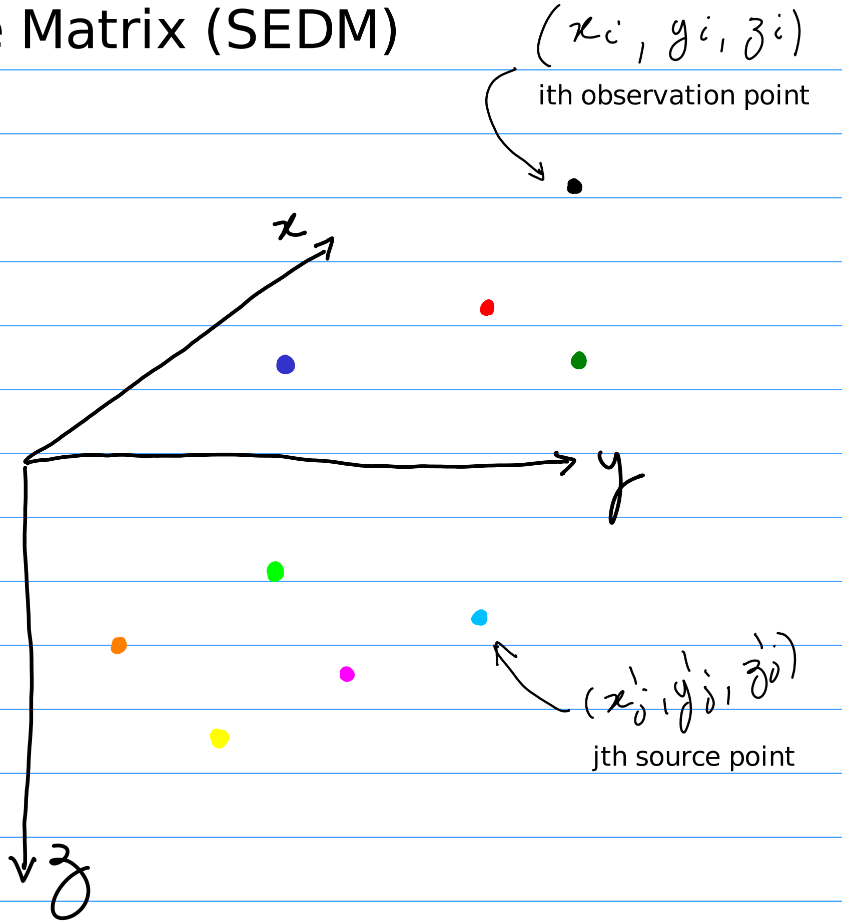
Squared Euclidean Distance Matrix (SEDM)



Observation points matrix

$$P = \begin{bmatrix} x_0 & x_1 & \cdots & x_{N-1} \\ y_0 & y_1 & \cdots & y_{N-1} \\ z_0 & z_1 & \cdots & z_{N-1} \end{bmatrix}_{3 \times N}$$

Source points matrix

$$S = \begin{bmatrix} x'_0 & x'_1 & \cdots & x'_{M-1} \\ y'_0 & y'_1 & \cdots & y'_{M-1} \\ z'_0 & z'_1 & \cdots & z'_{M-1} \end{bmatrix}_{3 \times M}$$

Squared Euclidean Distance Matrix (SEDM)

