Computer graphics
Curves and surfaces
Differential geometry in practice

BY SABIR ILYASS.

November 2022.

Note 1.

The code presented is available at my GitHub repository: https://github.com/SABIR-ILYASS/Curves-and-surfaces-differential-geometry-in-practice

Note 2.

MATLAB code will be written between ♣♣♣♣♣♣♣

1 Differential geometry of curves and surfaces

1.1 Planar curves

We will start by studying planar curves. Two types of curves will be studied : curves defined by their parametric equation, and curves defined manually by the user (points clicked with the mouse).

1.1.1 Study of an ellipse

Let's start by studying an ellipse defined by the following parametrization :

$$X(t) = \begin{pmatrix} a\cos(2\pi t) \\ b\sin(2\pi t) \end{pmatrix}$$

The function $\boldsymbol{X} = \mathbf{ellipse}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{n})$ returns a matrix X of two rows and n columns containing the coordinates of n points obtained by uniform sampling of the parameter t between 0 and 1. A succession of points X can be represented using the function **display_curve(*X*).**

Q1: We will try to write a function **tracer_frenet(X)** that computes the tangent vector at each point, normalizes it and displays it, but before that, setting up the mathematical formula of the tangent vector in the discrete case (in the case of a sequence of points).

For a cloud of points $(X_i)_{1\leqslant i\leqslant n}$ of $\mathbb{R}^2$. denote For all $i \in [\![1,\, n]\!]$ $T(i)$: the tangent vector of the point X_i.

$$T(i) = \frac{dX}{ds} = \frac{X_{i+1} - X_{i-1}}{\|X_{i+1} - X_{i-1}\|} := \begin{pmatrix} T_x(i) \\ T_y(i) \end{pmatrix} \tag{1}$$

With s is the arclength,$\|.\|$:represents the Euclidean norm of $\mathbb{R}^2$, $X_{-1} = X_n$ and $X_{n+1} = X_0$.

Figure 1. Exemple of a point cloud
In green the tangent vector.
In red the normal vector.

Here is the MATLAB code of the function **tracer_frenet(X)** which calculates the tangent vector at each point, normalizes it and displays it.

♣♣♣♣♣♣♣

```
function tracer_frenet (X)

%{
This function allows to draw the tangent vectors, the normal vectors
and the centers of curvature of the curve defined by a point cloud.

we present in this question only the tangent vectors.

Data input:
 - X: cloud of points

Author: SABIR ILYASS - 2022.
%}

% Get the number of points of X
n = size(X,2);

% Initializing tangent vectors
tangent_X = zeros(2,n - 2);

% Tangent vectors
for i = 1:n-2
    tangent_X(:,i) = (X(:,i + 2) - X(:, i)) / norm(X(:,i) - X(:, i + 2));
end

axis equal;
hold on;
```

```
quiver(X(1,2:n-1), X(2,2:n-1), tangent_X(1,:), tangent_X(2,:), 'off');
title('tangent vectors');
```

♣♣♣♣♣♣♣

For the ellipse with parameters $a=8$ and $b=5$: ($\boldsymbol{X}=\mathbf{ellipse(8,5,20)}$).

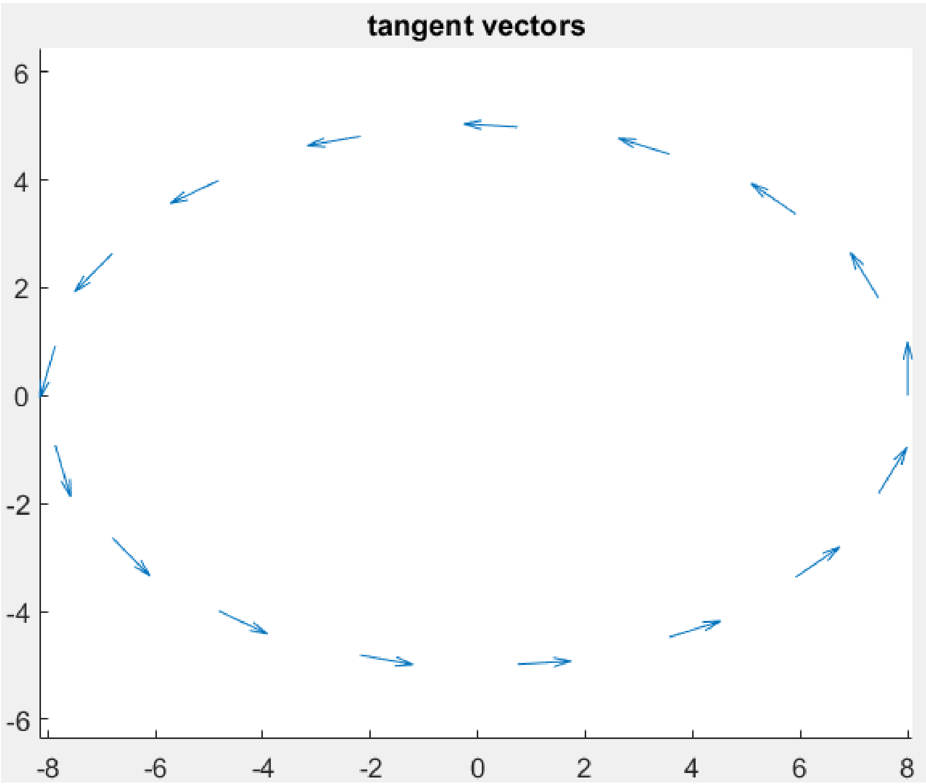

Figure 2. Tangent vectors of an ellipse ($a=8, b=5$)

Remark 3. we notice that the tangent divergence is zero on a closed contour

$$\oint_{s\in\mathcal{C}} T(s)ds=0$$

because the function $t\longrightarrow M(t):=\begin{pmatrix} a\cos(2\pi t) \\ b\sin(2\pi t) \end{pmatrix}$ is a C-infinity function.

Proof. We have for all $t\in[0,1]$,

$$\begin{aligned} s(t) &:= \int_0^t \left\| \frac{dM(u)}{du} \right\| du \\ &= 2\pi \int_0^t \sqrt{a^2\sin^2(2\pi u)+b^2\cos^2(2\pi u)}\,du \end{aligned}$$

And

$$\begin{aligned}\oint_{s\in\mathcal{C}} T(s)ds &= \oint_{s\in\mathcal{C}} \frac{dM(s)}{ds}\times \mathrm{ds}\\ &\oint_{s\in\mathcal{C}} dM(s)\end{aligned}$$

With for all $t\in[0,1]$ we have

$$\begin{aligned}dM(s) &= \frac{dM\left(2\pi\int_0^t \sqrt{a^2\sin^2(2\pi u)+b^2\cos^2(2\pi u)}\,du\right)}{\mathrm{dt}}\mathrm{dt}\\ &= 2\pi\frac{dM}{\mathrm{dt}}\left(2\pi\int_0^t \sqrt{a^2\sin^2(2\pi u)+b^2\cos^2(2\pi u)}\,du\right)\sqrt{a^2\sin^2(2\pi t)+b^2\cos^2(2\pi t)}\,\mathrm{dt}\end{aligned}$$

So by change of variable we have: $\oint_{s\in\mathcal{C}} T(s)ds$ is equal to

$$2\pi\int_0^1\frac{dM}{\mathrm{dt}}\left(2\pi\int_0^t \sqrt{a^2\sin^2(2\pi u)+b^2\cos^2(2\pi u)}\,du\right)\sqrt{a^2\sin^2(2\pi t)+b^2\cos^2(2\pi t)}\,\mathrm{dt}$$

The function defined by
$t\longmapsto \frac{dM}{\mathrm{dt}}\left(2\pi\int_0^t \sqrt{a^2\sin^2(2\pi u)+b^2\cos^2(2\pi u)}\,du\right)\sqrt{a^2\sin^2(2\pi t)+b^2\cos^2(2\pi t)}$ is 1-periodic, so its primitive function is also 1-periodic (via Fourier series for exemple).

