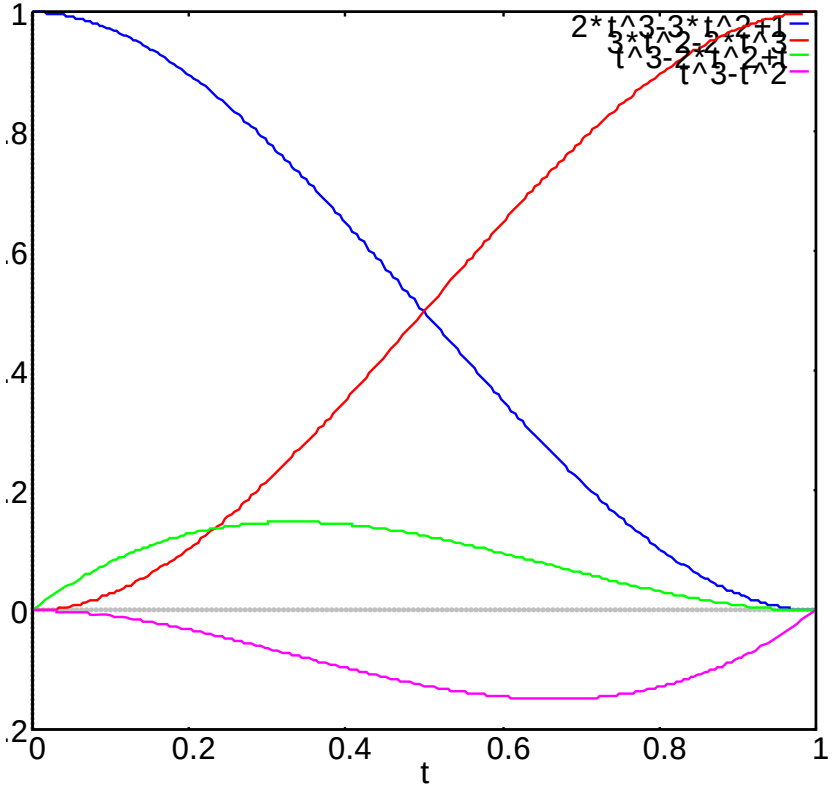
i.e.

$$g_1(t) = 2t^3 - 3t^2 + 1$$

$$g_2(t) = -2t^3 + 3t^2$$

$$g_3(t) = t^3 - 2t^2 + t$$

$$g_4(t) = t^3 - t^2$$



Meshless Interpolations

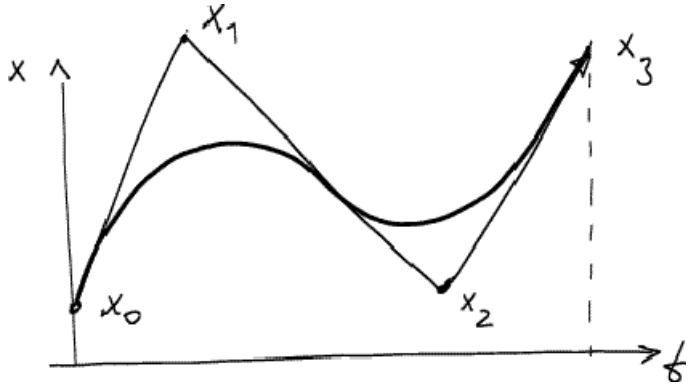
Bézier curve

Bicubic Bézier curve is given as similarly as:

$$P(t) = [P_0, P_1, P_2, P_3]\ \boldsymbol{M}_B\ [t^3, t^2, t, 1]^T \quad t \in < 0,1 >$$

Important property of the Bézier curve is that it is always inside of the convex hull of the given control points

$$\boldsymbol{M}_B = \begin{bmatrix} -1 & 3 & 3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



General definition

$$P(t) = \sum_{i=0}^{n} B_i^n(t) P_i$$

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-1} t^i$$

Meshless Interpolations

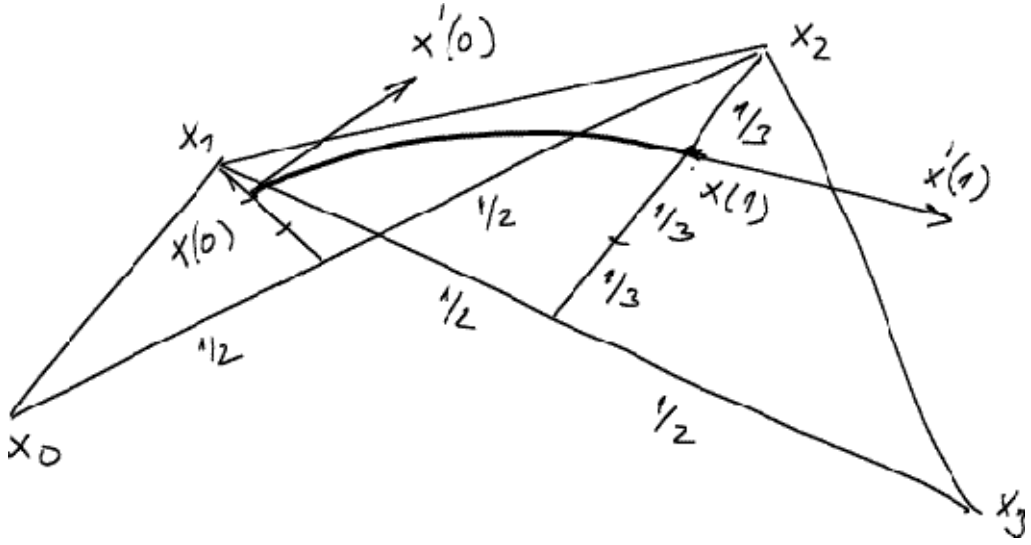
Coons curve

This curve is different – it does not pass the control points but it is naturally C^2 continuous.

$$\boldsymbol{P}(t) = [P_0, P_1, P_2, P_3]\, \boldsymbol{M}_C\, [t^3, t^2, t, 1]^T \quad t \in < 0,1 >$$

The Coons matrix is given as

$$\boldsymbol{M}_C = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



Meshless Interpolations

General form of cubic parametric curves

$$\boldsymbol{P}(t) = [P_0, P_1, P_2, P_3]\, \boldsymbol{M}_F\ [t^3, t^2, t, 1]^T \quad t \in < 0,1 >$$

or as

$$x(t) = \boldsymbol{x}^T \boldsymbol{M}_F\, \boldsymbol{t} \qquad y(t) = \boldsymbol{y}^T \boldsymbol{M}_F\, \boldsymbol{t}$$
$$z(t) = \boldsymbol{z}^T \boldsymbol{M}_F\, \boldsymbol{t} \qquad \boldsymbol{t} = [t^3, t^2, t, 1]^T \quad t \in \langle 0,1 \rangle$$

where $\boldsymbol{M}_F$ is a matrix of the form used

and kernel functions are given as

$$\boldsymbol{g}_F(t) = \boldsymbol{M}_F\ [t^3, t^2, t, 1]^T \quad t \in \langle 0,1 \rangle$$

However Hermite or Bezier curves have to be smoothly connected

How to make it?

