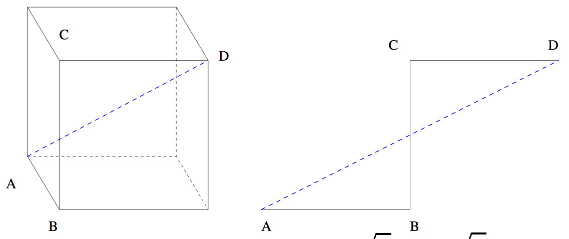
In [2]:
```
## settings
import matplotlib.pyplot as plt
import numpy as np
%matplotlib qt5
import scipy, scipy.stats
plt.rcParams['figure.figsize'] = (8.0, 4.0)
from ipywidgets import interact, fixed
%matplotlib inline
```

\usepackage*amssymb*

5.3 Multidimensional Scaling

Goal

- Construction of a low-dimensional 'projection' $\mathbb{R}^D \mapsto \mathbb{R}^d$ which is trying to maintain distances as good as possible ('abstandstreu')
- Example:



  - Assume that the edge length is 1: the diagonal then is $\sqrt{3}$, but $\sqrt{5}$ in the non-distance-preserving projection.

Given:

- $\vec{x}_1 \ldots \vec{x}_n \in \mathbb{R}^D$ given data points
  - $D =$ dimension of the data space
- $\vec{\phi}(\vec{x}_1) \ldots \vec{\phi}(\vec{x}_n) \in \mathbb{R}^d$ wanted low-dimensional image
  - $d =$ dimension of the projection space

Goal:

- to maximally preserve distances, i.e.

$$\|\vec{\phi}(\vec{x}_i) - \vec{\phi}(\vec{x}_j)\|_d \overset{!}{=} \|\vec{x}_i - \vec{x}_j\|_D \quad \forall\, i,j$$

5.3.1. Sammon-Mapping

- results from the minimization of a cost function E for distance distortions
- source: IEEE Trans.Comp. C-18, May 1969 401-409.
- Let

$$D_{ij} = \|\vec{x}_i - \vec{x}_j\|_D = \text{ distance in the high-dimensional space}$$
$$d_{ij} = \|\vec{\phi}(\vec{x}_i) - \vec{\phi}(\vec{x}_j)\|_d = \text{distance in the low-dimensional space}$$

Then the Sammon-Mapping bases on the following approach for a cost function E

$$E[\phi] = \frac{1}{\sum_{i<j} D_{ij}} \cdot \sum_{i<j} \frac{(d_{ij} - D_{ij})^2}{D_{ij}}$$

- the scaling factor serves only for scaling invariance
  - it is a constant for a constant data set
- the denominator downweighs the error for large data point distances:
  - distance deviations shall here be less culpable
- E is undefined for points i, j with $D_{ij} = 0$: such points need to be eliminated beforehand
  - alternatively one could protect the denominator from divergence by adding a small ϵ
- Minimization can for instance be done by gradient descent w.r.t. to ϕ-parameters
- This is usually a very high-dimensional minimization problem, the result of which is facilitated by good initial values
- Useful initial conditions are for instance the projection of $\vec{x}_i$ on the linear subspace spanned by the d eigenvectors corresponding to the largest eigenvalues of the der covariance matrix C.
- A more flexible choice for $\{\Phi(\vec{x}_i)\}_i$ = table of target positions $(x_1^p, y_1^p, \ldots, x_N^p, y_n^p)$
  - (p indicates that this is the coordinate in projection space)

5.3.2. Sammon-Mapping with a target table of coordinates

The target points $\Phi(\vec{x}_k) = \vec{v}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ are here simply provided via a look-up table.

