Interpolation and Filtering

Data is often discretized in time and/or space.
We have only a finite number of sample points, i.e., the continuous signal is only known at few points (data points). But in general data is also needed between this points. By interpolation we obtain a representation that matches the function at the data points. An evaluation at any other point is possible. We can reconstruct the signal at points that are not sampled. For this reconstruction some assumptions are necessary. Often we arrogate smooth functions.

1 Voronoi Diagrams and Delaunay Triangulation

Given are irregularly distributed points without connectivity information. The problem is to obtain connectivity to find a "good" triangulation.
For a set of points there are many possible triangulations. A measure for the quality of a triangulation is the aspect ratio of the so-defined triangles. Long and thin triangles are to be avoided.

Scattered data triangulation for 2D:

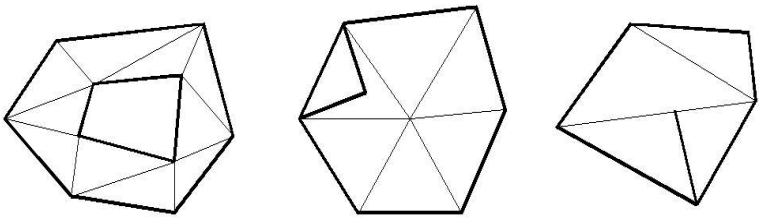
A triangulation of a set of data points $S = s_0, s_1, \ldots, s_m \in \mathbb{R}^2$ consists of

- Vertices (0D) = S
- Edges (1D) connecting two vertices
- Faces (2D) connecting three vertices

A triangulation must satisfy the following criteria:

- $\cup faces = conv(S)$, i.e. the union of all faces including the boundary is the convex hull of all vertices.
- The intersection of two triangles is either empty, or a common vertex, or a common edge, or a common face (tetrahedra).

The following triangulations are not valid.



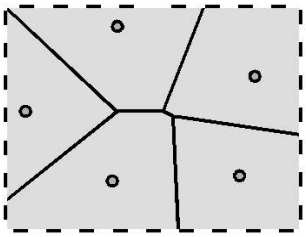
Non valid Triangulations

The triangulation a) is not valid because it “contains a hole”. In b) two faces overlap. The triangulation c) contains two vertices which form a “T” (T-vertices).
To get a triangulation from scattered data the Delaunay Triangulation, which is tightly connected to the Voronoi diagram, can be used.

1.1 The Voronoi diagram

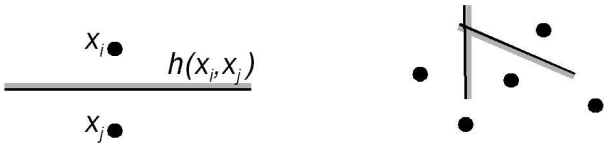
Given is a set of points $X = \{x_1, \ldots, x_n\}$ from $\mathbb{R}^d$ and a distance function *dist(x,y)*.

The Voronoi diagram *Vor(X)* contains for each point x_i a cell $V(x_i)$ with
$V(x_i)=\{x|dist(x,x_i)<dist(x,x_j)\forall\ j\neq i\}$
For each sample every point within a Voronoi region is closer to it than to every other sample.



A voronoi diagram

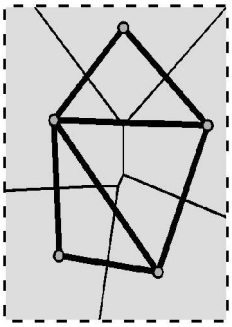
For a point x_i the corresponding Voronoi cell is given by the intersection of all half spaces $h(x_i,x_j), j\neq i$:
$V(x_i)=\cap_{j\neq i}h(x_i,x_j)$
$h(x_i,x_j)$ is seperated by the perpendicular bisector between x_i and x_j . $h(x_i,x_j)$ contains x_i .
Voronoi cells are convex.



Construction of voronoi cells

1.1.1 The Delaunay triangulation

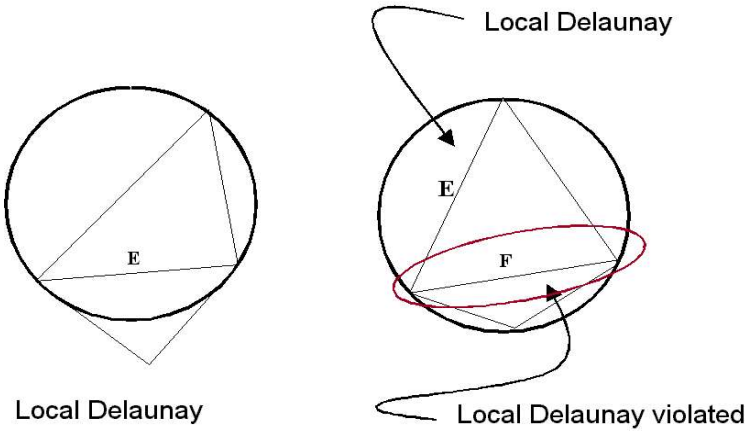
The Delaunay graph *Del(X)* is the geometric dual of the Voronoi diagram *Vor(X)*. The points in X are nodes. Two nodes x_i and x_j are connected if the Voronoi cells $V(x_i)$ and $V(X_j)$ share the same edge. Delaunay cells are convex.
The Delaunay triangulation is the triangulation of the Delaunay graph.



A Delaunay graph

The Delaunay triangulation in 2D:
Three points x_i,x_j,x_k in X belong to a face from *Del(X)* if no further point lies inside the circle around x_i,x_j,x_k .
Two points x_i,x_j form an edge if there is a circle around x_i,x_j that does not contain a third point from X.
For each triangle the circumcircle does not contain any other sample. The smallest angle and the ratio of $\frac{radius\ of\ incircle}{radius\ of\ circumcircle}$ is maximized. The triangulation is unique

(independent of the order of samples) for all but some very specific cases.