Meshless Interpolations

Curves joining

In applications a connection of cubic curves is needed to get a complex shape.

Continuity (parametric)

- C^0 – two segments are connected, i.e. share a common point
- C^k – a k^{th} derivative of the first segment at the end point is equal to a k^{th} derivative of the second segment in the starting point

Unfortunately the C^k (k>1) continuity does not ensure continuity (smoothness) if a curve is rendered in x,y, resp. x,y,z.

Usually segments are smoothly connected; however each segment might be, in principle, of different form.

Meshless Interpolations

Smooth Hermite curves C^2 joining and ${}^2P(t)$

$$P(t) = \boldsymbol{P}^T \boldsymbol{M}_H \boldsymbol{t} \qquad \boldsymbol{P} = [P_0, P_1, P_0', P_1']^T \qquad \boldsymbol{t} = [t^3, t^2, t, 1]^T$$

For

- C^0 connection - ${}^1P(1) = {}^2P(0)$ - points share the same coordinates
- C^1 connection - $\frac{d}{dt}\,{}^1P(1) = \frac{d}{dt}\,{}^2P(0)$ simple solution ${}^{i-1}P'(1) = {}^{i-1}P'(0)$ – speed of a moving object changes continuously
- C^2 connection $\frac{d^2}{dt^2}\,{}^1P(1) = \frac{d^2}{dt^2}\,{}^2P(0)$ – acceleration changes continuously – **How to make it?**

Meshless Interpolations

Let us consider a simple example of C^2 connectivity

$$P(t) = at^3 + bt^2 + ct + d \qquad \frac{d}{dt}P(t) = 3at^2 + 2bt + c \qquad \frac{d^2}{dt^2}P(t) = 6at + 2a$$

$${}^{i-1}P''(1) = {}^{i}P''(0) \qquad \text{leads to a condition} \qquad 6^{i-1}a + 2^{i-1}b = 2\ {}^{i}b$$

$$a = P(0) \qquad b = P'(0)$$

$$c = -3P(0) + 3P(1) - 2P'(0) - P'(1) \qquad d = -2P(0) - 2P(1) + P'(0) + P'(1)$$

Putting together with ${}^{1}P(1) = {}^{2}P(0)$

$$2\left[3\left({}^{i}P - {}^{i-1}P\right) - 2^{i-1}P' - {}^{i}P'\right] + 6\left[2\left({}^{i-1}P - {}^{i}P\right) + 2^{i-1}P' + {}^{i}P'\right]$$
$$= 2\left[3\left({}^{i+1}P - {}^{i}P\right) - 2\ {}^{i}P' - {}^{i+1}P'\right]$$

Simplifying

$${}^{i-1}P' + 4\ {}^{i}P' + {}^{i+1}P' = 3\left({}^{i+1}P - {}^{i-1}P\right)$$

Meshless Interpolations

In a matrix form

$$\begin{bmatrix} 1 & 0 & & & & \\ 1 & 4 & 1 & 0 & & \\ 0 & 1 & 4 & 1 & 0 & \\ & & & \ddots & & \\ & & 0 & 1 & 4 & 4 \\ & & & & 0 & \end{bmatrix} \begin{bmatrix} {}^0P' \\ {}^0P' \\ \vdots \\ \vdots \\ {}^{m-2}P' \\ {}^{m-1}P' \end{bmatrix} = \begin{bmatrix} {}^0P' \\ 3\left({}^2P - {}^0P\right) \\ \\ \\ 3\left({}^{m-1}P - {}^{m-3}P\right) \\ {}^{m-1}P' \end{bmatrix}$$

If ${}^0P''(0) = {}^mP''(0) = 0$ => *natural cubic spline*

It can be shown, that a relation between Hermite and Bézier forms exists as $x_0' = 3(x_1^B - x_0^B)$ and $x_1' = 3(x_3^B - x_2^B)$

$$\left[x_0\ , x_0', x_1\ x_1'\right]^T = \begin{bmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_0^B \\ x_1^B \\ x_2^B \\ x_3^B \end{bmatrix}$$

that is actually a transformation from Bézier form to Hermite form. Similarly between other forms, see next.

Meshless Interpolations

As all the above mentioned parametric curves are of the form

$$\boldsymbol{P}(t) = [P_0, P_1, P_2, P_3]\, \boldsymbol{M}_F\, [t^3, t^2, t, 1]^T \quad t \in < 0,1 >$$

there are mutual transformations possible - $\boldsymbol{M}_{from,to}$

	From	**Hermite**	**Bezier**	**B-Spline**
To	**Hermite**	Identical	$\begin{bmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{bmatrix}$	$\frac{1}{6}\begin{bmatrix} 1 & 0 & -3 & 0 \\ 4 & 1 & 0 & -3 \\ 1 & 4 & 3 & 0 \\ 0 & 1 & 0 & 3 \end{bmatrix}$
	Bezier	$\frac{1}{3}\begin{bmatrix} 3 & 3 & 0 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$	Identical	$\frac{1}{6}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 4 & 4 & 2 & 1 \\ 1 & 2 & 4 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
	B-Spline	$\frac{1}{3}\begin{bmatrix} -3 & 6 & -3 & 6 \\ 6 & -3 & 6 & -3 \\ -7 & 2 & -1 & 2 \\ -2 & 1 & -2 & 7 \end{bmatrix}$	$\begin{bmatrix} 6 & 0 & 0 & 0 \\ -7 & 2 & -1 & 2 \\ 2 & -1 & 2 & -7 \\ 0 & 0 & 0 & 6 \end{bmatrix}$	Identical

Meshless Interpolations

Rational Bézier curve

Euclidean $\boldsymbol{X} = [X, Y, Z]^T$

$$\boldsymbol{X}(t) = \frac{\sum_{i=0}^{n} B_i^n(t) w_i q_i}{\sum_{i=0}^{n} B_i^n(t) w_i} \qquad 0 \le t \le 1 \qquad B_i^n(t) = \binom{n}{i} (1-t)^{n-1} t^i$$

1st derivative $\left[\frac{a}{b}\right]' = \frac{a'b - ab'}{b^2}$ quite complicated

Projective $\boldsymbol{x} = [x, y, z : w]^T$

$$\boldsymbol{x}(t) = \sum_{i=0}^{n} B_i^n(t) q_i \qquad \boldsymbol{x}'(t) = \sum_{i=0}^{n} (B_i^n(t))' q_i$$

How simple !

