Grenoble INP UGA
ENSIMAG
Modélisation mathématique, Image et Simulation

Modélisation Géométrique

Rendu Mini-Projet noté

Réalisé par :
Mohammed Reda Belfaida

Année : 2024/2025

1 Objectif du Mini-Projet

L'objectif de ce mini-projet est de concevoir et d'implémenter un programme interactif permettant de tracer des courbes de splines Hermite cubiques, en explorant les propriétés géométriques et qualitatives de ces courbes. Ce projet comporte deux volets principaux :

1. **Construction et Visualisation de Splines Hermite Cubiques** :
   - À partir d'un ensemble de points donnés, créer une courbe spline Hermite cubique qui interpole ces points aux paramètres spécifiés.
   - Les splines Hermite sont définies par des segments de courbes polynomiales de degré 3 qui assurent la continuité de la courbe et de ses dérivées premières. Ces courbes C^1 sont construites en utilisant les points de contrôle P_k et des tangentes m_k associées.
   - Un paramètre de tension c permet de contrôler les tangentes pour ajuster la continuité et la forme de la courbe, influençant ainsi sa "qualité" visuelle.
2. **Comparaison avec d'Autres Méthodes d'Interpolation** :
   - Une analyse comparative est menée entre les splines Hermite et d'autres méthodes d'interpolation, notamment l'interpolation polynomiale (Lagrange) et les splines cubiques C^2. Cette comparaison vise à évaluer la capacité de chaque méthode à générer une courbe lisse, naturelle et fidèle aux points d'interpolation.
   - L'utilisateur pourra visualiser et comparer ces courbes, observer l'effet de la variation des tangentes, et étudier la courbure de la spline Hermite, un indicateur de sa qualité géométrique.

En développant ce programme, l'objectif final est de fournir un outil permettant de mieux comprendre les caractéristiques des courbes paramétriques et de tester diverses configurations de tangentes et de paramètres pour optimiser l'apparence et la précision de la courbe spline Hermite.

2 Première partie

L'énoncé commence par définir les polynômes d'interpolation d'Hermite pour deux points P_i et P_{i+1} et deux tangentes m_i et m_{i+1} dans $\mathbb{R}^2$, aux paramètres u_i et u_{i+1} :

$$P_{/[u_i\,;u_{i+1}]}(u) = P_i H_0(t) + P_{i+1} H_1(t) + (u_{i+1} - u_i) m_i H_2(t) + (u_{i+1} - u_i) m_{i+1} H_3(t)$$

pour tout $u \in [u_i\,;u_{i+1}]$ et $t = \frac{u-u_i}{u_{i+1}-u_i}$ et $i \in [1\,;N]$

2.1 Démonstration

0. On a

$$P_{/[u_i\,;u_{i+1}]}(u_i) = P_iH_0(0) + P_{i+1}H_1(0) + (u_{i+1} - u_i)m_iH_2(0) + (u_{i+1} - u_i)m_{i+1}H_3(0)$$

Donc :

$$P_{/[u_i\,;u_{i+1}]}(u_i) = P_i$$

On a de même :

$$P_{/[u_i\,;u_{i+1}]}(u_{i+1}) = P_iH_0(1) + P_{i+1}H_1(1) + (u_{i+1} - u_i)m_iH_2(1) + (u_{i+1} - u_i)m_{i+1}H_3(1)$$

Donc :

$$P_{/[u_i\,;u_{i+1}]}(u_{i+1}) = P_{i+1}$$

Car :

$$H_0(0) = 1 \quad H_1(0) = 0 \quad H_2(0) = 0 \quad H_3(0) = 0$$
$$H_0(1) = 0 \quad H_1(1) = 1 \quad H_2(1) = 0 \quad H_3(1) = 0$$

D'autre part, on a pour $u \in [u_i\,;u_{i+1}]$:

$$P'_{/[u_i\,;u_{i+1}]}(u) = \frac{\mathrm{d}t}{\mathrm{d}u}\frac{\mathrm{d}}{\mathrm{d}t}P_{/[u_i\,;u_{i+1}]}(t)$$

Donc :

$$P'_{/[u_i\,;u_{i+1}]}(u) = \frac{1}{u_{i+1} - u_i}(P_iH'_0(t)+P_{i+1}H'_1(t)+(u_{i+1}-u_i)m_iH'_2(t)+(u_{i+1}-u_i)m_{i+1}H'_3(t))$$

D'où :

$$P'_{/[u_i\,;u_{i+1}]}(u_i) = m_i$$
$$P'_{/[u_i\,;u_{i+1}]}(u_{i+1}) = m_{i+1}$$

Car :

$$H'_0(0) = 0 \quad H'_1(0) = 0 \quad H'_2(0) = 1 \quad H'_3(0) = 0$$
$$H'_0(1) = 0 \quad H'_1(1) = 0 \quad H'_2(1) = 0 \quad H'_3(1) = 1$$

Le résultat en découle.