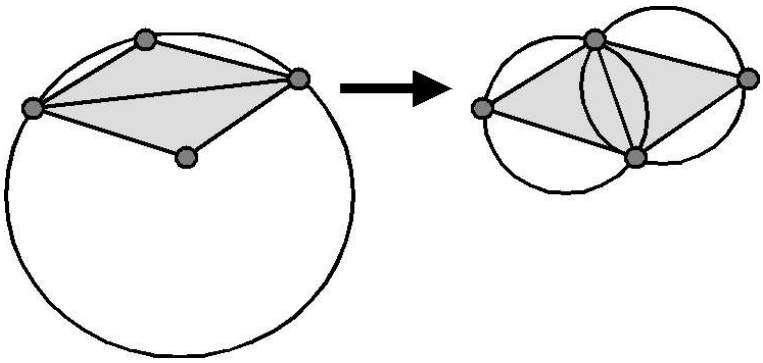


The local Delaunay property

1.1.1.1 Algorithms for Delaunay triangulations

- Edge flip algorithm

```
find an initial (valid) triangulation
find all edges where local Delaunay property is violated
mark these edges and push them onto the stack
while (stack not empty)
        pop edge from stack
        if (edge does not satisfy Delaunay property)
                flip this edge
                flip all adjacent edges for which the Delaunay
property is violated due to the flip
```



The edge flip algorithm

- Plane-sweep algorithm for finding an initial triangulation

In this algorithm an imaginary vertical sweepline passes from left to right.
As the sweepline moves:

Problem has been solved for the data to the left of the sweepline
Is currently being solved for the data at or near the sweepline and
Is going to be solved sometime later for the data to right of the sweep-line

This reduces a problem in 2D space to a series of problems in 1D space.

```
sort points from left to right
construct initial triangle using first three vertices
for i=4 to n do
        use last inserted p_{i-1} as starting point
        walk counterclockwise along convex polygon (hull) of
triangulation until the tangent
points,inserting edges between p_i and polygon points
```

```
        walk clockwise along convex polygon of triangulation until
the tangent points, inserting
        edges between p_i and polygon points
        update convex hull
endfor
```

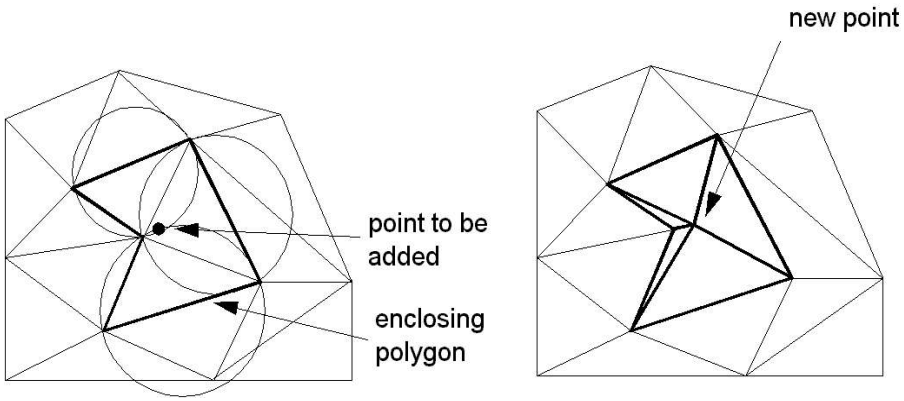
- Bowyer-Watson algorithm

The Bowyer-Watson algorithm builds the Delaunay triangulation from scattered points in one pass.

[Watson-1981-CDD]
[Bowyer-1981-CDT]

The idea of this algorithm is the incremental insertion of points into the triangulation:

- Start with initial triangulation which covers the domain (e.g. two triangles of bounding box)
- Incremental insertion of points into the triangulation
- All triangles whose circumcircles contain the inserted point are removed
- The resulting cavity is triangulated by linking the inserted point to all vertices of the cavity boundary
- The cavity is star-shaped: Edges from the location of the newly inserted point



The Bowyer Watson algorithm

The algorithm:

```
determine the super triangle that encompasses all vertices
add super triangle vertices to the end of the vertex list
add the super triangle to the triangle list
for (each point in the vertex list)
        calculate the triangle circumcircle center and radius
insert new point
        if (new point lies in a circumcircle)
                add the three triangle edges to the edge buffer
                remove the triangle from the triangle list
delete multiple specified edges from the edge buffer, which leaves
the edges of the enclosing polygon
add all triangles formed of the point and the enclosing polygon
remove all triangles from the triangulation that use the super
triangle vertices and remove their vertices from the vertex list
```

The following Applet visualizes the Voronoi diagram and the Delaunay triangulation:

DelaunayApplet/DelaunayApplet.html

Other techniques, e.g. Radial sweep, Intersecting halfspaces, Divide and conquer (merge-based or split-based), exist but are not discussed here.

1.2 Univariate Interpolation

Univariate interpolation means the interpolation for one variable.

1.2.1 Taylor Interpolation

For the taylor interpolation we use the basis functions $m_i = x^i \, with \, i \in \mathbb{N}_0$ (monom basis)
$\boldsymbol{P^m} = \{1, x, x^2, \ldots, x^m\}$ is an *m+1*-dimensional vector space of all polynomials with maximum degree *m*.

The task is to find coefficients $c_i \, with \, f = \sum_i c_i \cdot x^i$. This is the general approach, for x_i we can use any other basis function, but if we use the monom basis the interpolation problem can be solved with the Vandermond matrix.

The samples are represented by: $f(x_j) = f_j \, \forall \, j = 1 \ldots n$

The interpolation problem is given by: $\boldsymbol{V} \cdot \boldsymbol{c} = \boldsymbol{f}$ with the Vandermond matrix $\boldsymbol{V_{ij}} = x_i^{j-1}$

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

Properties of the Taylor interpolation:

- Unique solution
- Numerical problems / inaccuracies
- Complete system has to be solved again if a single value is changed
- Not intuitive

Example:

Given are 3 samples: f(1)=2, f(2)=5, f(4)=3

This leads to the following linear system of equations.

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 2 \\ 1 & 2 & 4 & 5 \\ 1 & 4 & 16 & 3 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 1 & 2 \\ 0 & 1 & 3 & 3 \\ 0 & 0 & 6 & -8 \end{array}\right)$$

$$\Rightarrow c_1 = -\frac{11}{3} \;\; , \;\; c_2 = 7 \;\; , \;\; c_3 = -\frac{4}{3}$$

The interpolated function: $f(x) = -\frac{4}{3}x^2 + 7x - \frac{11}{3}$

1.2.2 Generic interpolation problem

Given are *n* sampled points $X = \{x_i\} \subseteq \Omega \subseteq \mathbb{R}^d$ with function values f_i .