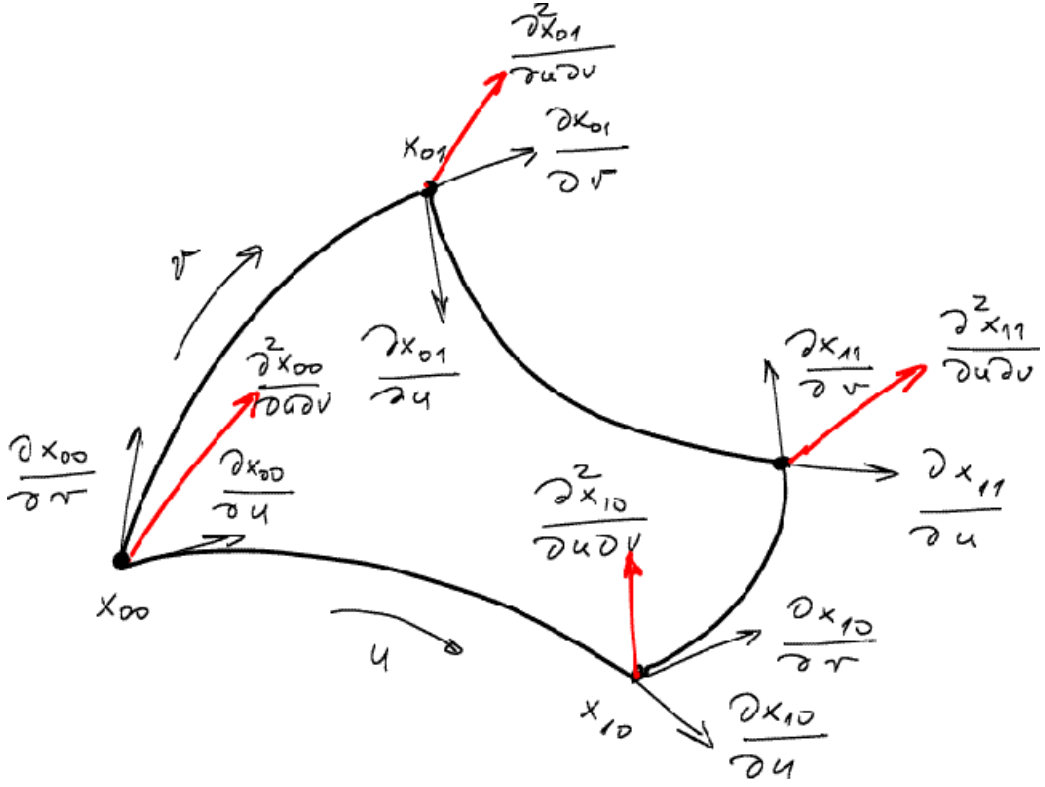
Meshless Interpolations

Parametric surfaces

Meshless Interpolations

Parametric bicubic surfaces

A parametric bicubic surface (patch) $P(u,v)$ is given as a "tensor" product of cubic curves, i.e. curves $P(u)$ and $P(v)$, $u,v \in \langle 0,1 \rangle$. Requirement: is that $P(u,v)$ for $u = const$ or $v - const$ are cubic curves as well.



However, diagonal and antidiagonal curves are of the degree !!!!

Note, that the domain for (u,v) must be squared.

As the patch is rendered as a triangular mesh, at the end, due to the non-linear parameterization triangles are of a different size!

Meshless Interpolations

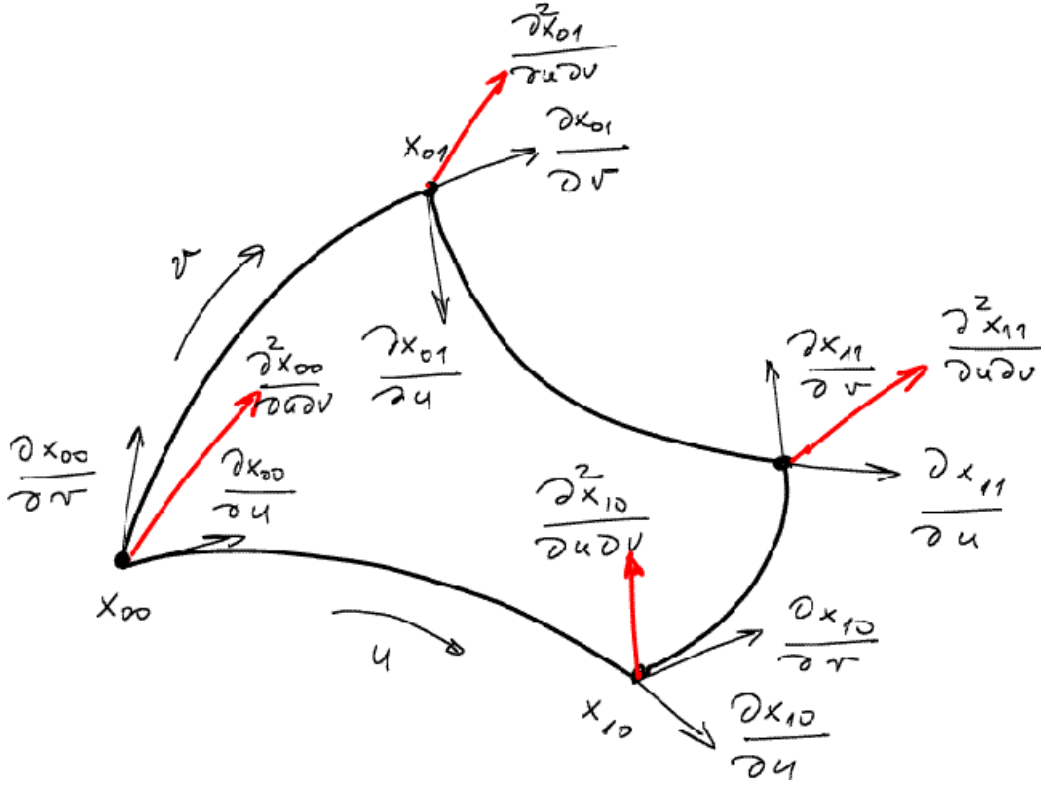
Hermite (Ferguson) bicubic patch

- 4 corner points
- 8 tangential vectors
- 4 twist vectors

16 control values for x, y, z

$\boldsymbol{P}_H$ - matrix of control values

$\boldsymbol{M}_H$ - Hermite matrix



$$\boldsymbol{P}(u,v) = [u^3, u^2, u, 1]\, \boldsymbol{M}_H^T\, \boldsymbol{P}_H\, \boldsymbol{M}_H\, [v^3, v^2, v, 1]^T \qquad u, v \in \langle 0,1 \rangle$$