The cost function in turn can be rewritten as

$$E[\Phi] = \frac{1}{\sum_{ij} D_{ij}} \frac{1}{2} \sum_i \sum_{j \neq i} \frac{(x_i - x_j)^2 + (y_i - y_j)^2 + D_{ij}^2 - 2D_{ij}\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{D_{ij}}$$

With this, the gradient (here for the example of the x-component, y in complete analogy) becomes:

$$\frac{\partial E}{\partial x_i} = \frac{1}{\sum_{ij} D_{ij}} \frac{1}{2} \sum_{j \neq i} \frac{1}{D_{ij}} (2(x_i - x_j) - 2D_{ij} \frac{1}{2d_{ij}} 2(x_i - x_j))$$

- here the chain rule is used for derivation
- the sum over i disappears

Further simplification results in

$$\frac{\partial E}{\partial x_i} = \frac{1}{\sum\limits_{ij} D_{ij}} \left[\sum_j \left(\frac{1}{D_{ij}} - \frac{1}{d_{ij}} \right) (x_i - x_j) \right] = \frac{1}{\sum\limits_{ij} D_{ij}} \left[\sum_j a_{ij} (x_i - x_j) \right]$$

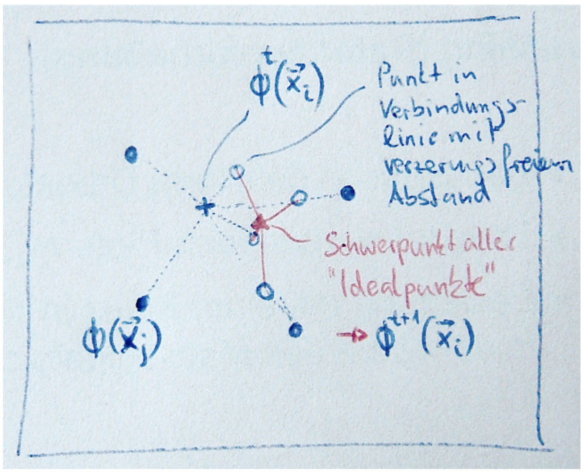
with

$$a_{ij} = \left(\frac{1}{D_{ij}} - \frac{1}{d_{ij}} \right) = \frac{d_{ij} - D_{ij}}{D_{ij} d_{ij}}.$$

The coefficients are positive (resp. negative), if the projection distance is larger (resp. smaller) than the original distance of points ii and j.

From this the update step becomes:

$$\Delta\overrightarrow{v_k} = -\eta \nabla_{\vec{v_k}} E = -\frac{\eta}{\sum\limits_{j,i} D_{ij}} \sum_j a_{kj} (\vec{v}_k - \vec{v}_j)$$


$\phi^t(\vec{x}_i)$
Punkt in Verbindungslinie mit verzerrungsfreiem Abstand
Schwerpunkt aller "Idealpunkte"
$\phi(\vec{x}_j)$
$\phi^{t+1}(\vec{x}_i)$

- each point connection between $\vec{v}_i$ und $\vec{v}_j$ effects as a spring with a forcefree length of D_{ij}
- the spring excerts a force proportional to the distance difference $D_{ij} - d_{ij}$ towards the equilibrium.
- if only a single point $\vec{v}_i$ is allowed to move, the sum of all spring forces causes the point to oscillate around an ideal point which minimizes the overall distortion.
- friction is usually added so that the mass moves right into the potential trough

5.3.2. Variant: Non-metric multidimensional Scaling

Idea: Instead of a mapping that is true for distances, we aim at best possible maintenance of distance rank order

- i.e. we don't look at the absolute distance but its rank in a sorted list, similar to the difference between linear correlation and rank correlation
- method is known as ordinal method of MDS
$$D_{ij} < D_{kl} \Leftrightarrow d_{ij} < d_{kl} \quad \forall\, i, k, l$$
- Note that equality is not necessarily required

Advantage: higher robustness w.r.t. the choice of the distance metrics D.

- Solving the above equation can be done by minimizing a suitable cost function
$$\text{stress}^2 = E(D, \phi_1 \ldots \phi_n; \Theta) := \frac{\sum_{ij} (\Theta(D_{ij}) - d_{ij})^2}{\sum_{ij} d_{ij}^2}$$
- Minimization is done w.r.t. the image points $\phi_1 \ldots \phi_N$ and the function $\Theta(\cdot)$, from which we demand that it is monotonous increasing.
- The introduction of $\Theta(\cdot)$ results in the property that only the rank order of given distances D_{ij} is relevant for the optimal solution.
- it can be shown that the minimum is obtained for a step function $\Theta(\cdot)$ with maximal $\binom{n}{2}$ steps, where n is the number of data points.
  - so that $\Theta(\cdot)$ can be parameterized accordingly.

See for instance: Ripley: Pattern Recognition and Neural Networks,