The *n*-dimensional function space $\Phi_n^d(\Omega)$ has the basis $\{\oint_{i=1\ldots n}\}$

We search coefficients $c_i \, with \, f(\lambda) = \sum_i c_i \cdot \oint_i (x)$

The samples are represented by: $f(x_j) = f_j \, \forall \, j = 1 \ldots n$

We have to solve the linear system of equations $M \cdot \boldsymbol{c} = \boldsymbol{f} \, with \, M_{ji} = \oint_i (x_j), \boldsymbol{c_i} = c_i, \boldsymbol{f_j} = f_j$

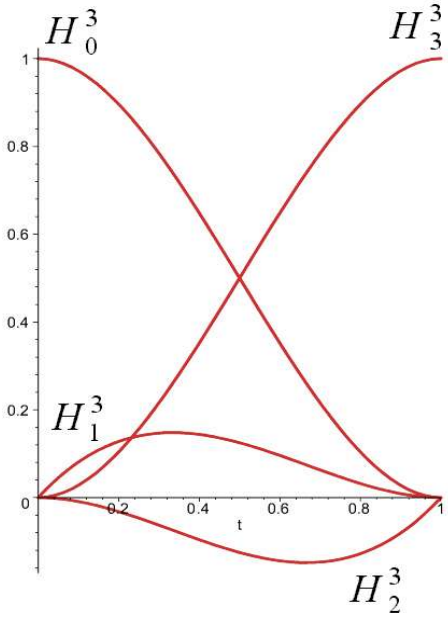
Note: The number of points *n* determines dimension of vector space (= degree of polynomials)

Other basis functions result in other interpolations schemes:

- Lagrange interpolation
- Newton interpolation
- Bernstein basis: Bezier curves (approximation)
- Hermite basis

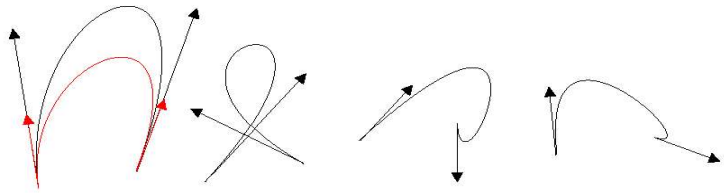
1.2.3 Cubic Hermite polynomials H

The coefficients describe the end points and the tangent vectors at the end points.



Hermite polynomials

$$H_0^3(t) = (1-t)^2(1+2t)$$
$$H_1^3(t) = t(1-t)^2$$
$$H_2^3(t) = -t^2(1-t)$$
$$H_3^3(t) = (3-2t)t^2$$

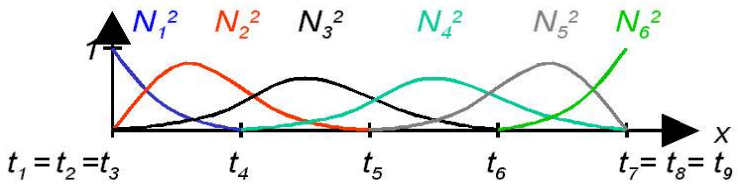


Example for Hermite interpolation

The problem of this approach is the coupling of the number of samples *n* and the degree of polynomials.
Solution: Spline interpolation

1.2.4 Spline interpolation

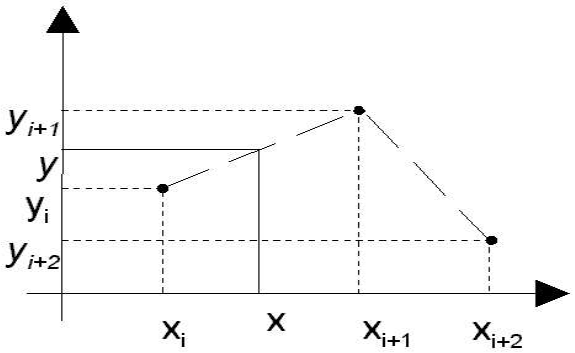
- Smooth piecewise polynomial function
- Continuity / smoothness at segment boundaries!



B-Spline basis functions.

1.2.5 Piecewise linear interpolation

This is the most simple approach (except for nearest-neighbor sampling). The main advantage is that it is fast to compute. The piecewise linear interpolation is often used in visualization applications.
Given are data points $(x_0, y_0), \ldots, (x_n, y_n)$



Piecewise linear interpolation

For any point x with $x_i \leqslant x \leqslant x_{i+1}$ evaluate $f(x) = (1-u)\, y_i + u y_{i+1}$ where $u = \frac{(x - x_i)}{(x_{i+1} - x_i)} \in [0,1]$.

x_i, x_{i+1}

1.3 Differentiation on grids

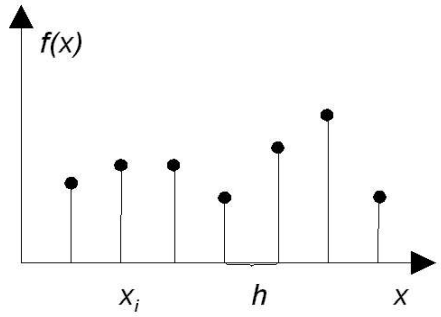
First Approach
Idea: Replace differential by “finite differences”.

Note that approximating the derivate by $f'(x) = \frac{df}{dx} \to \frac{\Delta f}{\Delta x}$ causes subtractive cancellation and large rounding errors for small *h*.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Second Approach
Approximate / interpolate (locally) by differentiable function and differentiate this function.

1.3.1 Finite differences on uniform grids with grid size h (1D case)



Finite differences on uniform grids with grid size h

- Forward differences: $f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{h}$
- Backward differences: $f'(x_i)=\frac{f(x_i)-f(x_{i-1})}{h}$
- Central differences: $f'(x_i)=\frac{f(x_{i+1})-f(x_{i-1})}{2}h$

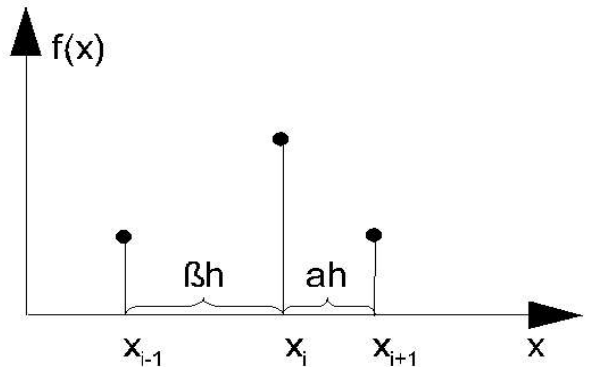
Error estimation:
- Forward / backward differences are first order.
- Central differences are second order.