Meshless Interpolations

Hermite bicubic patch

$$\boldsymbol{P}(u,v) = [u^3, u^2, u, 1]\, \boldsymbol{M}_H^T\, \boldsymbol{P}_H\, \boldsymbol{M}_H\, [v^3, v^2, v, 1]^T \quad u, v \in \langle 0,1 \rangle$$

$$\boldsymbol{P}(u,v) = \left[P_x(u,v), P_y(u,v), P_z(u,v)\right]^T$$

Hermite control values

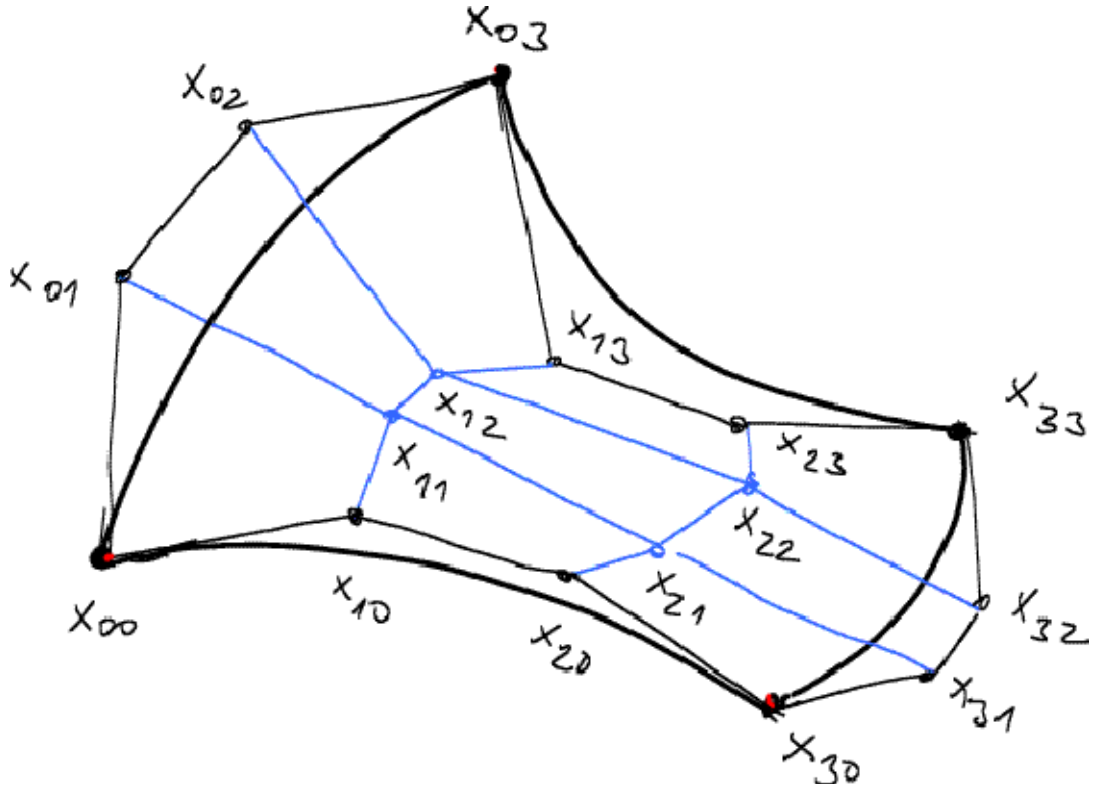
$$\boldsymbol{P}_H = \begin{bmatrix} \begin{bmatrix} P_{00} & x_{01} \\ P_{10} & x_{11} \end{bmatrix} & \dfrac{\partial}{\partial v}\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} \\ \dfrac{\partial}{\partial u}\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} & \dfrac{\partial^2}{\partial u \partial v}\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} \end{bmatrix}$$

Matrix of the Hermite form

$$\boldsymbol{M}_H = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Meshless Interpolations

Bézier bicubic patch



- 4 corner points
- 8 outer points
- 4 inner points

16 control values for x, y, z

$\boldsymbol{P}_B$ - control points

$\boldsymbol{M}_B$ - matrix of the Bézier form

Meshless Interpolations

$$\boldsymbol{P}(u,v) = [u^3, u^2, u, 1]\, \boldsymbol{M}_F^T\, \boldsymbol{P}_F\, \boldsymbol{M}_F\, [v^3, v^2, v, 1]^T \qquad u, v \in \langle 0,1 \rangle$$