So

$$\int_0^1\frac{dM}{\mathrm{dt}}\left(2\pi\int_0^t \sqrt{a^2\sin^2(2\pi u)+b^2\cos^2(2\pi u)}\,du\right)\sqrt{a^2\sin^2(2\pi t)+b^2\cos^2(2\pi t)}\,\mathrm{dt} = 0$$

Therefore $\oint_{s\in\mathcal{C}} T(s)ds = 0$.

□

Q2: We complete the function **tracer_frenet(X)** by computing the vector "derivative of the tangent".

For a cloud of point $(X_i)_{1\leqslant i\leqslant n}$ of $\mathbb{R}^2$.

We derive the tangent vector defined in equation (1)

$$\begin{aligned}\forall i\in[\![2,n-1]\!]\frac{\mathrm{dT}}{ds}(i) &= \frac{\frac{X_{i+1}-X_i}{\|X_{i+1}-X_i\|}-\frac{X_i-X_{i-1}}{\|X_i-X_{i-1}\|}}{\left\|\frac{X_{i+1}+X_i}{2}-\frac{X_i-X_{i-1}}{2}\right\|}\\ &= 2\frac{\frac{X_{i+1}-X_i}{\|X_{i+1}-X_i\|}-\frac{X_i-X_{i-1}}{\|X_i-X_{i-1}\|}}{\|X_{i+1}-X_{i-1}\|}\end{aligned}$$

♣♣♣♣♣♣♣

```
function tracer_frenet (X)

%{
This function allows to draw the tangent vectors, the normal vectors
and the centers of curvature of the curve defined by a point cloud.

we present in this question only the tangent and normal vectors.

Data input:
 - X: cloud of points

Author: SABIR ILYASS - 2022.
%}

% Get the number of points of X
n = size(X,2);

% Initializing tangent vectors
tangent_X = zeros(2,n - 2);

% Initializing the derivation of the tangent vectors and the normal vectors
derived_tengant_vector = zeros(2, n - 2);
normal_X = zeros(2,n - 2);


% Tangent vectors
for i = 1:n-2
    tangent_X(:,i) = (X(:,i + 2) - X(:, i)) / norm(X(:,i) - X(:, i + 2));
end

% Derivation of the tangent vectors and the normal vectors
for i = 1:n-2
    derived_tengant_vector(:,i) =2 * ((X(:,i+2) - X(:,i+1))/norm(X(:,i+2) - X(:,i+1)) - (X(:,i+1) - X(:,i))/norm(X(:,i+1) - X(:,i))) / norm(X(:,i+2) - X(:,i));
    normal_X(:,i) = derived_tengant_vector(:,i) / norm(derived_tengant_vector(:,i));
end

axis equal;
hold on;

quiver(X(1,2:n-1), X(2,2:n-1), tangent_X(1,:), tangent_X(2,:), 'off');
quiver(X(1,2:n-1), X(2,2:n-1), normal_X(1,:), normal_X(2,:), 'off');
title('tangent and normal vectors');
```

♣♣♣♣♣♣♣

We obtain for the ellipse with parameters $a=8$ and $b=5$:

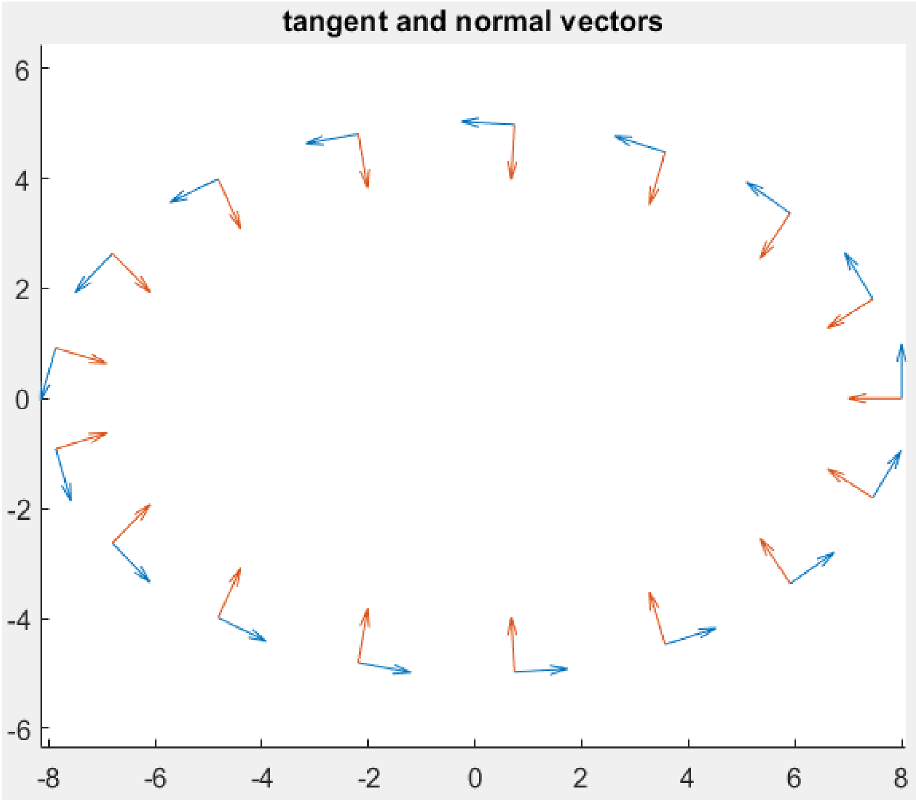

Figure 3. Tangent and normal vectors of an ellipse ($a = 8, b = 5$)

Remark 4.

We can also obtain the normal vector by the following formula

$$\forall i \in [\![1, n]\!], N(i) = \begin{pmatrix} -T_y(i) \\ T_x(i) \end{pmatrix}$$

Q3 and Q4: We add the calculation of the center of curvature by calculating the curvature at a given point of the cloud

For all $i \in [\![1, n]\!]$, we have the curvature at point i is:

$$\gamma_i = \left\| \frac{\mathrm{dT}}{ds}(i) \right\|$$

And the center of curvature at this point is

$$\Omega_i = X_i + \frac{1}{\gamma_i} N(i)$$

With $N(i)$ the normal vector at point i.