1.3.2 Finite differences on non-uniform rectilinear grids

Forward and backward differences as for uniform grids with

$x_{i+1}-x_i=\alpha h$

$x_i-x_{i-1}=\beta h$



Finite differences on non-uniform grids

αh βh

The central differences are given by the Taylor expansion around the point x_i .

$$f(x_{i+1})=f(x_i)+\alpha h f'(x_i)+\frac{(\alpha h)^2}{2}f''(x_i)+\ldots$$

$$f(x_{i-1})=f(x_i)-\beta h f'(x_i)+\frac{(\beta h)^2}{2}f''(x_i)+\ldots$$

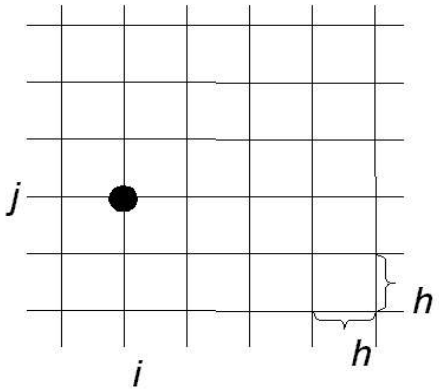
$$\Rightarrow \frac{1}{\alpha^2}(f(x_{i+1})-f(x_i))-\frac{1}{\beta^2}(f(x_{i-1})-f(x_i))=\frac{h}{\alpha}f'(x_i)+\frac{h}{\beta}f'(x_i)+O(h^3)$$

The final approximation of the derivate:

$$f'(x_i)=\frac{1}{h(\alpha+\beta)}\left(\frac{\beta}{\alpha}f(x_{i+1})-\frac{\alpha}{\beta}f(x_{i-1})+\frac{\alpha^2-\beta^2}{\alpha\beta}f(x_i)\right)$$

1.3.3 2D or 3D uniform or rectangular grids

The differentiation can be computed along each coordinate axis in the same way as in the univariate case.



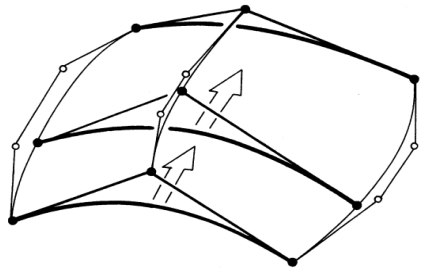
2D uniform grid

The gradient in a uniform grid is given by:

$$\textbf{\textit{grad}}\, f\begin{pmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\\ \frac{\partial f}{\partial z}\end{pmatrix}=\begin{pmatrix}\frac{f_{i+1,j,k}-f_{i-1,j,k}}{2}h\\ \frac{f_{i,j+1,k}-f_{i,j-1,k}}{2}h\\ \frac{f_{i,j,k+1}-f_{i,j,k-1}}{2}h\end{pmatrix}$$

1.4 Interpolation on grids

For manifolds with more than 1D we use a combination of several univariate interpolations.
Example for a 2D surface:



A 2D surface, with 3 basis functions in x-direction and 4 basis functions in y-direction.

Given are $n \cdot m$ values f_{jl} with $j=1\ldots n$ and $l=1\ldots m$ at points $X\times Y=(x_1,\ldots,x_n)\times(y_1,\ldots,y_m)$
n univariate basis functions $\xi_j(x) on X$ and m univariate basis functions $\psi_l(y) on Y$ are combined to $n \cdot m$ basis functions on $X\times Y$:

$\Phi_{ij}(x,y)=\xi_j(x)\cdot\psi_l(y)$

The tensor product is: $f(x,y)=\sum_{i=1,j=1}^{n,m}\Phi_{ij}(x,y)c_{ij}$

This means that we gain one bivariate basis function with two variables out of two univariate basis functions with one variable. The task is to solve a linear system of equations for the unknown coefficients c_{ij} .
An extension to *k* dimension is done in the same way.

Example:

Given are 4 samples: f(0,0)=2, f(0,1)=0.5, f(1,0)=3, f(1,1)=1

We choose the monom basis for ξ_j and ψ_l : $\xi_1=1,\ \xi_2=x$, $\psi_l=1,\ \psi_2=x$

$\Rightarrow\Phi_{11}=1,\ \Phi_{12}=x,\ \Phi_{21}=y,\ \Phi_{22}=xy$

This leads to the following system of equations:

$2=c_{11}$

$\frac{1}{2}=c_{11}+c_{21}$

$3=c_{11}+c_{12}$

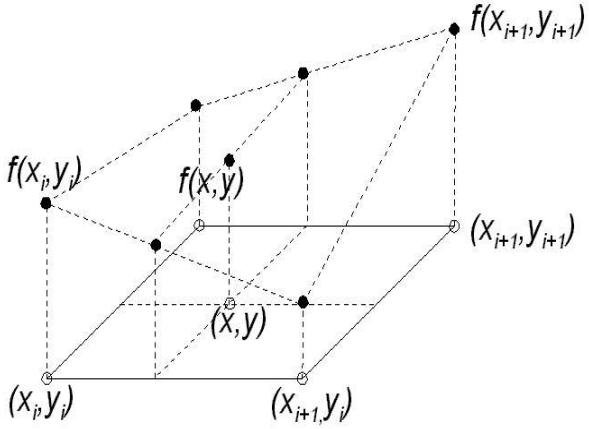
$1=c_{11}+c_{12}+c_{21}+c_{22}$

$\Rightarrow c_{11}=2,\ c_{12}=1,\ c_{21}=-\frac{3}{2},\ c_{22}=-\frac{1}{2}$

The interpolated function is given by: $f(x,y)=-\frac{1}{2}xy+x-\frac{3}{2}y+2$

1.4.1 Bilinear interpolation on a rectangle

- Tensor product for two linear interpolations
- 2D local interpolation in a cell
- Known solution of the linear system of equations for the coefficients c_{ij}
- Four data points $(x_i,y_j),\ldots,(x_{i+1},y_{j+1})$ with scalar values $f_{kl}=f(x_k,y_l)$
- Bilinear interpolation of points (x,y) with $x_i\leqslant x\leqslant x_{i+1}$ and $y_i\leqslant y\leqslant y_{i+1}$