$$\boldsymbol{P}(u,v) = \left[P_x(u,v), P_y(u,v), P_z(u,v)\right]^T$$

Bézier control points

$$\boldsymbol{P}_B = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix}$$

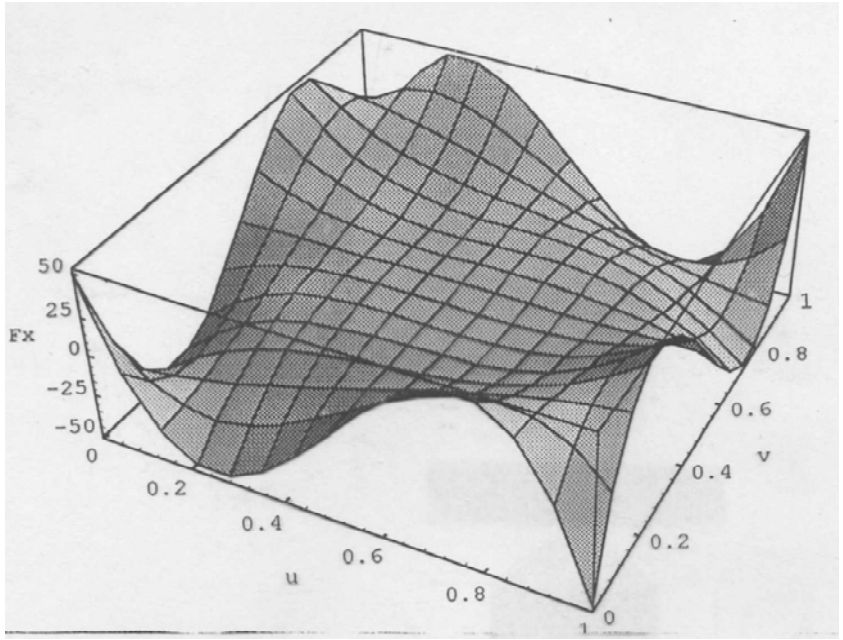
Matrix of the Bézier form

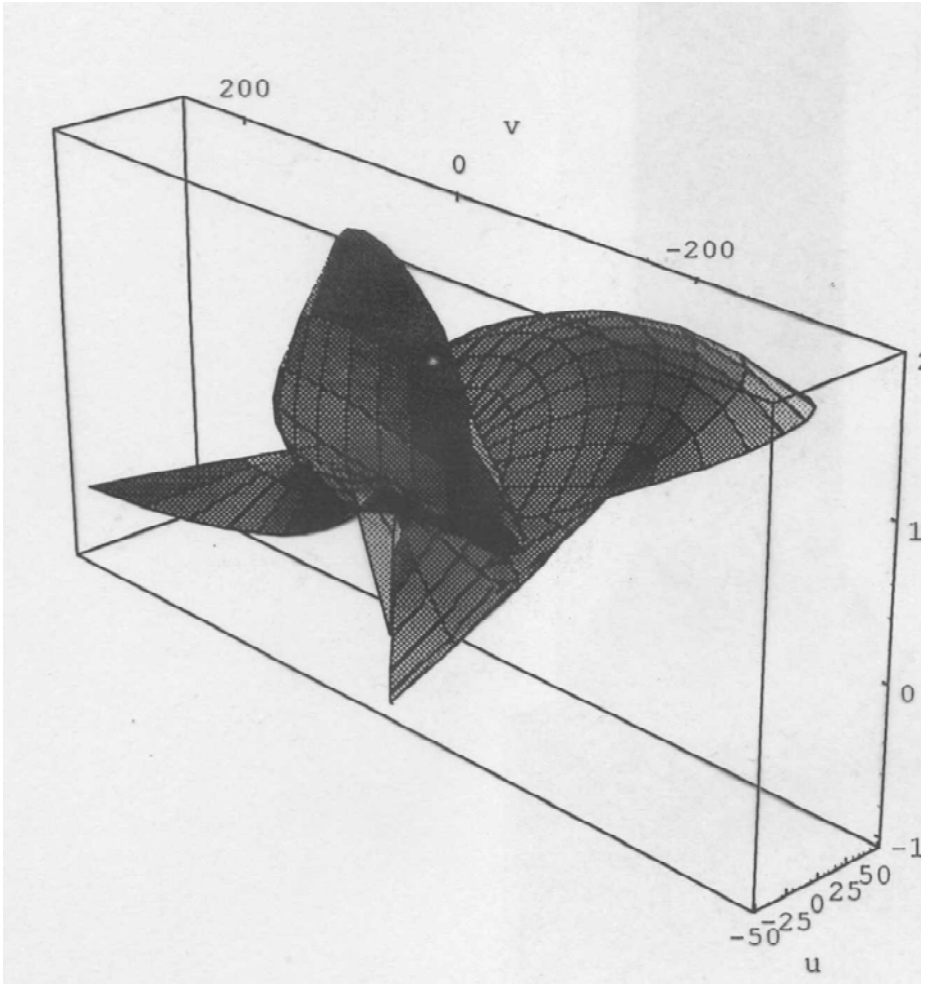
$$\boldsymbol{M}_B = \begin{bmatrix} -1 & 3 & 3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Important

Surface is inside of the convex hull determined by the control points

Meshless Interpolations





Parametric space

Euclidean space

Meshless Interpolations

Coons form

Hermite (Ferguson) and Bézier surfaces passes the "corner" control points. Coons surfaces do not pass control points, but they are C^2 continuous naturally.

$$\boldsymbol{P}(u,v) = [u^3, u^2, u, 1]\, \boldsymbol{M}_C^T\, \boldsymbol{P}_C\, \boldsymbol{M}_C\, [v^3, v^2, v, 1]^T \qquad u, v \in \langle 0,1 \rangle$$

$$\boldsymbol{P}(u,v) = \left[P_x(u,v), P_y(u,v), P_z(u,v)\right]^T$$

Coons control points

$$\boldsymbol{P}_C = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix}$$

Matrix of the Coons form

$$\boldsymbol{M}_C = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Meshless Interpolations

General form and mutual transformations

So far all bicubic parametric patches are of the form

$$P(u,v) = [u^3, u^2, u, 1]\, \boldsymbol{M}_F^T\, \boldsymbol{P}_F\, \boldsymbol{M}_F\, [v^3, v^2, v, 1]^T$$

where $\boldsymbol{M}_F$ is a matrix 4×4.

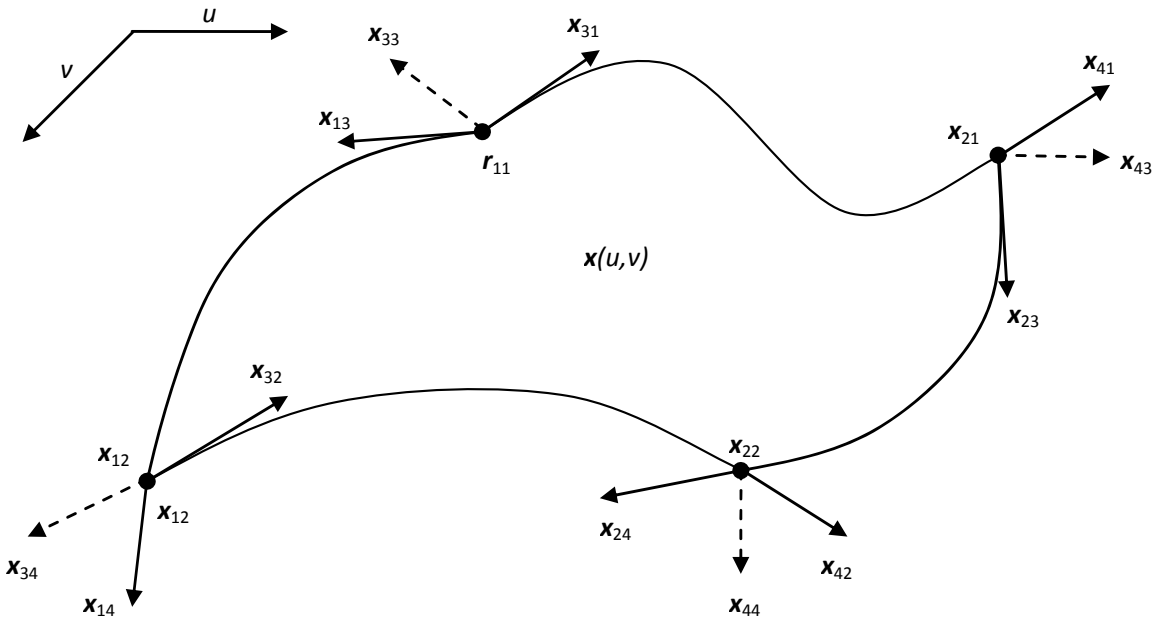
It means that we can describe a smooth surface by 3×16 values, i.e. by 16 values for each coordinate.