♣♣♣♣♣♣♣

```
function tracer_frenet (X)

%{
This function allows to draw the tangent vectors, the normal vectors
and the centers of curvature of the curve defined by a point cloud.

Data input:
 - X: cloud of points

Author: SABIR ILYASS - 2022.
%}

% Get the number of points of X
n = size(X,2);

% Initializing tangent vectors
tangent_X = zeros(2,n - 2);

% Initializing the derivation of the tangent vectors and the normal vectors
derived_tengant_vector = zeros(2, n - 2);
normal_X = zeros(2,n - 2);

% Initializing the center of curvature
center_of_curvature = zeros(2, n-2);

% Tangent vectors
for i = 1:n-2
    tangent_X(:,i) = (X(:,i + 2) - X(:, i)) / norm(X(:,i) - X(:, i + 2));
end

% Derivation of the tangent vectors and the normal vectors
for i = 1:n-2
    derived_tengant_vector(:,i) = 2 * ((X(:,i+2) - X(:,i+1))/norm(X(:,i+2) - X(:,i+1))
- (X(:,i+1) - X(:,i))/norm(X(:,i+1) - X(:,i))) / norm(X(:,i+2) - X(:,i));
    % center of curvature
    curvature = norm(derived_tengant_vector(:,i));
    normal_X(:,i) = derived_tengant_vector(:,i) / curvature;
    center_of_curvature(:,i) = X(:,i + 1) + normal_X(:,i) ./ curvature;
end

axis equal;
hold on;

quiver(X(1,2:n-1), X(2,2:n-1), tangent_X(1,:), tangent_X(2,:), 'off');
quiver(X(1,2:n-1), X(2,2:n-1), normal_X(1,:), normal_X(2,:), 'off');
title('tangent and normal vectors');

for i = 1 : n -2
    cercles(X(1,i+1), X(2,i+1),center_of_curvature(1,i), center_of_curvature(2,i));
```

```
end
```

♣♣♣♣♣♣♣

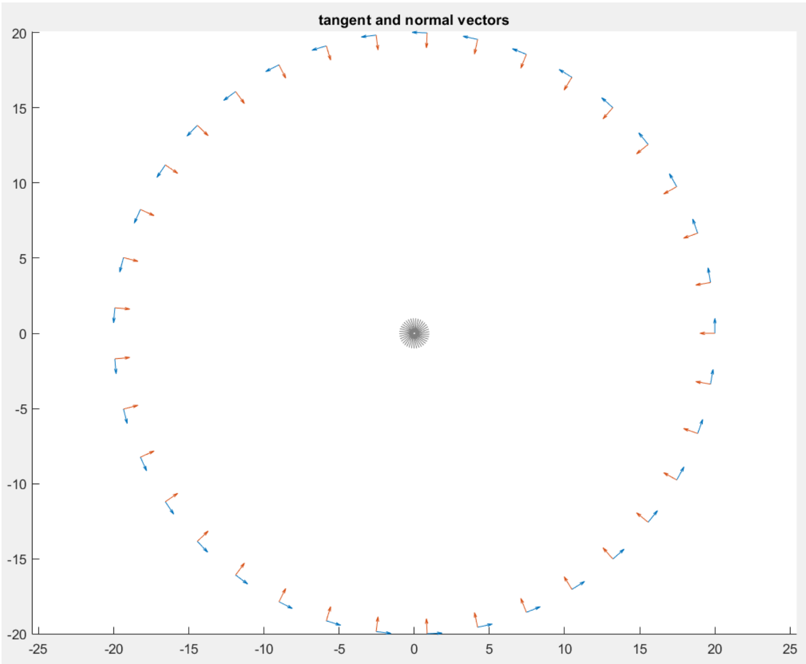

Figure 4. Tangent and normal vectors and the centers of curvature of an ellipse $(a=20, b=20)$

Remark 5. It is normal not to have a single point of curvature for a **circle**, since it depends on the precision of the circle algorithm and the rounding problems.

1.1.2 Study of a manually entered curve

Q5: To study other manually entered curves. We will use the same function **tracer_frenet**. Except the change of the second to last line to:

```
cercles(center_of_curvature(1,i), center_of_curvature(2,i),X(1,i+1), X(2,i+1));
```

To plot the tangent and normal vector at all points we adapt the input points by repeating some points to be adapted to the function **tracer_frenet**

♣♣♣♣♣♣♣

```
% 1.1.2 Study of a manually entered curve
% Question 5:
X2 = saisir_courbe;
n = size(X2,2);
Y = zeros(2, n + 2);
Y(:,1:n) = X2;
Y(:,n+1) = X2(:,1); X2(:,n+2) = X2(:,2);
tracer_frenet(Y);
```

♣♣♣♣♣♣♣

Here is a cloud of points entered by mouse:

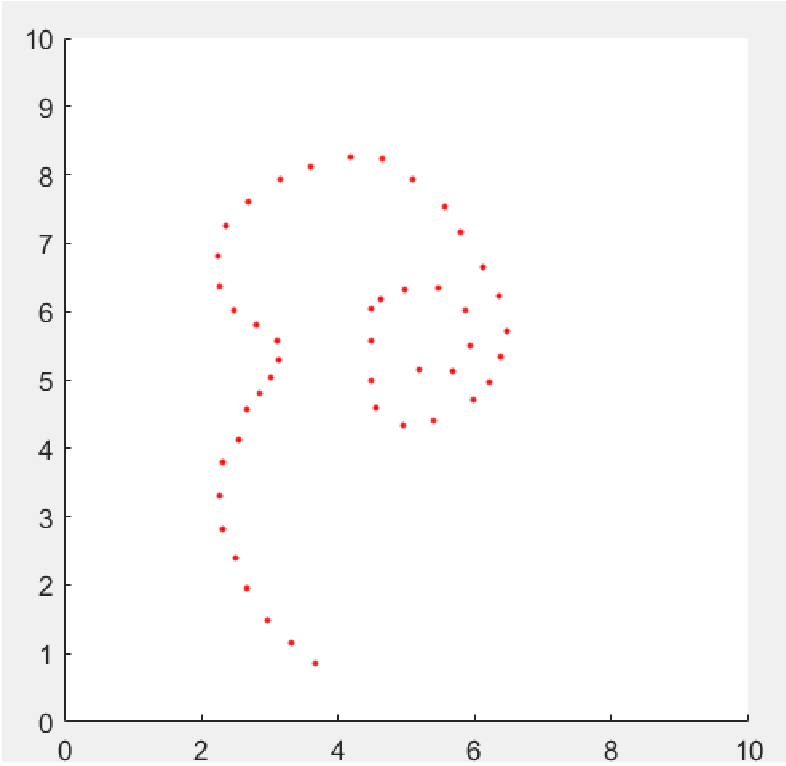

Figure 5. Cloud of points entered by mouse

And here is the result that we obtain by applying the function **tracer_frenet**:

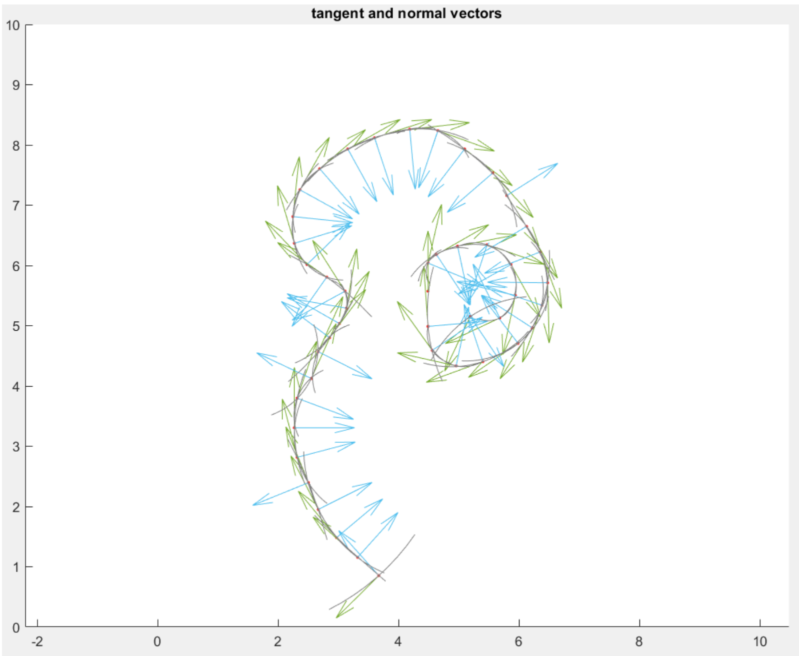

Figure 6. Frenet frames and portions of osculating circles.

1.2 An example of a surface : the digital terrain model

A DTM is presented as a matrix (image) defining at any point (u, v) of a uniform sampling grid the height $z(u, v)$. The DTM can thus be considered as a parameterized surface :

$$X(u,v) = \begin{pmatrix} u \\ v \\ z(u,v) \end{pmatrix}$$

Using the notation used in class, we will note the partial derivatives : $X_\bullet \equiv \frac{\partial X}{\partial \bullet}$, $X_{\bullet\star} \equiv \frac{\partial^2 X}{\partial \bullet \partial \star}$ etc. . . . The tangent plane to the surface X at a point (u, v) is given by the basis (X_u, X_v). The normal can be obtained directly : $N \equiv \frac{X_u \wedge X_v}{\|X_u \wedge X_v\|}$

In questions 6 and 7 we will work on the image presented in the following figure.

We denote H and W respectively the height and width the image. and for all $(i, j) \in [\![1, H]\!] \times [\![1, W]\!]$ we denote $\mathrm{im}(i, j)$: the pixel at position (i, j). and

$$X(i,j) = \begin{pmatrix} i \\ j \\ \mathrm{im}(i,j) \end{pmatrix}$$

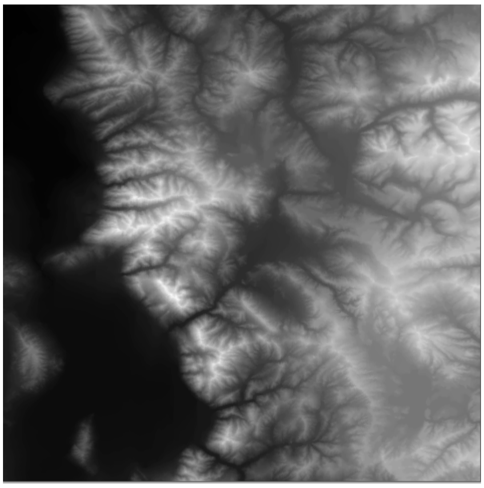

Figure 7. saltlake.png

Q6 The tangent and normal vectors to the DTM surface saltlake.png

We have for all $(i, j) \in [\![2, H-1]\!] \times [\![2, W-2]\!]$

$$\begin{aligned} X_u(i,j) &= \frac{\partial}{\partial i}\begin{pmatrix} i \\ j \\ \mathrm{im}(i,j) \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \\ \frac{\partial}{\partial i}\mathrm{im}(i,j) \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \\ \frac{1}{2}[\mathrm{im}(i+1,j) - \mathrm{im}(i-1,j)] \end{pmatrix} \end{aligned}$$

And

$$\begin{aligned} X_v(i,j) &= \frac{\partial}{\partial j}\begin{pmatrix} i \\ j \\ \mathrm{im}(i,j) \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 1 \\ \frac{\partial}{\partial j}\mathrm{im}(i,j) \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 1 \\ \frac{1}{2}[\mathrm{im}(i,j+1) - \mathrm{im}(i,j-1)] \end{pmatrix} \end{aligned}$$

The normal vector is obtained by applying the following formula

$$\begin{aligned} N(i,j) &= \frac{X_u(i,j) \wedge X_v(i,j)}{\|X_u(i,j) \wedge X_v(i,j)\|} \\ &= \frac{1}{\|X_u(i,j) \wedge X_v(i,j)\|}\begin{pmatrix} -\frac{1}{2}[\mathrm{im}(i+1,j) - \mathrm{im}(i-1,j)] \\ -\frac{1}{2}[\mathrm{im}(i,j+1) - \mathrm{im}(i,j-1)] \\ 1 \end{pmatrix} \end{aligned}$$

Here is above the MATLAB code of a function that allows to calculate and represent the tangential and normal vectors to the surface.

♣♣♣♣♣♣♣

```
function image_2_3D(im, f)

%{
The following function transforms a given input image into a parameterized
surface whose height is equal to the image pixel value.

The two tangent vectors and the normal vector are calculated and displayed.

Input data:
 - im: image.
 - f: real parameter.

Author: SABIR ILYASS - 2022.
%}

% Resize the image
im = imresize(im, f);
% Get the height and width
[h,w] = size(im);

image_3D_x = zeros(h,w);
```

```
image_3D_y = zeros(h,w);

for i = 1:h
    for j = 1:w
        image_3D_x(i, j) = i;
        image_3D_y(i, j) = j;
    end
end

figure;
surf(image_3D_x, image_3D_y, im);


if (h >= 2 && w >= 2)
    % Initializing tangent and normal vectors
    X_u_1 = zeros(h - 2,w - 2);
    X_u_3 = zeros(h - 2,w - 2);

    X_v_2 = zeros(h - 2,w - 2);
    X_v_3 = zeros(h - 2,w - 2);

    Normal_1 = zeros(h - 2,w - 2);
    Normal_2 = zeros(h - 2,w - 2);
    Normal_3 = zeros(h - 2,w - 2);

    for i = 1:h-2
        for j = 1:w-2
            norm_u_ij = sqrt((im(i + 2,j) - im(i, j))^2 / 4 + 1);
            X_u_1(i,j) = 1 / norm_u_ij;
            X_u_3(i,j) = (im(i + 2,j) - im(i,j)) / (2 * norm_u_ij);

            norm_v_ij = sqrt((im(i,j + 2) - im(i, j))^2 / 4 + 1);
            X_v_2(i,j) = 1 / norm_v_ij;
            X_v_3(i,j) = (im(i,j + 2) - im(i, j)) / (2 * norm_v_ij);

            norm_n = sqrt(X_u_3(i,j)^2 + X_v_3(i,j)^2 + 1);
            Normal_1(i,j) = - X_u_3(i,j)/norm_n;
            Normal_2(i,j) = - X_v_3(i,j)/norm_n;
            Normal_3(i,j) = 1/norm_n;
        end
    end

    hold on;
    quiver3(image_3D_x(2:h-1,2:w-1),image_3D_y(2:h-1,2:w-1),im(2:h-1,2:w-
1),X_u_1,zeros(h - 2,w - 2),X_u_3,'off');
    quiver3(image_3D_x(2:h-1,2:w-1),image_3D_y(2:h-1,2:w-1),im(2:h-1,2:w-
1),zeros(h - 2,w - 2),X_v_2,X_v_3,'off');
    quiver3(image_3D_x(2:h-1,2:w-1),image_3D_y(2:h-1,2:w-1),im(2:h-1,2:w-
1),Normal_1,Normal_2,Normal_3,'off');

end
```