1. On garde les mêmes notations ,et on se donne une distribution équidistante des $(u_i)_{1\leq i\leq N}$, soit i dans $[1\,;N]$ et on propose la forme de Bézier équivalente qui correspond à :

$$P_{/[u_i\,;u_{i+1}]}(u) = x_i(t) = \sum_{j=0}^{3} b_{3i+j}B_j^3(t)$$

On a :

$$P_{/[u_i\,;u_{i+1}]}(u_i) = x_i(0) = b_{3i}$$
$$P_{/[u_i\,;u_{i+1}]}(u_{i+1}) = x_i(1) = b_{3i+3}$$

D'après ce qui précède, on a donc :

$$b_{3i} = P_i$$
$$b_{3i+3} = P_{i+1}$$

D'autre part, on a :

$$P'_{/[u_i\,;u_{i+1}]}(u) = \frac{\mathrm{d}t}{\mathrm{d}u} x'_i(t)$$

D'où :

$$P'_{/[u_i\,;u_{i+1}]}(u) = \frac{3}{u_{i+1} - u_i} \sum_{j=0}^{2} \Delta b_{3i+j} B_j^2(t) = 3 \sum_{j=0}^{2} \Delta b_{3i+j} B_j^2(t)$$

Donc :

$$P'_{/[u_i\,;u_{i+1}]}(u_i) = 3\Delta b_{3i} = 3(b_{3i+1} - b_{3i})$$
$$P'_{/[u_i\,;u_{i+1}]}(u_{i+1}) = 3\Delta b_{3i+2} = 3(b_{3i+3} - b_{3i+2})$$

Or d'après ce qui précède :

$$P'_{/[u_i\,;u_{i+1}]}(u_i) = m_i$$
$$P'_{/[u_i\,;u_{i+1}]}(u_{i+1}) = m_{i+1}$$

Par conséquent, on obtient :

$$b_{3i+1} = P_i + \frac{m_i}{3}$$
$$b_{3i+2} = P_{i+1} - \frac{m_{i+1}}{3}$$

Finalement, les coefficients de la forme de bézier équivalente sont données par :

$$\begin{cases} b_{3i} = P_i \\ b_{3i+3} = P_{i+1} \\ b_{3i+1} = P_i + \frac{m_i}{3} \\ b_{3i+2} = P_{i+1} - \frac{m_{i+1}}{3} \end{cases}$$

2. La figure suivante représente un dessin pour deux polygones de contrôle consécutifs avec leurs éventuels points de contrôle et leurs tangentes :

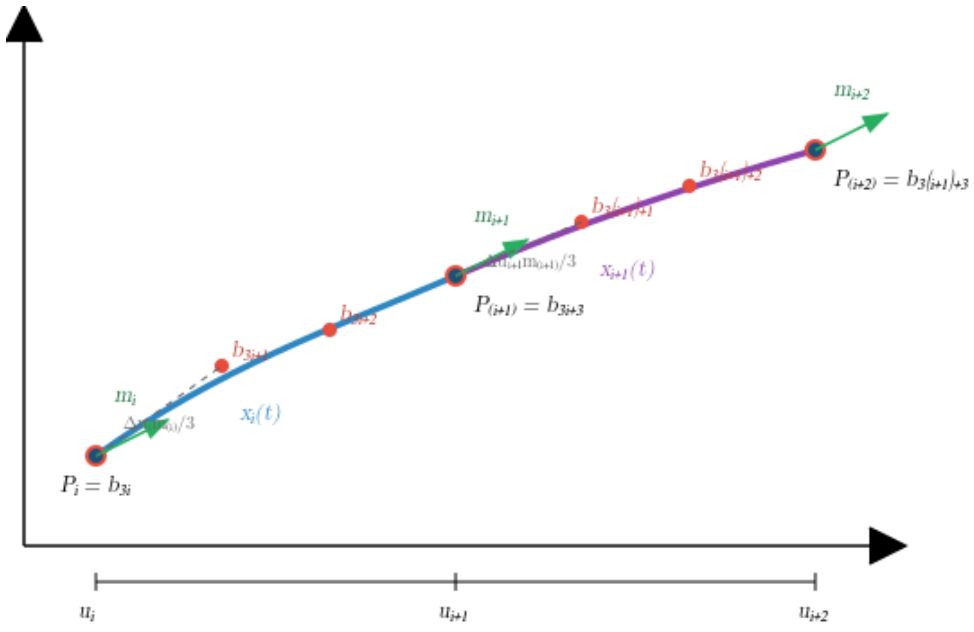

Figure 1: Deux polygones de contrôle consécutifs

L'énoncé utilise pour l'estimation **Cardinal splines** des tangentes définie par :

$$m_k = (1-c)\frac{P_{k+1} - P_{k-1}}{u_{k+1} - u_{k-1}} \quad \forall k \in [1\,;N-1] \quad \forall c \in [0\,;1]$$

Vu que les $(u_k)_{1 \leq k \leq N}$ forme une distribution équidistante, on a donc :

$$m_k = (1-c)\frac{P_{k+1} - P_{k-1}}{2} \quad \forall k \in [1\,;N-1] \quad \forall c \in [0\,;1]$$

3. D'abord il se sied de justifier le choix d'une telle estimation pour les tangentes. Pour comprendre pourquoi cette estimation est sensée, on peut réfléchir de plusieurs façons :

- 1. D'un point de vue géométrique : la tangente m_i représente le **taux de variation instantané** au point P_i. En prenant $P_{i+1} - P_{i-1}$, on estime ce taux de variation en regardant comment les points varient "autour" de P_i. Le facteur c permet d'ajuster l'amplitude de cette variation
- 2. D'un point de vue analytique, on peut le voir comme une approximation centrée de la dérivée : Si on imagine que nos points P_i sont des échantillons d'une fonction f aux points u_i.

