Least square & Crust

2D least squares

Line equation. We want to fit a line $y = ax + b$ on a set of input points using the least square method. Open the file "leastSquaresCurve.sce" with scilab and complete the function "mcDroite(x,y)" to find the coeficients a and b of the line. Remender : we want to solve the system

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \Longleftrightarrow Ax = b$$

We thus want to minimize $E(a,b) = \sum_{i=1}^{n}(ax_i + b - y_i)^2$. By computing the partial derivatives of E, we get $(A^T A)x = A^T b$ and then $x = (A^T A)^{-1} A^T b$.

Test and compare using the following command lines. The variable inputs=$\{1, 2, 3\}$ allows to test different data sets :

```
--> exec("leastSquaresCurve.sce",-1)
--> mc(1,inputs)
```

Quadric equation. Complete the function "mcQuadrique(x,y)" in the same way to find the coeficients $[a, b, c]$ of the parabola $y = ax^2 + bx + c$ that fits the inputs points. Try with :

```
--> exec("leastSquaresCurve.sce",-1)
--> mc(2,inputs)
```

Cubic equation. Same with "mcCubique(x,y)" to find the coeficients $[a, b, c, d]$ of the cubic $y = ax^3 + bx^2 + cx + d$ that fits the input points. Try with :

```
--> exec("leastSquaresCurve.sce",-1)
--> mc(3,inputs)
```

3D least squares

Plane surface. We now want to approximate a surface using a plane equation $z = ax + by + c$. Open the file "leastSquaresSurface.sce" and complete the function "mcPlan(x,y,z)". This function takes the points x, y and z as input and returns the coeficients of the plane a, b and c. On thus want to minimize :

$$E(a,b,c) = \sum_{i=1}^{n}(ax_i + by_i + c - z_i)^2$$

The resulting surface can be created using the following command lines. The variable inputs=$\{1, 2, 3\}$ allows to try with different data sets.

```
-->exec("leastSquaresSurface.sce",-1)
-->mcSurface(1,inputs)
```

Use geomview (in the console) to visualize and compare resulats with different inputs :

```
geomview points.vect surface.mesh
```

Quadric surface. Same with "mcQuadric(x,y,z)" to find the coeficients $[a, b, c]$ of the quadric surface $y = ax^2 + by^2 + c$ that fits the input points. Try with :

```
-->exec("leastSquaresSurface.sce",-1)
-->mcSurface(2,inputs)
```

Visualize with geomview and compare.

Explicit curve reconstruction

Crust The goal here is to implement the Crust algorithm as seen during the lecture. The Delaunay triangulation is given and can be used this way :

```
[T,C,r] = delaunay(S);
// T = list of indexed triangles
// C = centers of the circumscribed circles
// r = radii of the circumscribed circles
```

Open the file "crust.sce" and implement the Crust algorithm that allows the curve to be explicitely reconstructed from unordered input data. Try your implemenation with different input data sets.

Crust algorithm

Input: P : Set of points
Output: $CRUST$: display the edges of the output curve

Compute the Delaunay triangulation $\mathcal{T}$;
Compute the set $\mathcal{C}$ of the circumscribed circles associated to triangles $\mathcal{T}$;
Compute the Delaunay triangulation $\mathcal{D}$ of the set of points $\mathcal{T} \cup \mathcal{C}$;
CRUST = set of edges of $\mathcal{D}$ uniquely connected by points in $\mathcal{P}$;

Algorithm 1: CRUST