♣♣♣♣♣♣♣

This is what the function **image_2_3D** applied to the image gives:

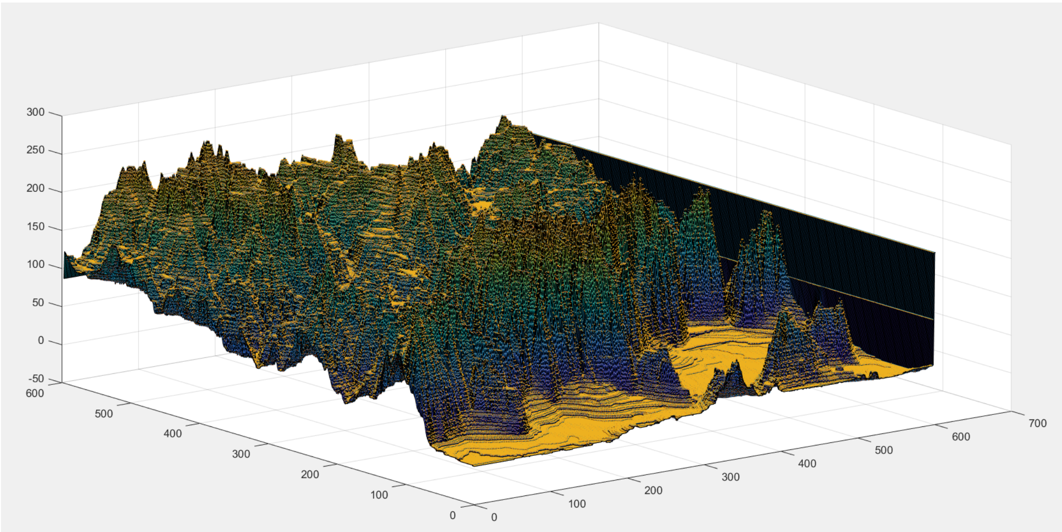

Figure 8. parameterized surface of the image "saltlake.png”

Let’s zoom in to illustrate the different tangent and normal vectors

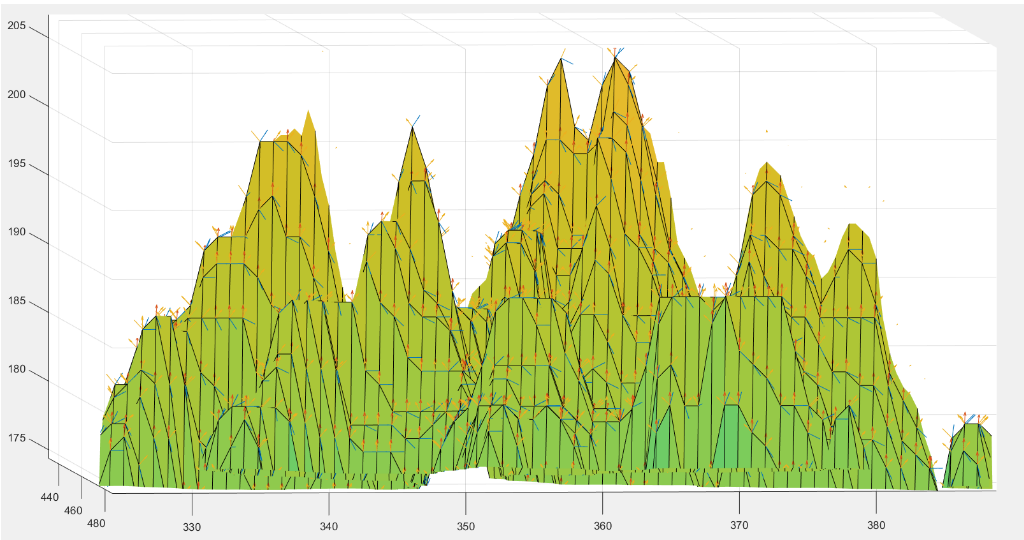

Figure 9. Zoom in the figure 8.

Q7

The Gauss curvature:

The Gauss curvature, sometimes also called the total curvature, of a parametric surface X in $X(P)$ is the product of the principal curvatures. Equivalently, the Gauss curvature is the determinant of the **Weingarten** endomorphism.

The Mean curvature:

The average curvature of a surface is the average of the minimum and maximum curvatures.

The mean curvature is defined as the average of the two main curvatures

Theorem 6.

Let a surface be parametrized by two parameters u and v, and let $\boldsymbol{I = Edu^2 + 2Fdudv + Gdv^2}$ the first fundamental form, $\boldsymbol{II = Ldu^2 + 2Mdudv + Ndv^2}$ the second fundamental form. Then

The Gaussian curvature is:

$$\boldsymbol{K = \frac{LN - M^2}{EG - F^2}}$$

The mean curvature is:

$$\boldsymbol{\gamma = \frac{EN + GL - 2MF}{EG - F^2}}$$

Proof. Let $(u, v) \longmapsto M(u, v)$ a parametrization of the surface.

A basis of the tangent plane is given by $\frac{\partial M}{\partial u}$ and $\frac{\partial M}{\partial v}$.

Let (a, b) a point in the surface, and x and y two vectors of the tangent plane at (a, b). and let X and Y be the components of these two vectors in t he basis $\left(\frac{\partial M}{\partial u}(a, b), \frac{\partial M}{\partial v}(a, b)\right)$

The first fundamental form gives the expression in this basis of the scalar product of the two vectors:

$$\langle x, y \rangle = X^t \begin{pmatrix} E & F \\ F & G \end{pmatrix} Y$$

The second fundamental form is the quadratic form associated with the Weingarten symmetric endomorphism W, whose two eigenvalues are the principal curvatures of the surface at the considered point.

$$\langle x, W(y) \rangle = X^t \begin{pmatrix} L & M \\ M & N \end{pmatrix} Y$$

Therefore, if y is an eigenvector of the Weingarten endomorphism, with eigenvalue λ, we have for all x:

$$\langle x, W(y) \rangle = X^t \begin{pmatrix} L & M \\ M & N \end{pmatrix} Y = \lambda \langle x, y \rangle = \lambda X^t \begin{pmatrix} E & F \\ F & G \end{pmatrix} Y$$

So

$$\left[\begin{pmatrix} L & M \\ M & N \end{pmatrix} - \lambda \begin{pmatrix} E & F \\ F & G \end{pmatrix} \right] Y = 0$$

And therefore $\begin{pmatrix} L & M \\ M & N \end{pmatrix} - \lambda \begin{pmatrix} E & F \\ F & G \end{pmatrix}$ is non-inversible, so

$$\det\left(\begin{pmatrix} L & M \\ M & N \end{pmatrix} - \lambda \begin{pmatrix} E & F \\ F & G \end{pmatrix} \right) = 0$$

We get

$$\lambda^2 - \frac{EN + GL - 2MF}{EG - F^2} \lambda + \frac{LN - M^2}{EG - F^2} = 0$$

And the product of the eigenvalues of W is $\frac{LN - M^2}{EG - F^2}$ which is none other than the Gaussian curvature.