L'illustration suivante permet de mettre en lumière ces propos :

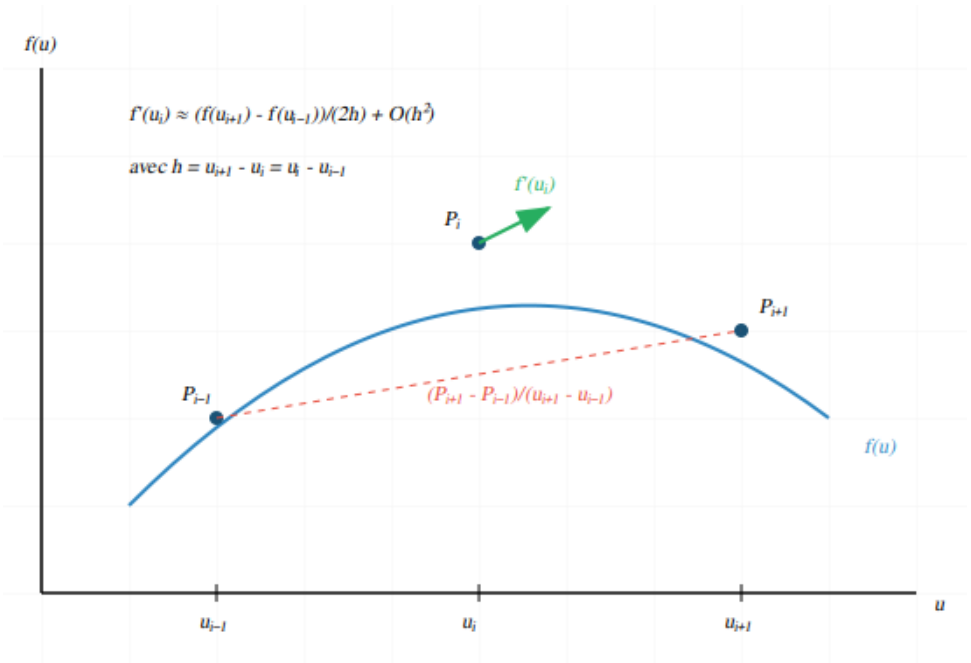

Figure 2: Méthode d'estimation de la tangente

Ce choix ne s'applique pas aux points aux extrémités, ce qui nécessite une approche cohérente avec la logique développée ci-dessous. Par conséquent, nous estimerons la tangente m_0 en utilisant la dérivée à droite et la tangente m_N en utilisant la dérivée à gauche. Ainsi, nous procéderons de la manière suivante :

$$m_0 = (1-c)(P_1 - P_0)$$
$$m_N = (1-c)(P_N - P_{N-1})$$

4. L'implémentation dans ce cadre sera réalisée en **Python**. La visualisation s'effectue via le script **main.py**. Une fois ce script exécuté, une fenêtre Matplotlib s'ouvre pour tracer la courbe, accompagnée d'un curseur permettant d'observer l'effet du coefficient caractéristique c sur le spline d'Hermite.

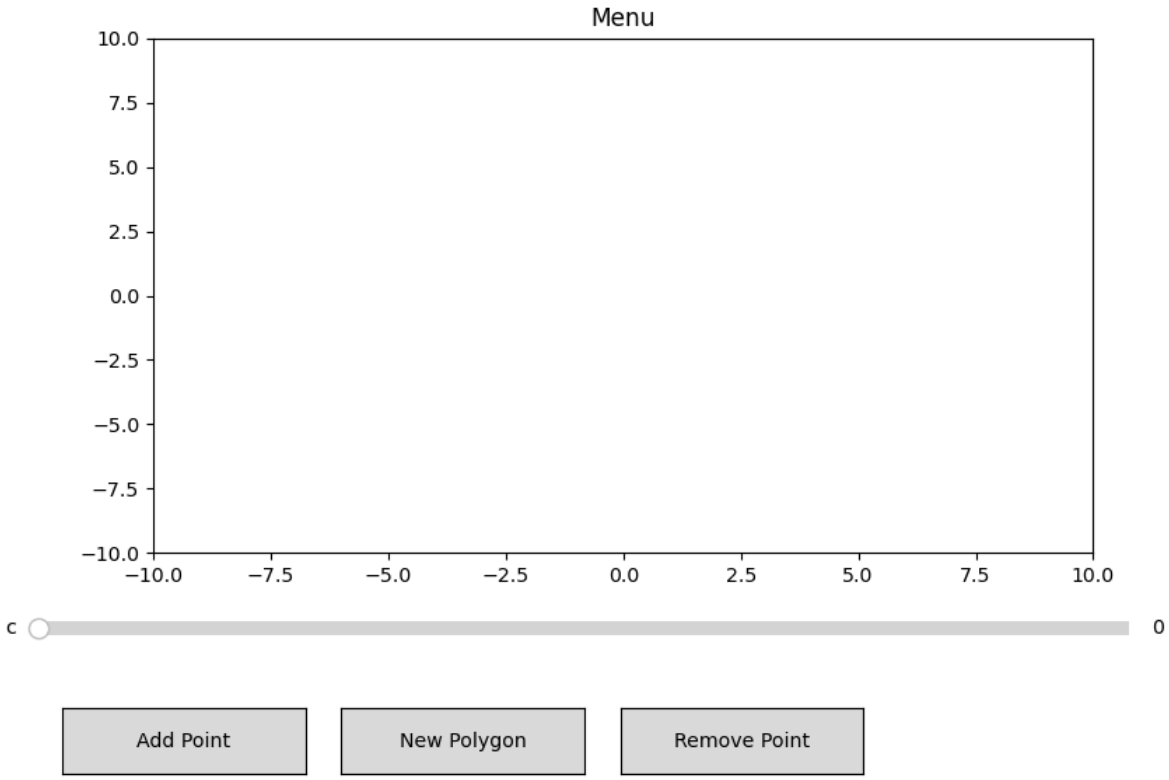

Figure 3: Interface du dessin principale

4.1.