Bilinear interpolation on a rectangle

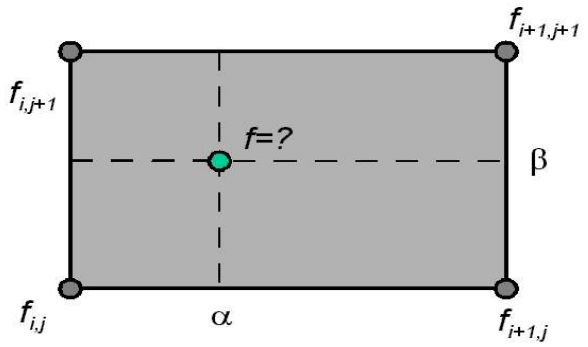
$f(x,y)=(1-\beta)[(1-\alpha)f_{i,j}+\alpha f_{i+1,j}]+\beta[(1-\alpha)f_{i,j+1}+\alpha f_{i+1,j+1}]\ =\ (1-\beta)f_j+\beta f_{j+1}$

with:

$$f_j=(1-\alpha)f_{i,j}+\alpha f_{i+1,j}$$
$$f_{j+1}=(1-\alpha)f_{i,j+1}+\alpha f_{i+1,j+1}$$

and local coordinates:

$$\alpha=\frac{x-x_i}{x_{i+1}-x_i}\quad,\quad \beta=\frac{y-y_i}{y_{i+1}-y_i}\quad,\quad \alpha,\beta\in[0,1]$$

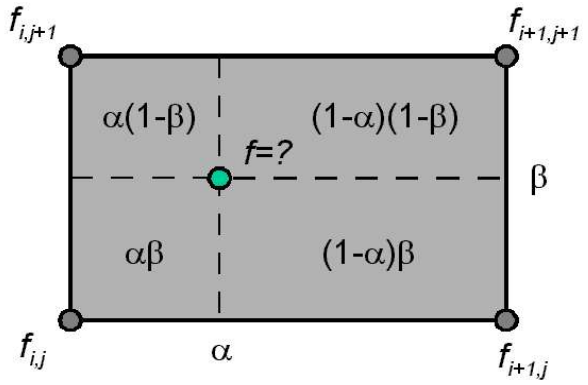


Bilinear interpolation on a rectangle

The equation above for f(x,y) can be rewritten as follows:

$$f(x,y)=(1-\alpha)(1-\beta)f_{i,j}+\alpha(1-\beta)f_{i+1,j}+(1-\alpha)\beta f_{i,j+1}+\alpha\beta f_{i+1,j+1}$$

The point to be interpolated divides the rectangle into 4 areas. The corresponding data value can be computed by weighting the 4 given data values with area of the corresponding opposite area.



Bilinear interpolation on a rectangle

Note: Bilinear interpolation is **not** linear. This is illustrated in the animation BilinearInterpolation.divx. You see a plane which is generated by interpolating between 4 points. In the animation you see how the shape of the plane changes when the points are moved.

Trilinear interpolation on a 3D uniform grid:

- Straightforward extension of bilinear interpolation
- Three local coordinates α,β,γ
- Known solution of the linear system of equations for the coefficients c_{ij}
- Trilinear interpolation is not linear!

Extension to higher order of continuity:

- Piecewise cubic interpolation in 1D
- Piecewise bicubic interpolation in 2D
- Piecewise tricubic interpolation in 3D
- Based on Hermite polynomials

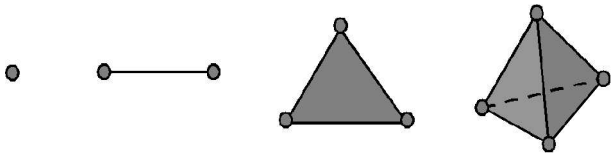
1.4.2 Interpolation on structured grids (triangle meshes etc.)

Some definitions:

> An affine combination **a** is a linear combination of n points x_i , $i \in \{1,\ldots,n\}$:
>
> $\boldsymbol{a} = \sum_{i=1}^{n} \alpha_i x_i$ where $0 \leqslant \alpha_i \leqslant 1, \forall i$ and $\sum_{i=1}^{n} \alpha_i = 1$

The α_i are called barycentric coordinates.
A set of points is called *affinely independent* if no point can be expressed as affine combination of the other points. The maximum size of a affinely independent set in $\mathbb{R}^d$ is d+1.
A simplex in $\mathbb{R}^d$ is the span of d+1 affinely independent points.
Examples:

- 0D : point
- 1D : line
- 2D : triangle
- 3D : tetrahedron



Simplexes

1.4.2.1 Barycentric interpolation on a simplex

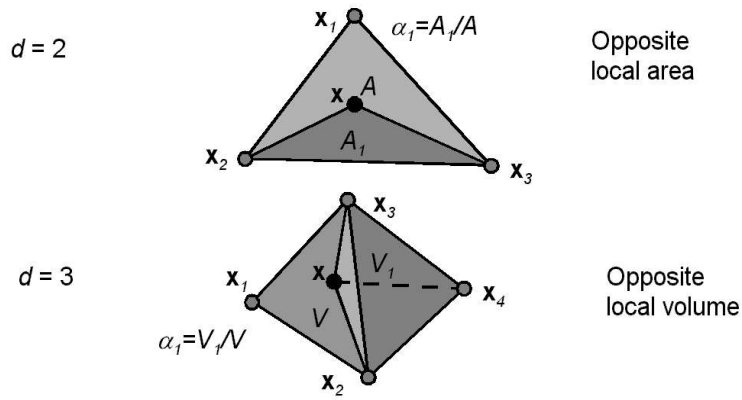
Given are d+1 points $\boldsymbol{x_i}$ with function values f_i .
Search coefficients α_i with $\boldsymbol{x} = \sum_i \alpha_i x_i$ and $\sum_i \alpha_i = 1$.
The function value at $\boldsymbol{x}$ is given by: $f = \sum_i \alpha_i x_i$.

1.4.2.2 Barycentric coordinates from area/volume considerations

$$\alpha_i = \frac{Vol(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots,x_{d+1})}{Vol(x_1,\ldots,x_{d+1})}$$

$$Vol(x_1,\ldots,x_{d+1}) = det\begin{pmatrix} x_1 & \ldots & x_{d+1} \\ 1 & \ldots & 1 \end{pmatrix}$$

Examples:



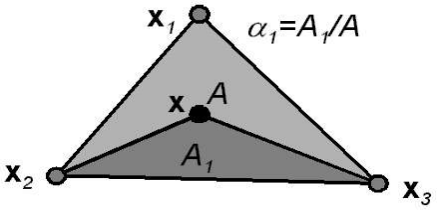
Barycentric coordinates from area/volume considerations for d=2 and d=3

Barycentric interpolation in a triangle:
Geometrically, barycentric coordinates are given by the ratios of the area of the whole triangle and the subtriangles defined by x and any two points of x_1, x_2, x_3 .