Meshless Interpolations

Transformation from Hermite to Bézier

For x coordinate

$$\boldsymbol{X}_B = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} =$$

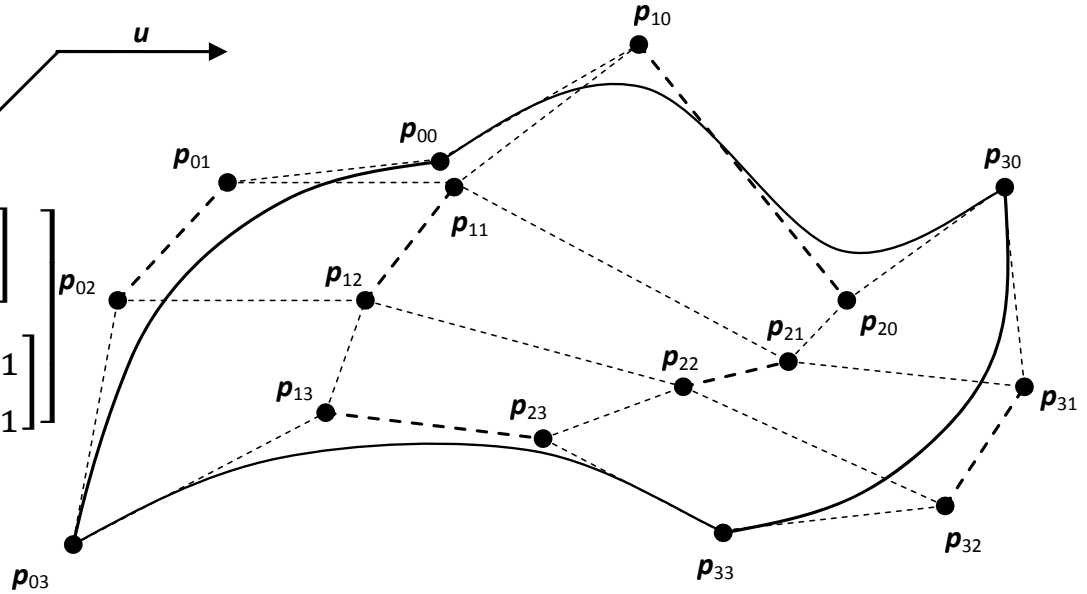


$$\begin{bmatrix} x_{11} & x_{11}+\frac{1}{3}x_{13} & x_{12}-\frac{1}{3}x_{14} & x_{12} \\ x_{11}+\frac{1}{3}x_{31} & x_{11}+\frac{1}{3}(x_{13}+x_{31})-\frac{1}{9}x_{33} & x_{12}+\frac{1}{3}(x_{32}-x_{14})-\frac{1}{9}x_{34} & x_{12}+\frac{1}{3}x_{32} \\ x_{21}-\frac{1}{3}x_{41} & x_{21}+\frac{1}{3}(x_{23}-x_{41})-\frac{1}{9}x_{43} & x_{22}-\frac{1}{3}(x_{24}+x_{42})-\frac{1}{9}x_{44} & x_{22}-\frac{1}{3}x_{42} \\ x_{21} & x_{21}+\frac{1}{3}x_{23} & x_{22}-\frac{1}{3}x_{24} & x_{22} \end{bmatrix}$$

Meshless Interpolations

Transformation from Bézier to Hermite

$$\boldsymbol{P}_H = \begin{bmatrix} \begin{bmatrix} P_{00} & x_{01} \\ P_{10} & x_{11} \end{bmatrix} & \dfrac{\partial y}{\partial v}\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} \\ \dfrac{\partial}{\partial u}\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} & \dfrac{\partial^2}{\partial u \partial v}\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} \end{bmatrix}$$



For x coordinate

$$\boldsymbol{X}_H = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} =$$

$$\begin{bmatrix} x_{00} & x_{03} & 3(x_{01} - x_{00}) & 3(x_{03} - x_{02}) \\ x_{30} & x_{33} & 3(x_{31} - x_{300}) & 3(x_{33} - x_{32}) \\ 3(x_{10} - x_{00}) & 3(x_{13} - x_{03}) & 9(x_{00} - x_{01} - x_{10} + x_{11}) & 9(x_{02} - x_{03} - x_{12} + x_{13}) \\ 3(x_{30} - x_{20}) & 3(x_{33} - x_{23}) & 9(x_{20} - x_{21} - x_{30} + x_{31}) & 9(x_{22} - x_{23} - x_{32} + x_{33}) \end{bmatrix}$$

Meshless Interpolations

There are the following issues to be considered if used as interpolation:

- if the patch is to be rendered usually is tessellated by a triangular mesh
- the z value is usually taken as value of a function $z = f(x(u), y(v))$, however this requires regular tessellation (squared), and due to the non-linearity of a parameterization, final result might be far from being expected

- if a patch tessellated by $\Delta u = const$, resp. $\Delta v = const$ then $\Delta x \neq const$, resp. $\Delta x \neq const$.

Meshless Interpolations

Summary and conclusion

Meshless Interpolations

Summary and conclusion

We have got within this course an understanding of:

- Numerical representation and precision issues
- Coordinate systems, duality, transformations
- Data types, structures, classification
- Meshes
- Interpolation of ordered data
- Interpolation of un-ordered data
- Approximation - Least Square Error
- Approximation of un-ordered data
- Examples & experimental results
- Parametric curves
- Parametric surfaces

Acknowledgment – some items included in this presentation were downloaded from the Internet open resources and authors are acknowledged if they are known. Thanks belong to them.

Meshless Interpolations

APPENDIX – RBF testing functions

Meshless Interpolations

RBF testing functions used for testing (used in references)

Function $F1$ $x, y \in \langle 0,1\rangle \times \langle 0,1\rangle$ Franke's function

$$F1(x,y) = \frac{3}{4}\exp\left(\frac{-1}{4}((9x-2)^2 + (9y-2)^2)\right) + \frac{3}{4}\exp\left(\frac{-1}{49}(9x-2)^2 - \frac{1}{10}(9y-2)^2\right) + \frac{1}{2}\exp\left(\frac{-1}{4}(9x-7)^2 - \frac{1}{4}(9y-3)^2\right) + \frac{1}{5}\exp(-(9x-4)^2 + (9y-7)^2) \quad (1)$$

Function $F2$ $x, y \in \langle 0,1\rangle \times \langle 0,1\rangle$

$$F2(x,y) = \frac{1}{9}[\tanh(9y - 9x)] + 1 \quad (2)$$

Function $F3$ $x, y \in \langle 0,1\rangle \times \langle 0,1\rangle$

$$F3 = \frac{1.25 + \cos(5.4y)}{6[1 + (3x-1)^2]} \quad (3)$$

Function $F4$ $x, y \in \langle 0,1\rangle \times \langle 0,1\rangle$

$$F4(x,y) = \frac{1}{3}\exp\left[-\frac{81}{16}\left(\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2\right)\right] \quad (4)$$

Function $F5$ $x, y \in \langle 0,1\rangle \times \langle 0,1\rangle$

$$F5(x,y) = \frac{1}{3}\exp\left[-\frac{81}{4}\left(\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2\right)\right] \quad (5)$$