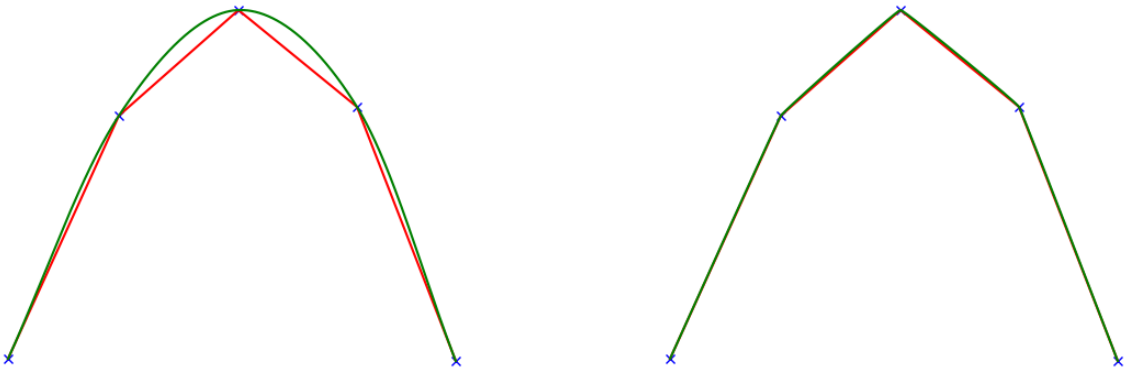

Figure 4: Comparaison des splines cardinales avec différents coefficients de tension

Ces deux images montrent des interpolations réalisées avec des splines de type Hermite (ou splines cardinales), où le paramètre de tension c influence la forme des courbes.

- 1. Image à gauche ($c = 0$) : Dans cette configuration, le coefficient de tension c est nul, ce qui donne des tangentes plus longues, et la courbe est plus lissée entre les points de contrôle. La spline effectue une transition douce et incurvée d'un point à l'autre, créant un effet de courbure accentué. Cela produit une courbe fluide qui suit de manière plus progressive les points de contrôle.

- 2. Image à droite ($c = 0.9$) : La courbe est tendue et suit une trajectoire presque rectiligne entre les points de contrôle. Ici, le coefficient de tension c est élevé (proche de 1), ce qui fait que les tangentes sont courtes et que la courbe se rapproche d'une interpolation linéaire entre les points de contrôle. Cela se traduit par une transition plus rigide, où la courbe est moins lisse et se rapproche davantage des segments reliant directement les points.

En effet, le coefficient de tension c intervient dans la formule d'estimation des tangentes des splines cardinales. Lorsque c est faible (proche de 0), les tangentes aux points de contrôle sont influencées par les segments adjacents, ce qui permet de créer une courbure plus marquée. En augmentant c, les tangentes deviennent plus courtes, et la courbe tend à suivre un tracé plus direct entre les points, diminuant l'effet de courbure et produisant une forme plus tendue.

4.2

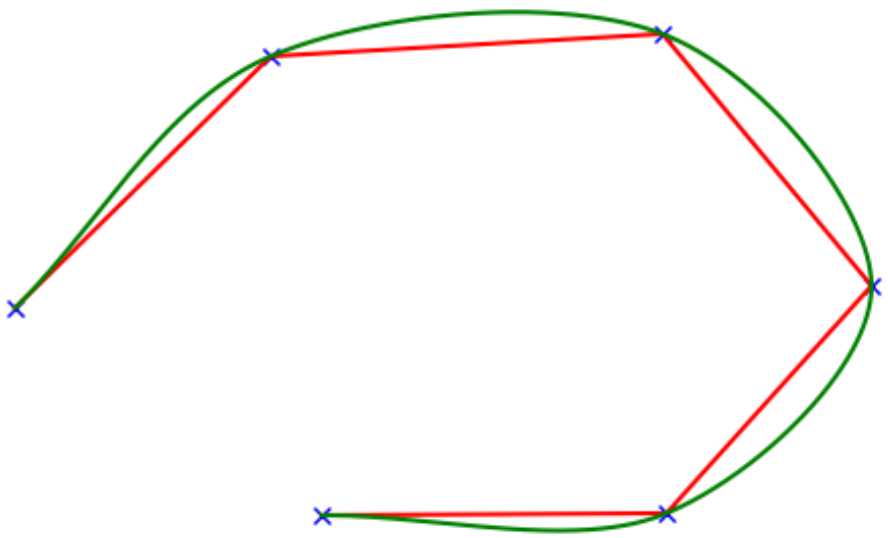

Figure 5: Exemple d'une Hermite Spline

Pour évaluer la qualité du résultat obtenu avec les splines cardinales, plusieurs aspects sont à considérer en fonction de l'objectif d'interpolation souhaité. Voici quelques éléments d'analyse :

- 1. **Lissage de la courbe** : Les splines cardinales sont généralement conçues pour produire des courbes lisses, c'est-à-dire sans ruptures de continuité dans la pente ou

la courbure. Dans les exemples fournis (les images avec $c = 0$ et $c = 0.9$), on observe que la courbe est lisse dans les deux cas. Cependant, la courbure peut varier en fonction du coefficient de tension c. Pour des valeurs de c proches de 1, la courbe tend à se rapprocher des segments droits reliant les points d'interpolation, ce qui diminue la **souplesse** de la spline. À l'inverse, des valeurs de c plus basses entraînent une courbure plus marquée, mais aussi un risque de formation d'ondulations indésirées.

- 2. **Présence d'ondulations non désirées** : Des ondulations indésirées peuvent apparaître lorsque le coefficient c est trop faible, car la courbe cherche à **lisser** le passage entre chaque point de contrôle, ce qui peut entraîner des dépassements ou des oscillations autour des points d'interpolation. Sur l'image où $c = 0$, la courbe semble un peu plus courbée, ce qui peut générer des ondulations si les points d'interpolation sont trop rapprochés. En revanche, pour $c = 0.9$, la courbe suit plus fidèlement les segments reliant les points, réduisant le risque d'ondulations.