$$Vol(x_1, x_2, x_3) = det\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix} = \pm 2\, Area(\Delta(x_1, x_2, x_3))$$

$$\alpha_1 = \frac{Vol(x, x_2, x_3)}{Vol(x_1, x_2, x_3)}$$

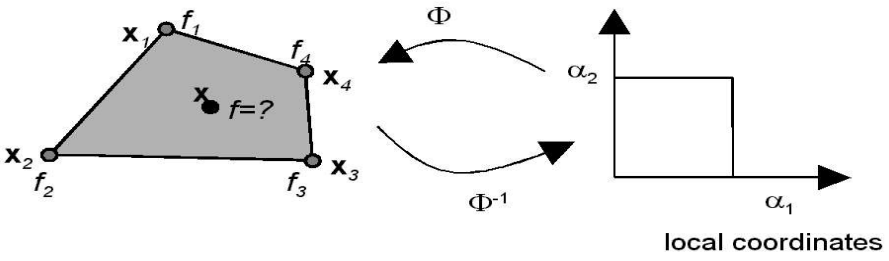
So we end up with $\boldsymbol{x} = \sum_i \alpha_i x_i$ and $\sum_i \alpha_i = 1$.



Barycentric interpolation in a triangle

1.4.2.3 Interpolation in a generic quadrilateral

The main application for this approach are curvilinear grids. The problem is to find a parameterization for arbitrary quadrilaterals.



Interpolation in a generic quadrilateral

The mapping Φ from rectangular domain to quadratic domains is known: Bilinear interpolation on a rectangle.

$$x_{12} = \alpha_1 \cdot x_1 + (1 - \alpha_1) \cdot x_2$$
$$x_{34} = \alpha_1 \cdot x_4 + (1 - \alpha_1) \cdot x_3$$
$$x = \alpha_2 \cdot x_{12} + (1 - \alpha_2) \cdot x_{34}$$

$\alpha_1 \in [0,1]$, $\alpha_2 \in [0,1]$

Computing the inverse of Φ is more complicated. There are two possibilties:

- Analytically solve quadratic system for α_1, α_2
- Numerical solution by Newton iteration

The final value is given by: $f = \alpha_2 \cdot (\alpha_1 \cdot f_1 + (1-\alpha_1) \cdot f_2) + (1-\alpha_2) \cdot (\alpha_1 f_4 + (1-\alpha_1) f_3)$

The *Jacobi matrix* $J(\Phi)$ is given by:

$$J(\Phi)_{ij} = \frac{\partial \Phi_i}{\partial \alpha_j}$$

$J(\Phi)_{ij}$ describes direction and speed of position changes of Φ when α_j are varied

Newton iteration:

start with seed points as start configuration, e.g. $\alpha_i = \frac{1}{2}$

while $(\left\| x - \Phi(\alpha_1, \alpha_2, \alpha_3) \right\| > \epsilon)$

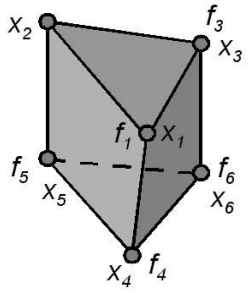
compute $J(\Phi(\alpha_1, \alpha_2, \alpha_3))$

transform $\boldsymbol{x}$ in coordinate system $J(\Phi)$:

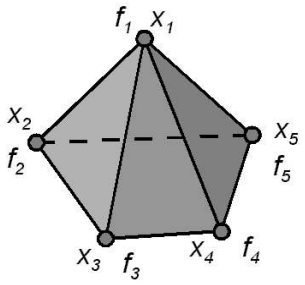
$$x_\alpha = J(\Phi(\alpha_1, \alpha_2, \alpha_3))^{-1} \cdot (x - \Phi(\alpha_1, \alpha_2, \alpha_3))$$

update $\alpha_i = \alpha_i + x_{\alpha,i}$

Other primitive cell types are possible. Examples:



Prism:
- twice barycentric
- once linear



Pyramid:
- bilinear on base face
- then linear

Other primitive cells

1.4.2.4 Inverse distance weighting

The *Shepard interpolation* [D. Shepard, A two-dimensional interpolating function for irregularly spaced data. Proc. ACM. nat. Conf., 517--524, 1968] was originally developed for scattered data.

Interpolated values: $f(x) = \sum_i \Phi_i(\boldsymbol{x}) f_i$

The sample points are the vertices of the cell.

Basis functions: $\phi_i(x) = \frac{\left\| x - x_i \right\|^{-p}}{\sum \left\| x - x_j \right\|^{-p}}$

Define values at sample points: $f(\boldsymbol{x_i}) := f_i = \lim_{x \to x_i} f(\boldsymbol{x})$

1.5 Interpolation without grids

1.5.1 Shepard interpolation

- Different exponents for inner and outer neighborhood (default: 2 in the inner neighborhood and 4 in the outer neighborhood)
- The neighborhood sizes determine how many points are included in inverse distance weighting
- The neighborhood size can be specified in terms of
  - Radius or
  - Number of points or
  - Combination of the two
- The neighborhood is not given explicitly (as opposed to inverse distance weighting on grids)

1.5.2 Radial basis functions

- *n* function values f_i given at *n* points $\boldsymbol{x_i}$
- Interpolant: $f(x)=\sum_{i=1}^{n}\lambda_i\phi(\|x-x_i\|)+\sum_{m=0}^{k}c_m p_m(x)$
- Univariate radial basis $\Phi(r)$
- Examples:
  - Polynomials r^v
  - Gaussians $\exp(r^{-2})$
- Polynomial basis $\{p_m\}$ for (*k*+1)-dimensional vector space
- Under-determined system: *n* equations for *n*+(*k*+1) unknowns
- Additional constraints (orthogonality / side conditions): $\sum_{i=1}^{n}\lambda_i p_m(x_i)=0 \quad \forall m=0\ldots k$
- Well-defined system of linear equations (vector / matrix notation):

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix}\begin{pmatrix} \lambda \\ c \end{pmatrix}=\begin{pmatrix} f \\ 0 \end{pmatrix}$$

$$A_{i,j}=\phi(\|x_i-x_j\|)$$

P: Polynomial basis

λ : Coefficients for radial function

f: Function values at sample points

c: Coefficients for polynomials

1.6 Filtering by Projection or Selection

Very often there is too much information to be visualized at once. The strategy is to reduce the displayed information by filtering. A popular approach is to reduce from $n\boldsymbol{d}m\boldsymbol{v}$ to $n'\boldsymbol{d}m'\boldsymbol{v}$, with $n'<n$ and / or $m'<m$.

The possible techniques are: projection, selection, and slicing. For these user input is needed.