Function $F6$ $x, y \in \langle 0,1\rangle \times \langle 0,1\rangle$

$$F6(x,y) = \frac{1}{9}\left[64 - 81\left(\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2\right)\right] - \frac{1}{2} \quad (6)$$

Function $F7$ $x, y \in \langle -1,1\rangle \times \langle -1,1\rangle$

Meshless Interpolations

$$F7(x,y) = \exp\left[-15\left(\left(x-\frac{1}{2}\right)^2 + y^2\right)\right] + \frac{1}{2}exp\left[-20\left(\left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{4}\right)^2\right)\right] - \frac{3}{4}\exp\left[-8\left(\left(x+\frac{1}{2}\right)^2 + \left(y+\frac{1}{2}\right)^2\right)\right] \quad (7)$$

Function $F8$ $\quad x,y \in \langle -2,2\rangle \times \langle -2,2\rangle$

$$F8(x,y) = \sin(3x)\cos(3y) \quad (8)$$

Function $F9$ $\quad x,y \in \langle 0,1\rangle \times \langle 0,1\rangle$

$$F9(x,y) = x\exp(-x^2 - y^2) \quad (9)$$

Function $F10$ Peak $x,y \in \langle -3,3\rangle \times \langle -3,3\rangle$

$$F10(x,y) = 3(1-x)^2\exp(-x^2-(y+1)^2) - 10\left(\frac{x}{5} - x^3 - y^3\right)\exp(-x^2-y^2) - \frac{1}{3}\exp(-(x+1)^2 - y^2) \quad (10)$$

Meshless Interpolations

Marschner-Lobb grid $511 \times 511 \times 511$

$$F1(x,y,z) = \frac{1 + \alpha\left(1 + \cos(2f_m)\,\pi\cos\left(\frac{1}{2}\pi\sqrt{x^2+y^2}\right)\right) - \sin\left(\frac{\pi z}{2}\right)}{2(1+\alpha)} \tag{11}$$

Peaks grid $353 \times 353 \times 1069$

$$F2(x,y,z) = \left(\sqrt{3}+\sqrt{3}x\right)^2 \exp(-x^2-(y+1)^2) - 10\left(\frac{x}{5} - x^3 - y^5\right)\exp(-x^2-y^2) - \frac{1}{3}\exp(-(x+1)^2-y^2) - z \tag{12}$$

Genus3 grid $511 \times 511 \times 511$

$$F3(x,y,z) = \left[1 - \left(\frac{x}{6}\right)^2 - \left(\frac{y}{3.5}\right)^2\right][(x-3.9)^2+y^2-1.44](x^2+y^2-1.44)[(x+3.9)^2+y^2-1.44] - 256z^2 \tag{13}$$

Six peaks grid $5595 \times 595 \times 373$

$$F4(x,y,z) = (3x^2-y^2)^2y^2 - (x^2+y^2)^4 - z^3 - 0.001z \tag{14}$$

Flower grid $511 \times 511 \times 511$

$$F5(x,y,z) = \sin(3\Theta)\sin(4\phi) - r \tag{15}$$

where: ϕ azimuth coordinate, Θ zenith coordinate, r maximum distance from a surface

Meshless Interpolations

References

[Ana93] Anand,V.: Computer Graphics and Geometric modeling for Engineers, John Wiley&Sons, 1993

[Bez86] Bezier,P.: The Mathematical Basis of the UNISURF CAD System, Butterworths, 1986

[Bis08] Biswas,S., Lovell,B.C.: Bezier and Splines in Image Processing, Springer, 2008

[Boe94] Boehm,W., Prautzsch,H.: Geometric Concepts for Geometric Design, A.K.Peters, 1994

[Coh01] Cohen,E., Riesenfeld,R.F., Elber,G.: Geometric Modeling with Splines, A.K.Peters, 2001

[Far99] Farin,G.E.: NURBS: From Projective Geometry to Practical Use, A.K.Peters, 1999

[Far02] Farin,G., Hoschek,J., Kim,M,.S. (Ed.): Handbook of Computer Aided Geometric Design, North Holland, 2002

[Fas07] Fasshauer,G.E.: Meshfree Approximation Methods with MATLAB, World Scientific Publishing, 2007

[Har71] Hardy, R.L.: Multiquadric Equations of Topography and other Irregular Surfaces. Journal of Geophysical Research, 76(8):1905–1915, 1971.

[Kav11] Kaw,A.: History of interpolation, http://www.saylor.org/site/wp-content/uploads/2011/11/ME205-5.1-TEXT2.pdf

[Mei02] Meijering,E.: A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing, *Proceedings of the IEEE*, vol. 90, no. 3, March 2002, pp. 319-342, Download http://www.imagescience.org/meijering/research/chronology/

[Mil96] Mills,T.: Join the Dots and See the World, Worner Research Lecture, 1996 http://www.latrobe.edu.au/worner/1996

[Rok96] Rockwookd,A., Chambers,P.: Interactive Curves and Surfaces, A.K.Peters, 1996

[Sab10] Sabin,M.: Analysis and Design of Univariate Subdivision Schemes, Springer 2010

[Saf08] Sarfaz,M.: Interactive Curve modeling, Springer, 2008

[Say11] http://www.saylor.org/site/wp-content/uploads/2011/11/ME205-5.1-TEXT2.pdf

[Ska08a] Skala,V.: Barycentric Coordinates Computation in Homogeneous Coordinates, Computers & Graphics, Elsevier, ISSN 0097-8493, Vol. 32, No.1, pp.120-127, 2008

[Ska08b] Skala,V.: Intersection Computation in Projective Space using Homogeneous Coordinates, Int. Journal of Image and Graphics, Vol.7, No.4, pp.615-628, 2008

[Ska06a] Skala,V.: Length, Area and Volume Computation in Homogeneous Coordinates, Int. Journal of Image and Graphics, Vol.6., No.4, pp.625-639, 2006

Meshless Interpolations

[Ska06b] Skala,V.: GPU Computation in Projective Space Using Homogeneous Coordinates , Game Programming GEMS 6 (Ed.Dickheiser,M.), pp.137-147, Charles River Media, 2006

[Ska06b] Skala,V.: A new approach to line and line segment clipping in homogeneous coordinates, The Visual Computer, Vol.21, No.11, pp.905-914, Springer, 2005

[War02] Warren,J., Weimer,H.: Subdivision Methods for Geometric Design: A Constructive Approach, Morgan Kaufmann Publ., 2002

References related

- Yamaguchi,F.:Computer-Aided Geometric Design: A Totally Four-Dimensional Approach, Springer, 2002
- Agoston,M.K.: Computer Graphics and Geometric Modeling - Mathematics, ISBN 1-58233-817-2, Springer, 2005

Additional references

[Adam08] Adams,B., Ovsjanikov,M., Wand,M., Seidel,H.-P., Guibas,L.J.: Meshless modeling of Deformable Shapes and their Motion, Eurographics/AACM SIGGRAPH Symposium on Computer Animation, 2008

[Buhm03] Buhmann,M.D.: Radial Basis Functions: Theory and Implementations, Cambridge University Press, ISBN 978-0-521-63338-3, 2008

[Carr01] Carr,J.C., Beatsom,R.K., Cherrie,J.B., Mitchell,T.J., Fright,W.R., Fright,B.C. McCallum,B.C, Evans,T.R.: Reconstruction and representation of 3d objects with radial basis functions, in Proceedings of SIGGRAPH'01, 2001, pp. 67–76.