- 3. **Préservation de la forme initiale** : Une spline de bonne qualité devrait préserver la forme globale décrite par les points d'interpolation. Par exemple, pour un ensemble de points formant un polygone convexe, la spline devrait aussi être convexe et ne pas créer d'excursions non désirées à l'extérieur des segments définis par les points de contrôle. Dans les exemples fournis, pour $c = 0.9$, la courbe semble bien respecter l'enveloppe convexe formée par les points. Cependant, pour des valeurs plus faibles de c, la spline peut avoir tendance à **déborder** en dehors de cette enveloppe convexe, créant des bosses ou des ondulations qui altèrent la forme générale.

- 4. **Sensibilité au coefficient de tension** c : Le paramètre c contrôle l'intensité de la courbure entre les points d'interpolation. Des valeurs proches de 0 donnent une courbe très lisse, mais peuvent introduire des oscillations, tandis que des valeurs proches de 1 tendent à rendre la courbe plus tendue et proche des segments droits. Cela signifie que le choix de c doit être fait en fonction des exigences d'interpolation; pour une courbe douce et régulière, des valeurs faibles de c conviennent, mais si la fidélité à la forme polygonale est essentielle, des valeurs plus élevées de c seront préférables.

4.3. (En option) Le choix des tangentes est curcial car pour $k \in [1\,;N-1]$, et pour un intervalle donné $[u_k\,;u_{k+1}]$ les coeffcients b_{3k+1} et b_{3k+2}, donc influe sur la construction du Hermite Spline. D'où l'intérêt de faire un choix raisonable des tangentes. D'autres choix d'estimation des tangentes peuvent être envisageables, dans cette partie on se limitera sur deux méthodes d'estimation des tangentes :

- 1. **Generalized Catmull–Rom spline :**

$$m_k = \frac{P_{k+1} - P_{k-1}}{\alpha} \quad \forall k \in [1\,;N-1]$$

On gardera le même choix pour les tangentes m_0 et m_N, ce qui revient à considérer:

$$m_0 = \frac{P_1 - P_0}{\alpha}$$

$$m_N = \frac{P_N - P_{N-1}}{\alpha}$$

avec $\alpha \in [0\,;1]$

- 2. **Estimation d'ordre 2 :**

$$m_k = \frac{4P_{k+1} - 3P_k + P_{k+2}}{2} \quad \forall k \in [0\,;N-2]$$

et aux deux derniers points :

$$m_{N-1} = \frac{P_N - P_{N-2}}{2}$$

$$m_N = P_N - P_{N-1}$$

Dans cette version, la tangente m_k est calculée en tenant compte d'une approximation de second ordre, en utilisant les points voisins P_{k-1}, P_k, et P_{k+1}. Cette approche permet une estimation plus lisse et précise des tangentes par rapport à une interpolation linéaire simple. Cependant, comme cette formule ne peut pas s'appliquer aux tangentes aux extrémités, des estimations spéciales sont faites pour les dernières tangentes m_{N-1} et m_N, en utilisant une dérivée centrée pour l'avant-dernière tangente et une dérivée gauche pour la dernière.

Le script utilisé pour effectuer la comparaison entre les splines d'Hermite est intitulé **comparaison.py**. La courbe en magenta représente la spline obtenue en utilisant le choix numéro 2, tandis que la courbe en jaune correspond à celle obtenue avec le choix numéro 1 avec $\alpha = 0.5$.

Figure 6: Comparaison des splines avec différentes estimations des tangentes

5. Le script **courbure.py** permet de tracer la courbure d'une Hermite Spline. Il fait appel au script **comparaison.py**, et l'utilisateur peut choisir les versions souhaitées en appelant les fonctions appropriées dans les méthodes définies de la classe **Index**. L'utilisateur entre ensuite les points de contrôle et le script trace les Hermite Splines selon les versions sélectionnées. Après cela, la fenêtre graphique se ferme pour laisser place à une nouvelle fenêtre affichant la courbure.

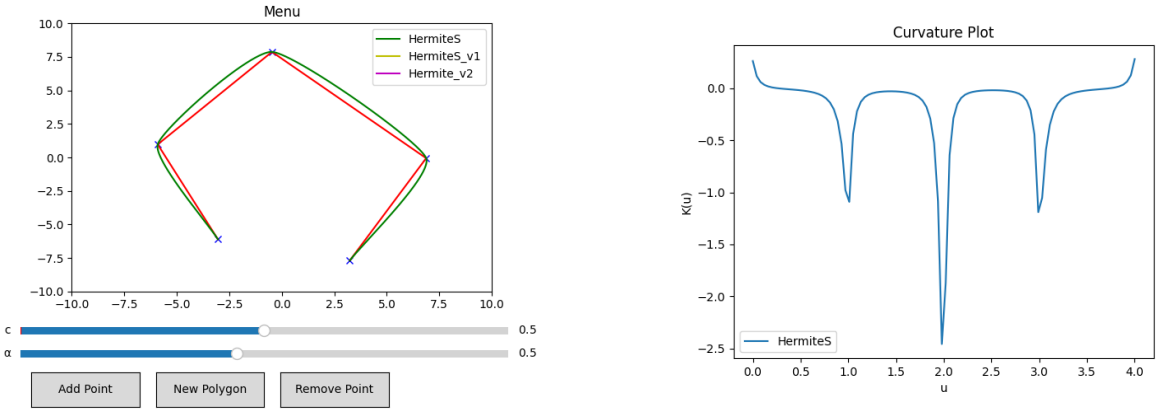

Figure 7: Courbure d'une Hermite Spline standard avec $c = 0.5$

5.1. les points d'interpolation correspondent souvent à des points où la courbure de la courbe change de manière significative, ce qui peut être interprété comme des points critiques dans la fonction de la courbure. Ces points peuvent correspondre à des maxima, minima ou points d'inflexion, où la concavité de la courbe change.