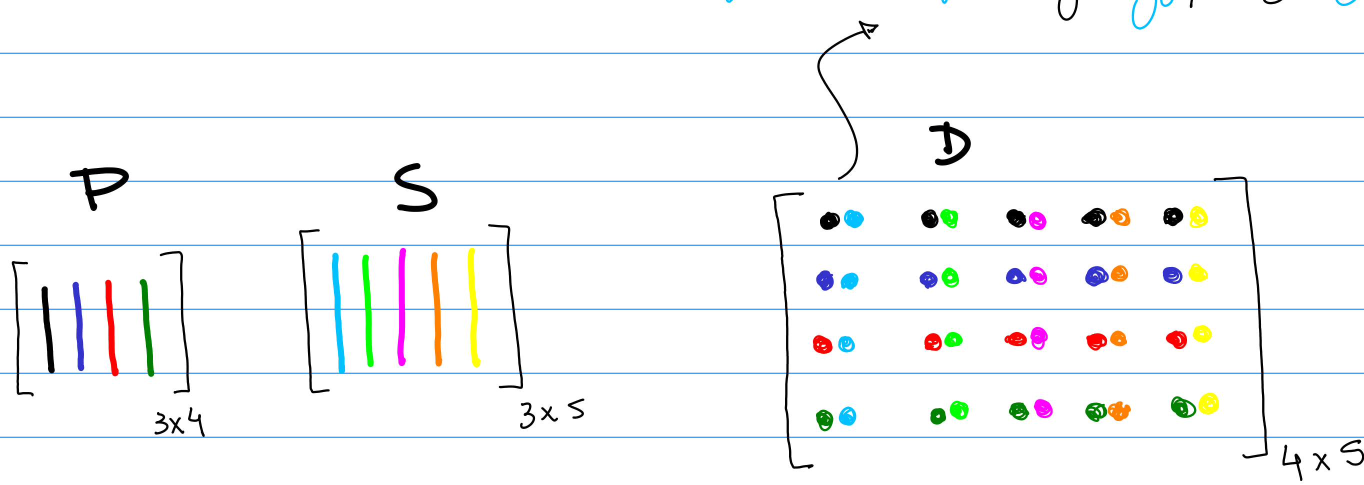
$$D = \begin{bmatrix} d^2_{00} & \cdots & d^2_{0\,M-1} \\ \vdots & & \vdots \\ d^2_{N-1\,0} & \cdots & d^2_{N-1\,M-1} \end{bmatrix}_{N \times M}$$

Squared Euclidean distance

$$d^2_{ij} = (x_i - x'_j)^2 + (y_i - y'_j)^2 + (z_i - z'_j)^2$$

$$i = 0, \ldots, N-1 \qquad j = 0, \ldots, M-1$$

$$d^2_{ij} = (x_i - x'_j)^2 + (y_i - y'_j)^2 + (z_i - z'_j)^2$$



$$d_{ij}^2 = (\mathbf{p}_i - \mathbf{s}_j)^T(\mathbf{p}_i - \mathbf{s}_j) =$$

$$= (x_i - x'_j)^2 + (y_i - y'_j)^2 + (z_i - z'_j)^2$$

$$= (x_i^2 - 2x_ix'_j + x'^2_j) + (y_i^2 - 2y_iy'_j + y'^2_j) + (z_i^2 - 2z_iz'_j + z'^2_j)$$

$$= (x_i^2 + y_i^2 + z_i^2) + (x'^2_j + y'^2_j + z'^2_j) - 2(x_ix'_j + y_iy'_j + z_iz'_j)$$

$$= \mathbf{p}_i^T\mathbf{p}_i + \mathbf{s}_j^T\mathbf{s}_j - 2\mathbf{p}_i^T\mathbf{s}_j$$

$$\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad \mathbf{s}_j = \begin{bmatrix} x'_j \\ y'_j \\ z'_j \end{bmatrix}$$

ith row of matrix $\mathbf{P}$

th row of matrix $\mathbf{S}$

constant values along rows

$$\mathbf{D}_1 = \begin{bmatrix} \mathbf{P}_0^T\mathbf{P}_0 & \cdots & \mathbf{P}_0^T\mathbf{P}_0 \\ \mathbf{P}_{N-1}^T\mathbf{P}_{N-1} & \cdots & \mathbf{P}_{N-1}^T\mathbf{P}_{N-1} \end{bmatrix}_{N\times M}$$

$$\mathbf{D}_2 = \begin{bmatrix} \mathbf{s}_0^T\mathbf{s}_0 & \mathbf{s}_{m-1}^T\mathbf{s}_{m-1} \\ \vdots & \vdots \\ \mathbf{s}_0^T\mathbf{s}_0 & \mathbf{s}_{m-1}^T\mathbf{s}_{m-1} \end{bmatrix}_{N\times M}$$

constant values along columns

$$\mathbf{D}_3 = 2\begin{bmatrix} \mathbf{P}_0^T\mathbf{s}_0 & \mathbf{P}_0^T\mathbf{s}_{m-1} \\ \mathbf{P}_{N-1}^T\mathbf{s}_0 & \mathbf{P}_{N-1}^T\mathbf{s}_{m-1} \end{bmatrix}_{N\times M} = 2\mathbf{P}^T\mathbf{S}$$

$$\boxed{\mathbf{D} = \mathbf{D}_1 + \mathbf{D}_2 - \mathbf{D}_3}$$

How to efficiently compute this with Numpy?