And we get the half-sum of the two roots $\frac{EN + GL - 2MF}{EG - F^2}$ which is none other than the average curvature. □

To implement these two curves, we need to calculate the second derivatives of X.

We have for all $(i,j) \in [\![3, H-2]\!] \times [\![3, W-2]\!]$

$$\begin{aligned} X_{uu} &= \frac{\partial X_u}{\partial u} \\ &= \frac{\partial}{\partial u}\begin{pmatrix} 1 \\ 0 \\ \frac{1}{2}[\mathrm{im}(i+1,j) - \mathrm{im}(i-1,j)] \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2}\left[\frac{\partial}{\partial u}\mathrm{im}(i+1,j) - \frac{\partial}{\partial u}\mathrm{im}(i-1,j)\right] \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ \frac{1}{4}[\mathrm{im}(i+2,j) + \mathrm{im}(i-2,j) - 2\mathrm{im}(i,j)] \end{pmatrix} \end{aligned}$$

And

$$\begin{aligned} X_{uv} &= \frac{\partial X_v}{\partial u} \\ &= \frac{\partial}{\partial u}\begin{pmatrix} 0 \\ 1 \\ \frac{1}{2}[\mathrm{im}(i,j+1) - \mathrm{im}(i,j-1)] \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2}\left[\frac{\partial}{\partial u}\mathrm{im}(i,j+1) - \frac{\partial}{\partial u}\mathrm{im}(i,j-1)\right] \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ \frac{1}{4}[\mathrm{im}(i+1,j+1) + \mathrm{im}(i-1,j-1) - \mathrm{im}(i-1,j+1) - \mathrm{im}(i+1,j-1)] \end{pmatrix} \end{aligned}$$

And

$$\begin{aligned} X_{vv} &= \frac{\partial X_v}{\partial v} \\ &= \frac{\partial}{\partial v}\begin{pmatrix} 0 \\ 1 \\ \frac{1}{2}[\mathrm{im}(i,j+1) - \mathrm{im}(i,j-1)] \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2}\left[\frac{\partial}{\partial v}\mathrm{im}(i,j+1) - \frac{\partial}{\partial v}\mathrm{im}(i,j-1)\right] \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ \frac{1}{4}[\mathrm{im}(i,j+2) + \mathrm{im}(i,j-2) - 2\mathrm{im}(i,j)] \end{pmatrix} \end{aligned}$$

Here is the MATLAB code of the Gauss curvature and the mean curvature:

♣♣♣♣♣♣♣

```
function H = Gaussian_mean_curvature(im,f, method)

%{
This function allows to calculate, display the gaussian curvature
and the mean curvature of a given image at the input.

Input data:
 - im: image.
 - f: real parameter.
 - method: "gaussian" or "mean"

Author: SABIR ILYASS - 2022.

%}
im = imresize(im, f);
[w,h] = size(im);

H = zeros(w - 4,h - 4);

if (w >= 2 && h >= 2)
 % initializing tangent vectors
 X_u_1 = zeros(w - 2,h - 2);
 X_u_3 = zeros(w - 2,h - 2);

 X_v_2 = zeros(w - 2,h - 2);
 X_v_3 = zeros(w - 2,h - 2);

 % initializing the derivation of tangent vectors

 X_uu_3 = zeros(w - 4,h - 4);

 X_uv_3 = zeros(w - 4,h - 4);

 X_vv_3 = zeros(w - 4,h - 4);

 Normal_1 = zeros(w - 2,h - 2);
 Normal_2 = zeros(w - 2,h - 2);
 Normal_3 = zeros(w - 2,h - 2);

 for i = 1:w-2
 for j = 1:h-2
         norm_u_ij = sqrt((im(i + 2,j) - im(i, j))^2 + 1);
         X_u_1(i,j) = 1 / norm_u_ij;
         X_u_3(i,j) = (im(i + 2,j) - im(i,j)) / norm_u_ij;
```

```
            norm_v_ij = sqrt((im(i,j + 2) - im(i, j))^2 + 1);
            X_v_2(i,j) = 1 / norm_v_ij;
            X_v_3(i,j) = (im(i,j + 2) - im(i, j)) / norm_v_ij;

            norm_n = sqrt(X_u_3(i,j)^2 + X_v_3(i,j)^2 + 1);
            Normal_1(i,j) = - X_u_3(i,j)/norm_n;
            Normal_2(i,j) = - X_v_3(i,j)/norm_n;
            Normal_3(i,j) = 1/norm_n;

        end
    end

    for i = 1:w-4
        for j = 1:h-4
            X_uu_3(i,j) = (im(i+4,j) + im(i,j) - 2 * im(i+2,j)) / abs(im(i+4,j) +
im(i,j) - 2 * im(i+2,j));
            X_vv_3(i,j) = (im(i,j+4) + im(i,j) - 2 * im(i,j+2)) / abs(im(i,j+4) + im(i,j)
- 2 * im(i,j+2));
            X_uv_3(i,j) = (im(i + 2,j + 2) + im(i,j) - im(i + 2,j) - im(i,j+2)) / abs(im(i
+ 2,j + 2) + im(i,j) - im(i + 2,j) - im(i,j+2));

            % First fundamental Coeffecients of the surface (E,F,G)
            E = X_u_1(i,j)^2 + X_u_3(i,j)^2;
            F = X_u_3(i,j) * X_v_3(i,j);
            G = X_v_2(i,j)^2 + X_v_3(i,j)^2;

            % Second fundamental Coeffecients of the surface (L,M,N)
            L = Normal_3(i,j) * X_uu_3(i,j);
            M = Normal_3(i,j) * X_uv_3(i,j);
            N = Normal_3(i,j) * X_vv_3(i,j);

            switch method
                case "gaussian"
                    H(i,j) = (L.*N - M.^2)./(E.*G - F.^2);
                case "mean"
                    H(i,j) = (E.*N + G.*L - 2.*F.*M)./(2*(E.*G - F.^2));

                otherwise
                    msg = 'Please enter the name of the corpure among the following
proposals: gaussian and mean.';
                    error(msg);
                end
            end
        end
    end

    image_3D_x = zeros(w - 4,h - 4);
    image_3D_y = zeros(w - 4,h - 4);

    for i = 1:w-4
```

```
    for j = 1:h-4
        image_3D_x(i, j) = i+2;
        image_3D_y(i, j) = j+2;
    end
end
figure;
surf(image_3D_x, image_3D_y, H);
title(sprintf('The %s curvature',method));
```