Quant à la question de savoir pourquoi un plot de courbure est un bon indicateur de qualité d'une courbe, voici quelques éléments de réponse :

- 1. **Comportement de la courbure** : La courbure d'une courbe indique comment la courbe se plie et change de direction. Une courbure trop élevée ou trop faible dans certaines régions pourrait signaler des irrégularités ou des oscillations indésirables dans la courbe, qui affectent sa fluidité et sa stabilité.

- 2. **Lissage et régularité** : Une courbe avec une courbure bien distribuée et sans pics ou creux marqués est généralement plus régulière et lisse. Cela est essentiel pour des applications comme l'approximation de trajectoires ou l'interpolation, où une courbe trop rigide ou "accrochée" pourrait être moins fiable ou peu esthétique.

- 3. **Représentation de la qualité de l'interpolation** : Les points d'interpolation sont censés correspondre à des points où la courbure ne présente pas de discontinuité importante. Un bon ajustement des splines doit produire une courbure qui est relativement lisse et continue à ces points. Si la courbure varie brusquement autour des points d'interpolation, cela peut indiquer un mauvais ajustement ou un problème avec la méthode d'interpolation utilisée.

Dans notre cas, un plot de courbure est un indicateur utile de la qualité d'une courbe car il permet de visualiser comment la courbe change localement, de repérer d'éventuelles anomalies et de vérifier la régularité de l'ajustement entre les points d'interpolation. Ce

qui en résulte c'est que nous sommes capables de visualiser surtout la **qualité de raccordement** dans les points de contrôle. Tant que la courbure des points de raccordement est proche de 0, la qualité est elevée, cela permet de conclure sur le caractère C^1 de note Hermite Spline.

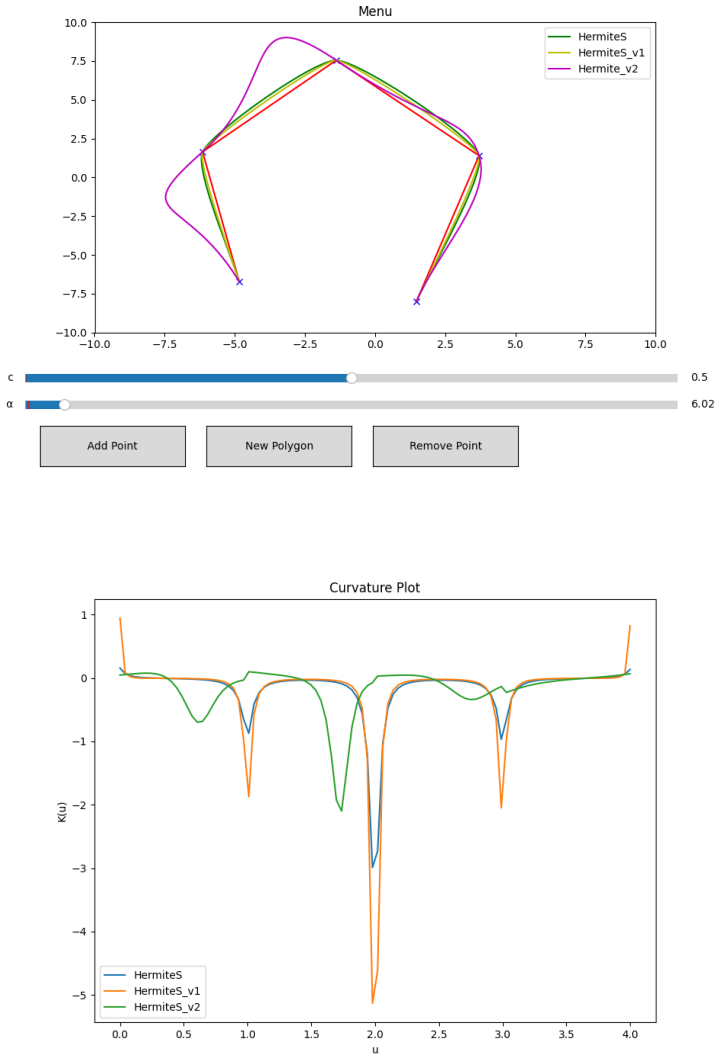

Figure 8: Analyse comparative des courbures avec $\alpha = 6.02$ et $c = 0.5$

5.2. Dans cette question, il faudra comparer entre les différentes méthodes.

1. **HermiteS (standard, c=0.5)** :
   - Présente des pics de courbure nets mais modérés.
   - Démontre une variation relativement douce de la courbure.
2. **HermiteS_v1** ($\alpha = 6.02$) :
   - Exhibe des pics de courbure plus prononcés.
   - Les variations sont plus brusques et plus importantes (notamment vers $u = 2.0$ où la courbure atteint -5).

- Présente des oscillations plus marquées.

3. **HermiteS_v2 (estimation d'ordre 2)** :
   - Montre des pics de courbure plus répartis et moins profonds.
   - Les variations sont globalement plus douces.
   - Démontre un comportement plus régulier.

Évaluation des améliorations

- La version v1 ($\alpha = 6.02$) semble dégrader la qualité car : Elle introduit des variations de courbure plus importantes et plus abruptes.
- La version v2 (estimation d'ordre 2) présente une amélioration car : Elle produit des transitions plus douces et des pics de courbure moins prononcés.