Matrix $\mathbf{D}_1$ can be rewritten as the outer product of an N x 1 vector and a 1 x M vector containing 1's.

$$\mathbf{D}_1 = \begin{bmatrix} \mathbf{P}_0^T\mathbf{P}_0 \\ \vdots \\ \mathbf{P}_{N-1}^T\mathbf{P}_{N-1} \end{bmatrix}_{N\times 1} \cdot \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}_{1\times M}$$

Similarly, $\mathbf{D}_2$ can be rewritten as the outer product of an N x 1 vector of 1's and a 1 x M vector.

$$\mathbf{D}_2 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1} \cdot \begin{bmatrix} \mathbf{S}_0^T\mathbf{S}_0 & \cdots & \mathbf{S}_{M-1}^T\mathbf{S}_{M-1} \end{bmatrix}_{1\times M}$$

So, matrix $\mathbf{D}$ is given by:

$$\mathbf{D} = \overbrace{\begin{bmatrix} \mathbf{P}_0^T\mathbf{P}_0 \\ \vdots \\ \mathbf{P}_{N-1}^T\mathbf{P}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}}^{\mathbf{D}_1} + \overbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S}_0^T\mathbf{S}_0 & \cdots & \mathbf{S}_{M-1}^T\mathbf{S}_{M-1} \end{bmatrix}}^{\mathbf{D}_2} - \overbrace{2\,\mathbf{P}^T\mathbf{S}}^{\mathbf{D}_3}$$

However, we may use Numpy to compute this in a tricky way (See the function 'vectorized' in 'sedm.py'), according to the steps below:

(1) Hadamard product

$$\mathbf{P} \circ \mathbf{P} = \begin{bmatrix} x_0^2 & \cdots & x_{N-1}^2 \\ y_0^2 & \cdots & y_{N-1}^2 \\ z_0^2 & \cdots & z_{N-1}^2 \end{bmatrix}_{3\times N}$$

(2) sum elements along the rows

$$\mathrm{sum}(\mathbf{P} \circ \mathbf{P},\ \mathrm{axis}=0) = \begin{bmatrix} x_0^2 & \cdots & x_{N-1}^2 \\ +\,y_0^2 & \cdots & +\,y_{N-1}^2 \\ +\,z_0^2 & \cdots & +\,z_{N-1}^2 \end{bmatrix}_{1\times N} = \begin{bmatrix} \mathbf{P}_0^T\mathbf{P}_0 & \cdots & \mathbf{P}_{N-1}^T\mathbf{P}_{N-1} \end{bmatrix}_{1\times N}$$

(3) Hadamard product

$$\mathbf{S} \circ \mathbf{S} = \begin{bmatrix} x_0'^2 & \cdots & x_{M-1}'^2 \\ y_0'^2 & \cdots & y_{M-1}'^2 \\ z_0'^2 & \cdots & z_{M-1}'^2 \end{bmatrix}_{3\times M}$$

(4) sum elements along the rows

$$\mathrm{sum}(\mathbf{S} \circ \mathbf{S},\ \mathrm{axis}=0) = \begin{bmatrix} x_0'^2 & \cdots & x_{N-1}'^2 \\ +\,y_0'^2 & \cdots & +\,y_{N-1}'^2 \\ +\,z_0'^2 & \cdots & +\,z_{N-1}'^2 \end{bmatrix}_{1\times M} = \begin{bmatrix} \mathbf{S}_0^T\mathbf{S}_0 & \cdots & \mathbf{S}_{M-1}^T\mathbf{S}_{M-1} \end{bmatrix}_{1\times M}$$

$$\begin{bmatrix} \mathbf{P}_0^T\mathbf{P}_0 & \cdots & \mathbf{P}_{N-1}^T\mathbf{P}_{N-1} \end{bmatrix} \xrightarrow{\text{NEWAXIS}} \qquad \begin{bmatrix} \mathbf{S}_0^T\mathbf{S}_0 & \cdots & \mathbf{S}_{M-1}^T\mathbf{S}_{M-1} \end{bmatrix} \xrightarrow{\text{NEWAXIS}}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{P}_0^T\mathbf{P}_0 \\ \vdots \\ \mathbf{P}_{N-1}^T\mathbf{P}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S}_0^T\mathbf{S}_0 & \cdots & \mathbf{S}_{M-1}^T\mathbf{S}_{M-1} \end{bmatrix} - 2\,\mathbf{P}^T\mathbf{S}$$

Python uses Numpy broadcasting rules to compute matrix D according to this equation. These rules, however, allow evaluating this equation without actually creating neither the vectors of 1's nor computing the outer products to form the full matrices D1 and D2.