Projection π :

- Functional description for both the
  - Domain and
  - Data values

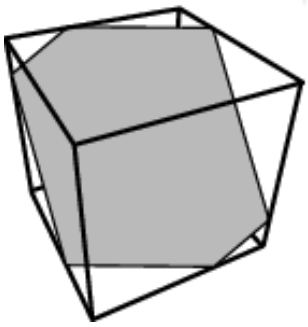
- Projection into subspaces
- Often a mapping to a sub set of the original values is chosen

Selection σ :

- Selection of data according to logical conditions (predicates)
- Example:
  - Height field $2\mathbf{d}1\mathbf{v}$ with data (x,y,h)
  - $D_\sigma = \{(x, y, h) | (x^2 + y^2 < 5\,km) \wedge (h > 1\,km)\}$

Slicing:

- Example: 2D cutting surface (slice) through a 3D volume



Slicing

1.7 Fourier Transform

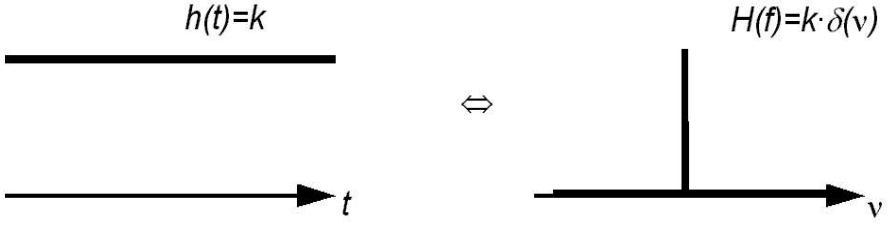
The fourier transformation is often used for image processing, especially for filtering. Here, the image is converted from the spatial domain to the frequency domain by using the fourier transformation. In the frequency domain the image is multiplied by a filter (e.g. Gauss-filter, box filter, etc.). Afterwards the image will be transformated back to the spatial domain.

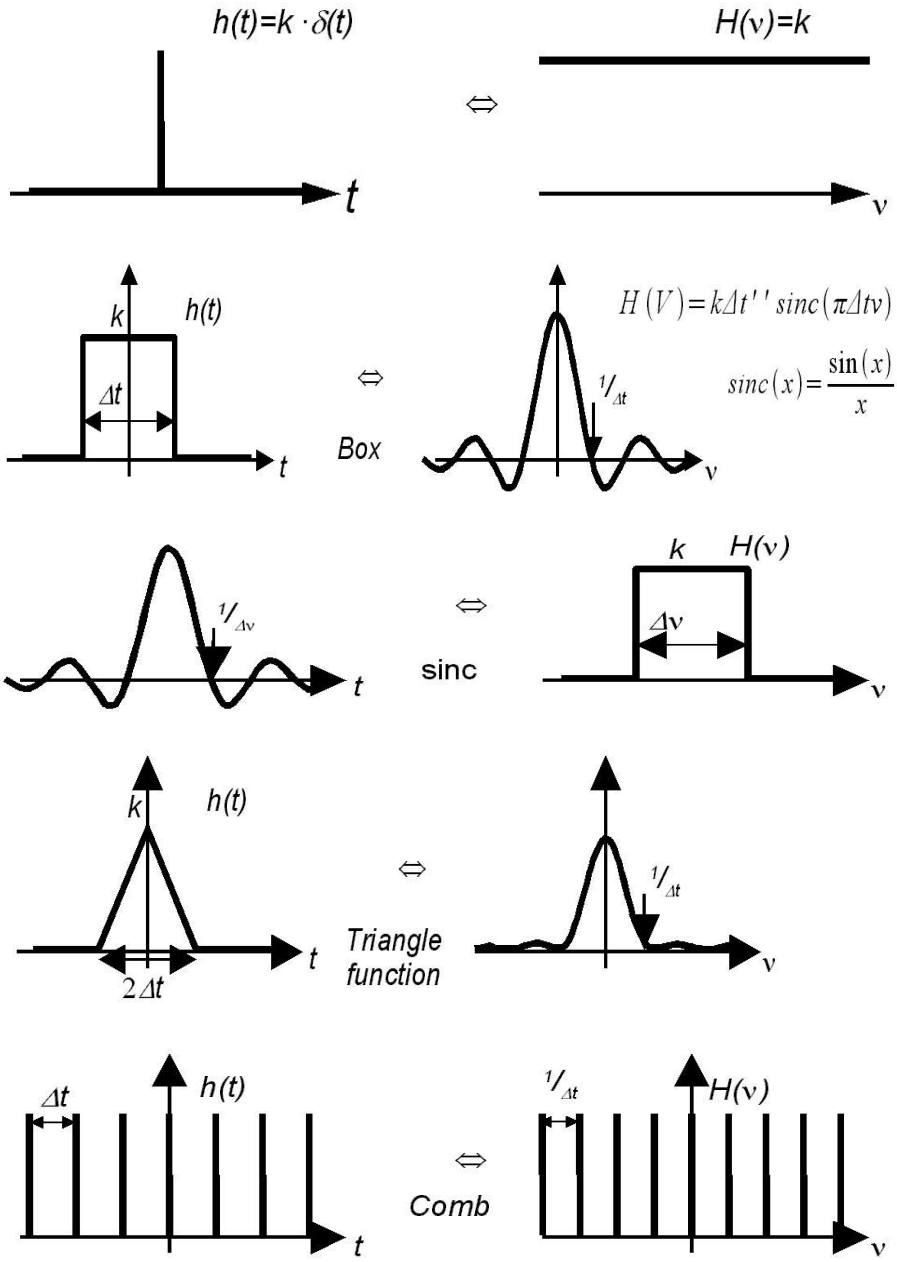
In the spatial domain a signal $h(t)$ is given by the amplitude value as a function of time. The analogous representation $H(v)$ in the frequency domain is a function of the frequency v .
Via the fourier transform these two representations can be converted into each other.

- forward transform: $H(v) = \int_{-\infty}^{\infty} h(t) \mathrm{e}^{-2\pi i v t}\, dt$
- inverse transform: $h(t) = \int_{-\infty}^{\infty} H(v) \mathrm{e}^{2\pi i v t}\, dv$

The *convolution* is defined by: $(g * h)(t) = \int_{-\infty}^{\infty} g(\tau) \cdot h(t - \tau)\, d\tau$

The *convolution theorem*: $(g * h)(t) \Leftrightarrow G(v) \cdot H(v)$,i.e. a convolution in the time domain corresponds to a multiplication in the frequency domain.
Examples:





Examples for functions in the time domain and the corresponding functions in the frequency domain

In applications mostly discrete Fourier transform, which are based in a discrete signal, are used.