Conclusion

- La méthode d'estimation d'ordre 2 (v2) apparaît comme la meilleure amélioration.
- La version v1 avec $\alpha = 6.02$ n'est pas recommandée en raison de l'augmentation des variations brusques de courbure.
- Il est conseillé d'utiliser soit la version standard soit la version v2, selon les besoins spécifiques de lissage.

6. On a choisi de dessine deux formes qui ressemblent à un signe d'amour par une interpolation d'Hermite

Figure 9: Signes d'amour tracés par le script **main.py**

3 Deuxième partie

7a. L'algorithme développé repose sur les étapes suivantes :

1. **Initialisation** : Les points de contrôle sont donnés sous forme d'une matrice Polygon de taille $2 \times (N+1)$, où la première ligne contient les abscisses u_i et la seconde les ordonnées y_i.

2. **Évaluation des** $L_i(u)$: Pour chaque i, on calcule $L_i(u)$ en excluant u_i du produit.

3. **Construction du polynôme** : Les coordonnées interpolées sont calculées en additionnant les contributions de chaque point de contrôle :

$$x(u) = \sum_{i=0}^{N} x_i \cdot L_i(u), \quad y(u) = \sum_{i=0}^{N} y_i \cdot L_i(u).$$

   Remarque : Dans l'algorithme u joue le rôle de l'intervalle $[0\,;N]$, tandis que la variable u des fonctions x et y est représenté par un élément de la liste numpy (numpy array) t.

4. **Mise à jour graphique** : Les points calculés sont ajoutés aux données existantes du tracé pour générer une courbe continue.

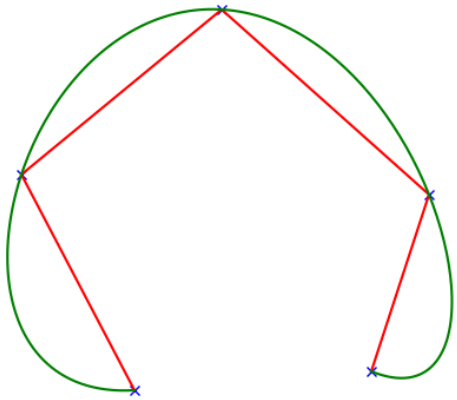

Figure 10: Exemple d'une interpolation de Lagrange

Le code suivant implémente l'algorithme d'interpolation de Lagrange et trace la courbe interpolée.

```
import numpy as np
import matplotlib.pyplot as plt
# curve est considée comme une variable globale

def PlotLagrangeCurve(Polygon):
    """Trace une courbe interpolée à l'aide de la méthode de Lagrange."""
    N = len(Polygon[0, :]) - 1
    t = np.linspace(0, N, 200)
    u = np.arange(N + 1)

    x_interp = np.zeros_like(t)
    y_interp = np.zeros_like(t)
```

```
    for i in range(N + 1):
        L_i = np.ones_like(t)
        for j in range(N + 1):
            if i != j:
                L_i *= (t - u[j]) / (u[i] - u[j])
        x_interp += Polygon[0, i] * L_i
        y_interp += Polygon[1, i] * L_i

    xdata = np.concatenate((curve.get_xdata(), x_interp))
    ydata = np.concatenate((curve.get_ydata(), y_interp))
    curve.set_xdata(xdata)
    curve.set_ydata(ydata)
    plt.draw()
```

7b. (En option) Pour implémenter les splines cubiques C^2, on utilise l'algorithme de raccordement couplé avec l'usage de la forme de Bezier-Hermite. L'implémentation de cet algorithme est décrite par la fonction **PlotSplineCubique**.

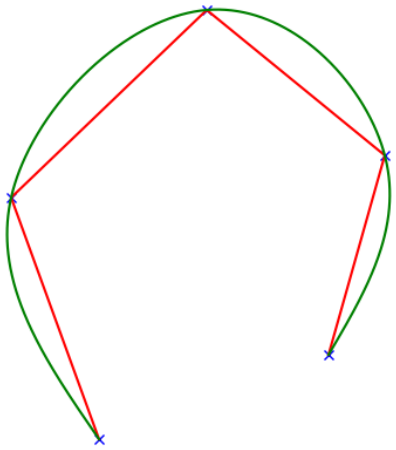

Figure 11: Exemple d'une Spline cubique C^2 interpolante

```
def derive_cubique(matrice):

    # Get number of columns (N>2)
    N = matrice.shape[1]
    col = np.zeros((matrice.shape[0], N))
    col[:, 0] = 3 * (matrice[:, 1] - matrice[:, 0])
    col[:, 1:N-1] = 3 * (matrice[:, 2:N] - matrice[:, 0:N-2])
    col[:, N-1] = 3 * (matrice[:, N-1] - matrice[:, N-2])

    # Create diagonal matrices
    # Lower diagonal
    matrice1 = np.diagflat(np.ones(N-1), -1)
```

```
    # Upper diagonal
    matrice2 = np.diagflat(np.ones(N-1), 1)
    # Main diagonal (multiplied by 4)
    matrice3 = 4 * np.eye(N)
    matrice3[0, 0] = 2
    matrice3[N-1, N-1] = 2

    # Sum the matrices
    A = matrice1 + matrice2 + matrice3

    # Solve the system and transpose
    derives = np.linalg.solve(A, col.T).T

    return derives

def PlotSplineCubique(Polygon):
    n = Polygon.shape[1]  # Nombre de points de contrôle
    derivees = derive_cubique(Polygon)
    for i in range(n - 1):
        tang1 = derivees[:,i]
        tang2 = derivees[:,i+1]
        bezier_points = np.array([
            Poly[:, i],
            Poly[:, i] + tang1 / 3,
            Poly[:, i + 1] - tang2 / 3,
            Poly[:, i + 1]
        ]).T
        PlotBezierCurve(bezier_points)
```

Détails de l'algorithme

- 1. Calcul des dérivées (`derive_cubique`)
  - Le système linéaire est construit à partir des différences entre les points de contrôle.
  - Une matrice tridiagonale est générée pour représenter les contraintes de continuité.
  - Le système est résolu à l'aide de `np.linalg.solve`.