♣♣♣♣♣♣♣

Here is the gaussian curvature for the image "saltlake.png"

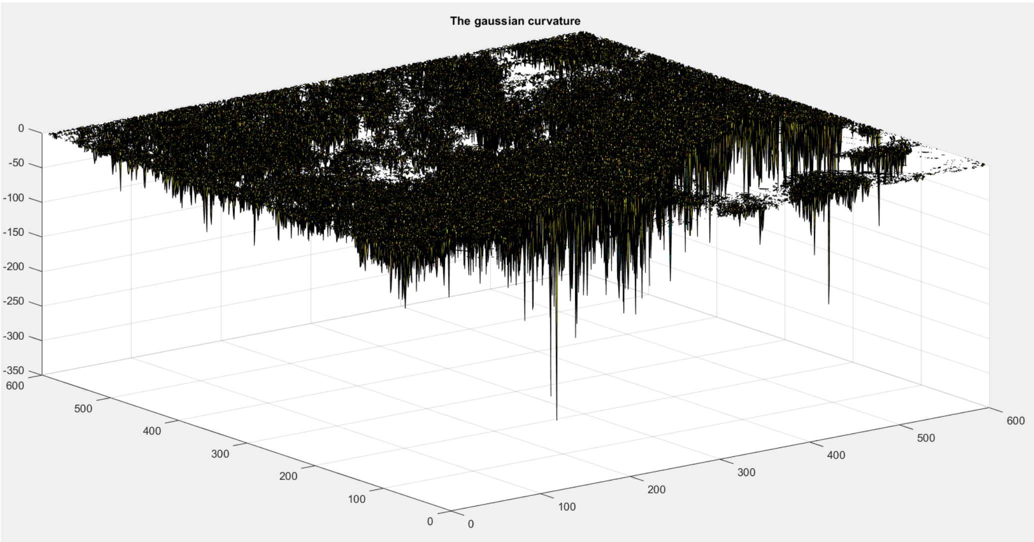

Figure 10. The gaussian curvature of the image "saltlake.png".

And here is the mean curvature for the image "saltlake.png"

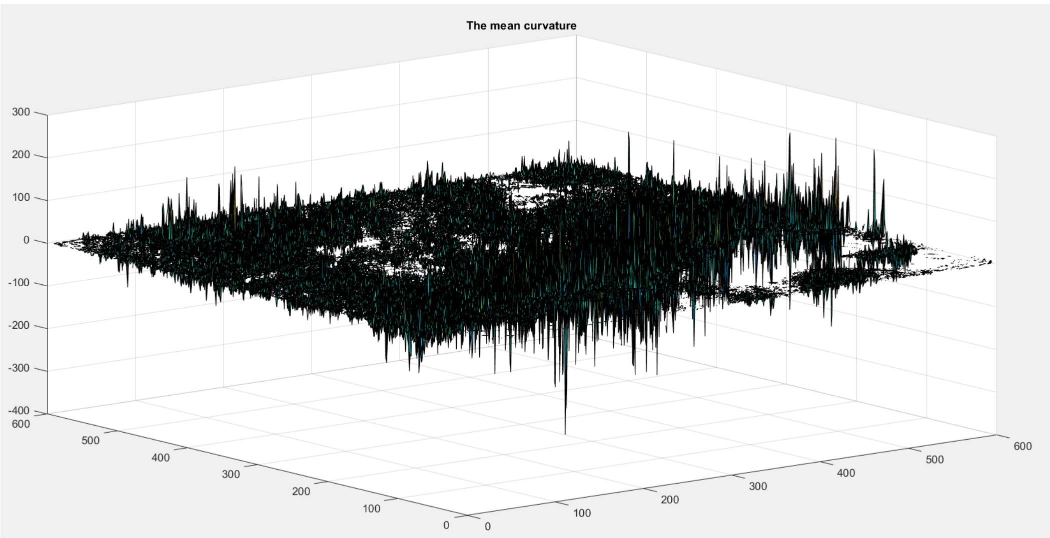

Figure 11. Mean curvature of the image "saltlake.png".

2 Modeling of curves and surfaces

2.1 Curve/surface reconstruction from a point cloud

Q8 We will apply the CRUST algorithm to reconstruct a curve from a collection of non-uniformly sampled points on the curve.

♣♣♣♣♣♣♣

```
function Crust(X)

    %{
    We apply the Crust algorithm that is written in the
    article A New Voronoi-Based Surface Reconstruction Algorithm available at:
    http://compbio.mit.edu/publications/C01_Amenta_Siggraph_98.pdf

    Data input:
     - X: cloud of points.

    Author: SABIR ILYASS - 2022.
    %}

    % Get the number of points of X
    n1 = size(X,2);

    % The Voronoi diagram of X
    [V,~] = voronoin(X');

    % Get the number of vertices in the Voronoi diagram
    n2 = size(V,1);

    % Plot the vertices of the Voronoi diagram
    plot(V(2:n2,1),V(2:n2,2),'.');

    % Concatenate the points of the point cloud X
    % and the vertices of the voronoi diagram V
    poles_and_points = zeros(2,n1 + n2 - 1);
    poles_and_points(:,1:n1) = X;
    poles_and_points(:,n1+1:n1+n2 - 1) = V(2:n2,:)';

    % Delaunay triangulation
    DT = delaunay(poles_and_points(1,:), poles_and_points(2,:));

    % If an edge of the Delaunay triangulation connects two points of the point cloud X,
    % we draw the edge between these two points.
    for i = 1:size(DT,1)
        x1 = DT(i,1); x2 = DT(i,2); x3 = DT(i,3);
        if (x1 <= n1 && x2 <= n1)
            plot([X(1,x1) X(1,x2)], [X(2,x1) X(2,x2)])
        end
        if (x1<= n1 && x3 <= n1)
            plot([X(1,x1) X(1,x3)], [X(2,x1) X(2,x3)])
```

```
        end
        if (x2<= n1 && x3 <= n1)
            plot([X(1,x3) X(1,x2)], [X(2,x3) X(2,x2)])
        end
    end
```

♣♣♣♣♣♣♣

We apply this function with the following code:

♣♣♣♣♣♣♣

```
% 2 Modeling of curves and surfaces0
% 2.1 Curve/surface reconstruction from a point cloud
% Question 8:
X3 = saisir_courbe();
Crust(X3);
```

♣♣♣♣♣♣♣

We obtain for a point cloud entered by mouse:

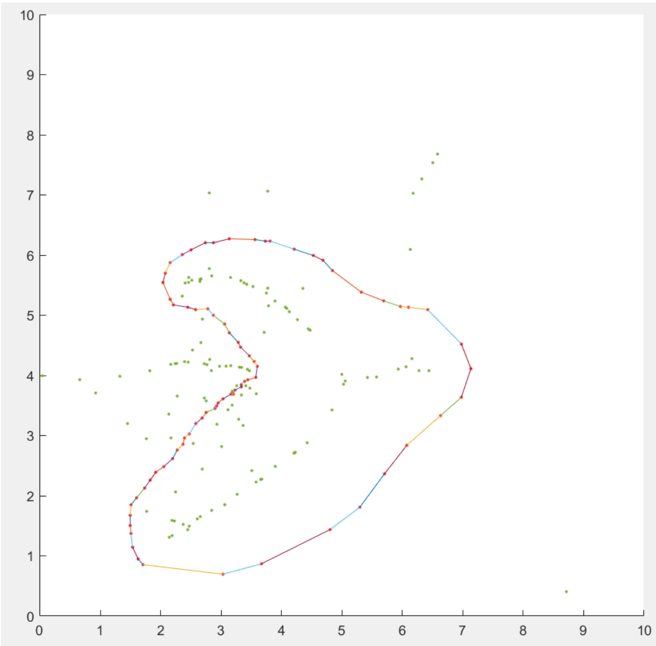

Figure 12. A set of points (red), the vertices of the Voronoi diagram approximating the median axis (blue), the curve reconstructed by the CRUST algorithm (segments).

2.2 Modeling of smooth curves

Q9: Enter $n+1$ points with the mouse P_0 to P_n, draw the control polygon connecting the points. Draw the Bézier curve defined by :

$$\forall u \in [0,1]\, \mathcal{C}(u) = \sum_{i=0}^{n} B_{n,i}(u) P_i \text{ with } B_{n,i}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{(n-i)}$$

we apply the following function:

♣♣♣♣♣♣♣

```
function [C_u_1, C_u_2] = bezier(P, u)

%{
The following function allows to compute the Bézier's curve
of a point cloud given at the input applied to a point u.

Input data:
 - P: cloud of points.
 - u: real number in [0,1].

Output data:
 - C_u_1: the first Bézier's curve coordinate at the point u.
 - C_u_2: the second Bézier's curve coordinate at the point u.

Author: SABIR ILYASS - 2022.
%}

% The following function returns for an integer n
% the i-th Berstein polynomial applied to the point u
bernstein = @(n, i, u) factorial(n) * (u^i) * ((1-u)^(n-i)) / (factorial(i) * factorial(n - i));

n = size(P,2);
C_u_1 = 0; C_u_2 = 0;

for i = 0:n - 1
    C_u_1 = C_u_1 + bernstein(n - 1,i,u) * P(1,i + 1);
    C_u_2 = C_u_2 + bernstein(n - 1,i,u) * P(2,i + 1);
end
```

♣♣♣♣♣♣♣

We apply this function to draw the Bézier curve for a point cloud given manually by the mouse via the following code:

♣♣♣♣♣♣♣

```
% 2.2 Modeling of smooth curves
% Question 9:
% Enter a set of points manually with the mouse
X = saisir_courbe();

% Get the number of points
n = size(X,2);

% initialization of the Bezier curve
% we take 100 * n points to draw more
% points of the beziers curve and to show more its curvature
Bezier_points = zeros(2,n * 100);
```

```
for i = 1: 100 * n
    [Cu_1, Cu_2] = bezier(X, (i - 1) / (100 * n));
    Bezier_points(1,i) = Cu_1;
    Bezier_points(2,i) = Cu_2;
end

% plot segments between the points of the point cloud X
for i = 1:(n - 1)
    plot([X(1,i) X(1, i + 1)],[X(2,i) X(2, i + 1)]);
end

% plot the Bezier curve
for i = 1:(100 * n - 1)
    plot([Bezier_points(1,i) Bezier_points(1, i + 1)],[Bezier_points(2,i) Bezier_points(2, i + 1)]);
end
```

♣♣♣♣♣♣♣

We obtain for a point cloud entered by mouse:

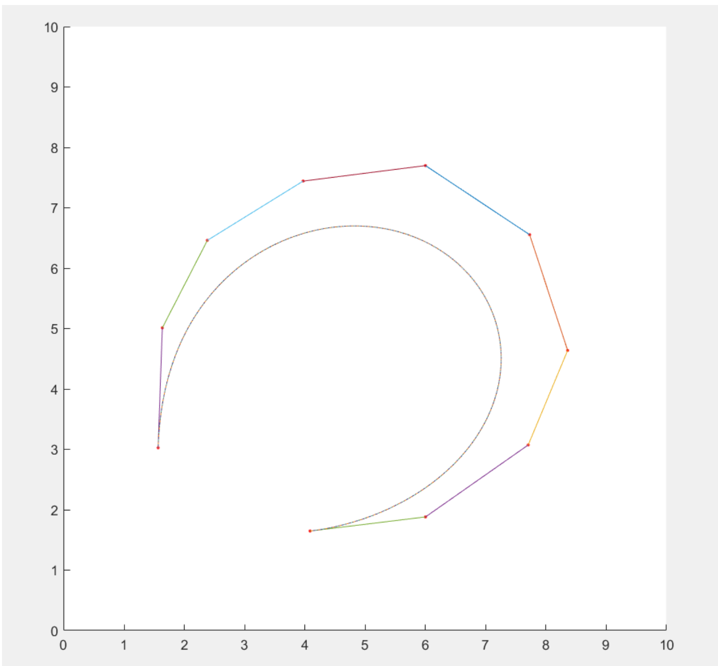

Figure 13. The Bézier curve of a set of points entered by the mouse.

Remark 7.

We notice that the bézier curve is tangent to the segment that defines the first two points and the last two points of the cloud of points of the input.

Proof. We propose a demonstration in the general case

Let $d \geqslant 2, n \in \mathbb{N}^{\star}$ and $P_0, \ldots, P_n \in \mathbb{R}^d$

The Bézier curve of $P_0, \dots, P_n$ is defined for all $u \in [0,1]$ by

$$\mathcal{C}(u) = \sum_{i=0}^{n} B_{n,i}(u) P_i \text{ with } B_{n,i}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

$u \longmapsto \mathcal{C}(u)$ is derivable and we have for all $u \in [0,1]$

$$\begin{aligned} \frac{d\mathcal{C}(u)}{du} &= \sum_{i=0}^{n} \frac{dB_{n,i}(u)}{du} P_i \\ &= n! \sum_{i=1}^{n} \frac{1}{(i-1)!(n-i)!} i u^{i-1} (1-u)^{n-i} P_i - n! \sum_{i=0}^{n-1} \frac{1}{i!(n-i-1)!} u^i (1-u)^{n-i-1} P_i \end{aligned}$$

So for $u=0$ and $u=1$ we have

$$\left.\frac{d\mathcal{C}(u)}{du}\right|_{u=0} = n\,(P_0 - P_1) \text{ and } \left.\frac{d\mathcal{C}(u)}{du}\right|_{u=1} = n\,(P_n - P_{n-1})$$

Therefore $\left.\frac{d\mathcal{C}(u)}{du}\right|_{u=0}$ is proportional to $P_0 - P_1$, and $\left.\frac{d\mathcal{C}(u)}{du}\right|_{u=1}$ is proportional to $P_n - P_{n-1}$

This ends the proof. □