1.8 Sampled Signals

Assume that a signal $h(t)$ is *band limited* with frequencies smaller than B .
The so called *Nyquist frequency* is defined as $\nu_{Nyq}=2B$

The signal can be discretized with a constant step size $\Delta t=\frac{1}{\nu_{Nyq}}=\frac{1}{2B}$

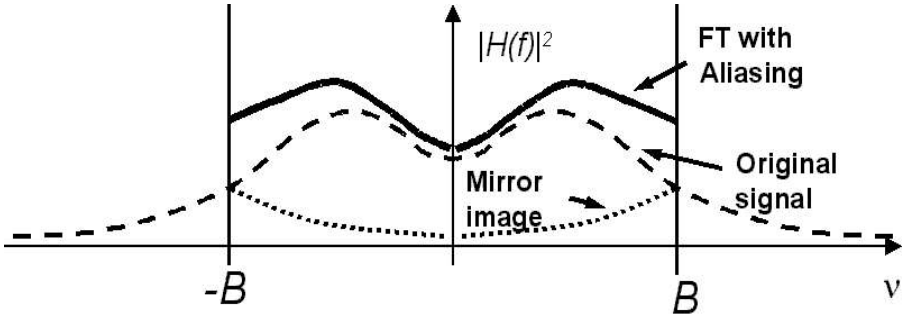
Sampled signal: $h_j=h(j\cdot\Delta t)$
If only a finite interval j=0..n-1 is used, periodicity is assumed.
Sampling Theorem (Shannon 1949):

If $H(f)=0$ for all $|\nu|>B=\frac{\nu_{Nyq}}{2}$, then $h(t)$ is uniquely given by the samples h_i :

$$h(t)=\sum_{j=0..n-1} h_j \cdot sinc(\pi \cdot v_{Nyq}(t-j\cdot\Delta t))$$

Issue 1: **Undersampling**

If h(t) has frequencies larger than $B=\frac{v_{Nyq}}{2}$, then h(t) cannot be reconstructed from the sampled values (Aliasing).

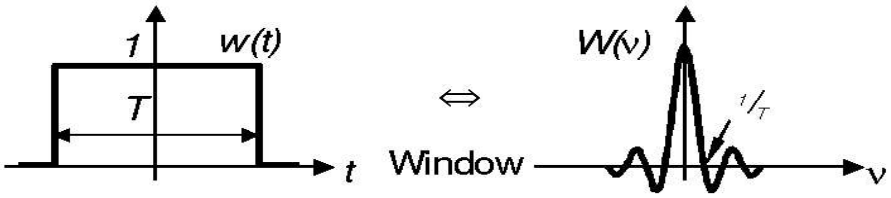


An undersampled signal

Issue 2: **Finite window size**

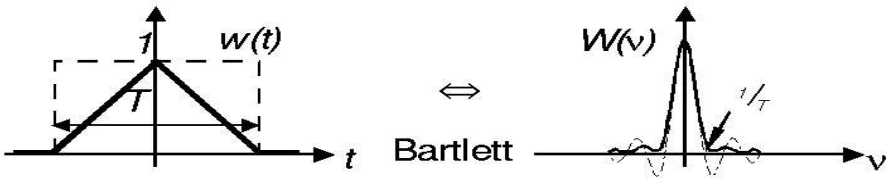
The Fourier Transform is theoretically defined for signals of infinite duration or for periodic signals. Often the signal h(t) is measured on a finite interval $\left[-\frac{T}{2},\frac{T}{2}\right]$ (without periodicity).

This can be considered as a multiplication with a window function: $h(t)\cdot \mathbf{1}_{\left[-\frac{T}{2},\frac{T}{2}\right]}(t)$. In the frequency space this means a convolution with a sinc() function.



Finite window size

A problem of a finite window size is the fact that the differences between the starting and the ending values of the segment produces a discontinuity which generates high-frequency spurious components. The use of a Bartlett window solves this problem.



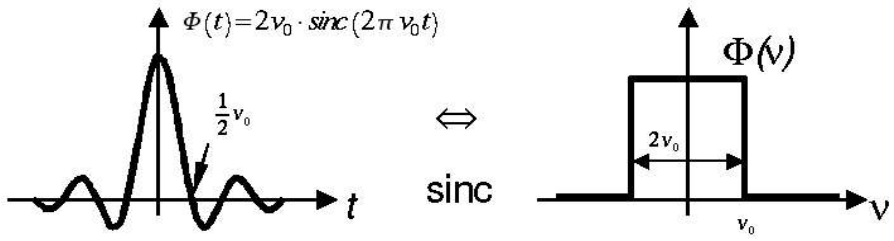
Bartlett window

1.9 Reconstruction and Frequency Filtering

Filter design is mostly based on the Fourier analysis.

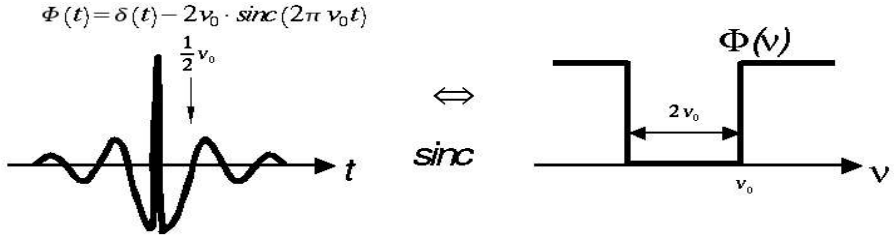
A *low pass filter* with limit frequency v_0 can be realized with

- a convolution with a sinc() function in coordinate space or
- a Fast Fourier Transformation (FFT), a multiplication with a box filter $\Phi(v)$ and then an inverse FFT



A low pass filter

A *high pass filter* emphasizes features, e.g. edges.



High pass filter

Reconstruction issues:
The measurements $m(t)$ of the original signal $s(t)$ are based on a point-spread function $p(t-t_i)$, not on the ideal delta function $\delta(t-t_i)$. A convolution in coordinate space corresponds to a multiplication in the frequency space:

$$m(t)=\int_{-\infty}^{\infty} p(t-\tau)s(\tau)d\tau \;\Leftrightarrow\; M(\nu)=P(\nu)S(\nu)$$

An Applet for Fourier Analysis can be found at: ***http://www.gris.uni-tuebingen.de/projects/grdev/applets/fourier/html/index.html***