- 2. Construction des courbes de Bézier (`PlotSplineCubique`)
  - Les points de contrôle intermédiaires sont calculés à partir des dérivées.
  - Une courbe de Bézier est tracée pour chaque segment entre deux points de contrôle.

4 Analyse Comparative entre les différentes méthodes d'interpolation

4.1 Courbure (première image)

1. **Hermite Spline** :
   La courbure semble présenter des variations importantes, notamment des pics qui traduisent une forte dérivée seconde. Cela peut indiquer que les tangentes imposées

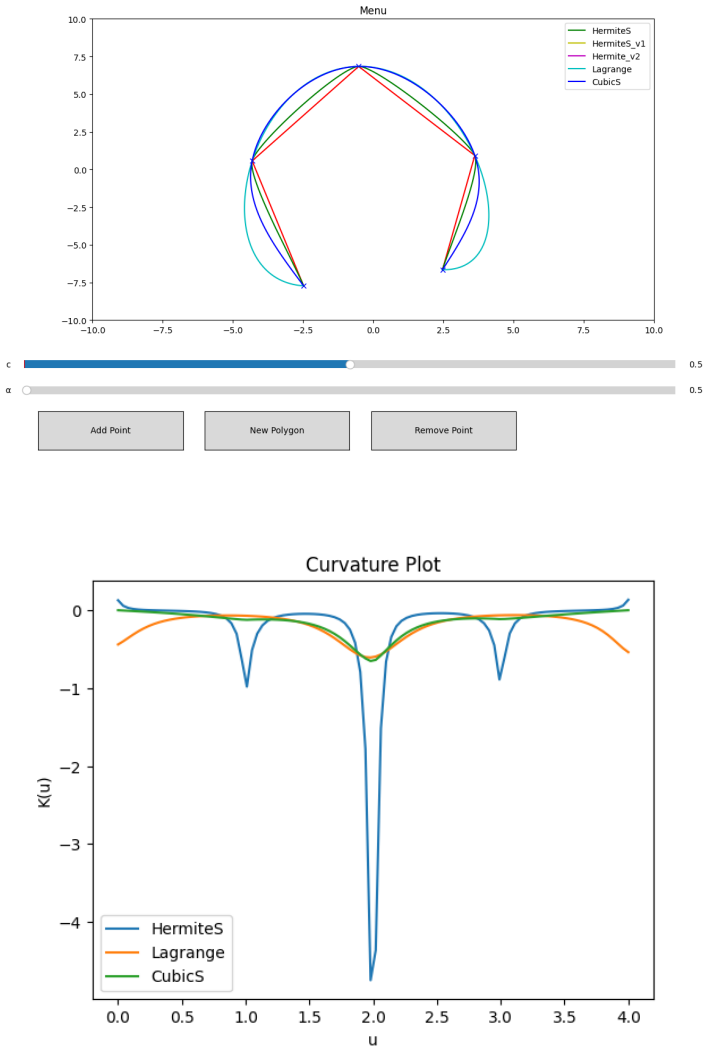

Figure 12: Analyse comparative des courbures

influencent fortement la forme de la courbe, ce qui rend cette méthode très flexible, mais potentiellement plus instable dans certains cas.

2. **Lagrange Interpolation** :
   La courbure de l'interpolation de Lagrange est plus régulière et relativement douce comparée à Hermite. Cela est dû à la continuité C^0 et au fait qu'elle ne prend pas en compte de contraintes sur les dérivées. Cependant, pour des interpolations globales avec un grand nombre de points, l'interpolation de Lagrange peut souffrir du phénomène de Runge.

3. **Cubic Spline** :
   La courbure est encore plus lisse, en raison de la continuité C^2 assurée par les splines cubiques. Cela permet une transition fluide entre les segments interpolés, réduisant les variations brutales de la courbure.

4.2 Forme (deuxième image)

1. **Hermite Spline** :
   Hermite offre plus de contrôle en permettant de spécifier les tangentes à chaque point. Cela peut produire des courbes personnalisées, mais peut aussi introduire des oscillations ou des courbures abruptes si les tangentes ne sont pas bien choisies.

2. **Lagrange Interpolation** :
   L'interpolation de Lagrange passe par tous les points donnés. Cependant, elle peut produire des oscillations inutiles (phénomène de Runge) si le nombre de points est élevé ou si les points ne sont pas bien répartis.

3. **Cubic Spline** :
   Les splines cubiques produisent des courbes naturelles et régulières, adaptées pour des applications où une transition fluide est essentielle, comme en modélisation géométrique ou animation.

4.3 Conclusions

- **Hermite Spline** est utile lorsque le contrôle des tangentes est nécessaire pour des applications spécifiques, mais il faut veiller à éviter des choix de tangentes qui introduisent une instabilité.

- **Lagrange Interpolation** est simple et directe, mais elle est plus adaptée pour un petit nombre de points. Elle devient problématique pour des distributions de points non uniformes ou un grand nombre de points.

- **Cubic Spline** est généralement la meilleure option pour obtenir des courbes douces et naturelles, particulièrement pour des applications où la continuité C^2 est cruciale.

Ces choix dépendent largement de l'application et des besoins en termes de continuité, contrôle et stabilité de la courbe.