[Cerm04] Cermak,M., Skala,V.: Edge spinning algorithm for implicit surfaces, Applied Numerical Mathematics, ISSN 0168-9274, Elsevier, Vol.49, No.3-4, pp.331-342, , 2004

[Cerm05] Cermak,M., Skala,V.: Polygonization of Implicit Surfaces with Sharp Features by the Edge Spinning, The Visual Computer, Springer Verlag, Vol.21, No4., ISSN 0178-2789, pp.252-264, 2005

[Cerm07] Cermak,M., Skala,V.: Polygonisation of disjoint implicit surfaces by the adaptive edge spinning algorithm, Int.J.Computational Science and Engineering, ISSN 1742-7185, Vol.3, No.1, pp.45-52, 2007

Meshless Interpolations

[Duch77] Duchon,J.: Splines minimizing rotation-invariant semi-norms in Sobolev space, Constructive Theory of Functions of Several Variables, LNCS 571, Springer, 1997

[Hard71] Hardy,L.R.: Multiquadric equation of topography and other irregular surfaces, Journal of Geophysical Research 76 (8), 1905-1915, 1971

[Fass07] Fasshauer,G.E.: Meshfree Approximation Methods with MATLAB, World Scientific Publishing, ISBN 981-270-634-8, 2007

[Lazz2002] Lazzaro,D., Montefusco,L.B.: Radial Basis functions for multivariate interpolation of large data sets, J.of Computational and Applied Mathematics, 140, pp. 521-536, 2002

[Mace11] Macedo,I., Gois,J.P., Velho,L.: Hermite Interpolation of Implicit Surfaces with Radial Basis Functions, Computer Graphics Forum, Vol.30, No.1, pp.27-42, 2011

[Naka11] Nakata,S., Takeda,Y., Fujita,N., Ikuno,S.: Parallel Algorithm for Meshfree Radial Point Interpolation Method on Graphics Hardware,IEEE Trans.onMagnetics, Vol.47, No.5, pp.1206-1209, 2011

[PanR11] Pan,R., Skala,V.: A two level approach to implicit modeling with compactly supported radial basis functions, Engineering and Computers, Vol.27. No.3, pp.299-307, ISSN 0177-0667, Springer, 2011

[PanR12] Pan,R., Skala,V.: Surface Reconstruction with higher-order smoothness, The Visual Computer, on-line, Vol.28, No.2., pp.155-162, ISSN 0178-2789, Springer, 2012

[Ohta03] Ohtake,Y., Belyaev,A., Seidel,H.-P.: A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In: Proceedings of international conference shape modeling. IEEE Computer Society, Washington, pp 153–161, 2003

[Savc95] Savchenko,V., Pasko,A., Okunev,O., Kunii,T.L.: Function representation of solids reconstructed from scattered surface points and contours. Computer Graphics Forum 14(4), pp.181–188, 1995

[Skal12]Skala,V.: Projective Geometry and Duality for Graphics, Games and Visualization - Course SIGGRAPH Asia 2012, Singapore, ISBN 978-1-4503-1757-3, 2012

[Skal13a] Skala,V., Pan,R.J., Nedved,O.: Simple 3D Surface Reconstruction Using Flatbed Scanner and 3D Print - poster, ACM SIGGRAPH Asia 2013, ISBN 978-1-4503—2511, 2013

Meshless Interpolations

[Skal13b] Skala,V.: Projective Geometry, Duality and Precision of Computation in Computer Graphics, Visualization and Games, Tutorial Eurographics 2013, Girona, 2013

[Suss10] Süßmuth,J., Meyer,Q., Greiner,G.: Surface Reconstruction Based on Hierarchical Floating Radial Basis Functions. Comput. Graph. Forum 29(6): 1854-1864 (2010)

[Vasa09] Váša,L., Skala,V.: Combined Compression and Simplification of Dynamic 3D Meshes, The Journal of Comp. Animation and Virtual Worlds, Willey Interscience, Vol.20, No.4., pp.447-456, 2009

[Vasa10] Váša,L., Skala,V.: Cobra: Compression of the basis for PCA Represented Animations, Computer Graphics Forum, ISSN 0167-7055, Vol.28, No.6, pp.1529-1540, 2010

[Vasa11] Váša,L., Skala,V.: A perception correlated comparison method for dynamic meshes, IEEE Trans. on Visualization and Computer Graphics, ISSN 1077-2626, Vol.17, No.2, pp.220-230, 2011

[Vasa12] Váša,L., Rus,J.: Dihedral Angle Mesh Error: a fast perception correlated distortion measure for fixed connectivity triangle meshes, Computer Graphics Forum, Vol. 31(5), 2012.

[Wend10] Wendland,H.: Scattered Data Approximation, Cambridge University Press, , 2010

[Wrig03] Wright,G.B.: G. B. Wright. Radial Basis Function Interpolation: Numerical and Analytical Developments. University of Colorado, Boulder. PhD Thesis, 2003

[Zapl09] Zapletal,J., Vanecek,P., Skala,V.: RBF-based Image Restoration Utilising Auxiliary Points, CGI 2009 proceedings, ACM ISBN 978-60558-687-8, pp.39-44, 2009

Meshless Interpolations

Join our research activities in meshless methods at

Meshfree methods workshop

to be held as a part of the:

23rd WSCG International Conference on Computer Graphics,
Visualization and Computer Vision
(http://www.wscg.cz or http://www.wscg.eu)

Abstract submission: by May 12, 2015

upto 2 pages of the WSCG format sent via @mail
(accepted will appear in the WSCG 2015 abstracts proceedings)

Final paper submission by July 30, 2015 – accepted and presented

papers will be published in proceedings with ISBN
Will be sent for indexing to Scopus, ISI and other;
Selected papers will be published in the Journal of WSCG

Meshless Interpolations

Questions

???

Contact

Vaclav Skala
c/o University of West Bohemia
Plzen, Czech Republic

http://www.VaclavSkala.eu
or
skala@kiv.zcu.cz **subject: "meshless"**

Supported by the Ministry of Education of the Czech Republic, projects LH12181 and LG13047