Le but de ce TD est d'étudier différentes méthodes d'interpolation par des courbes "splines".

Rappel : toute fonction écrite doit être testée plusieurs fois, avec des données produisant un résultat contrôlable.

I Paraboles

Soient A, B deux points d'abscisses distinctes. On cherche les coefficients a, b, c d'une parabole d'équation $y = a + bx + cx^2$ passant par A et B. Cette condition étant insuffisante pour déterminer une unique parabole, on impose en plus que la pente de la tangente en A soit $p \in \mathbb{R}$ fixé.
Remarque : on cherche a, b, c dans cet ordre, pour obtenir des résultats compatibles avec le TD sur les polynômes.

1.1 Déterminer les coefficients

Ecrire en fonction des coordonnées de A et B et de p le système linéaire dont (a, b, c) est une solution.
Créer ensuite une fonction `coefficients(A, B, p)` qui prend deux listes A, B de longueur 2 et p un nombre comme arguments et retourne le triplet (a, b, c).
On pourra utiliser

```
np.linalg.solve(M, Y)
```

qui retourne la seule colonne X telle que MX = Y où M est une matrice carrée et Y la colonne second membre.
Pour tester votre fonction, on pourra utiliser la parabole $y = x^2 - x$ qui passe par les points de coordonnées $(0, 0)$ et $(1, 0)$ avec une tangente de pente -1 en $(0, 0)$.

1.2 Tracé

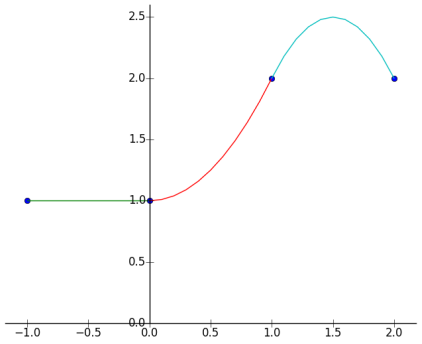
Ecrire une fonction `evalue(L, x)` qui retourne la valeur de $P(x)$ où P est le polynôme dont les coefficients sont donnés par la liste L ie $P(X) = \sum_{i=0}^{\mathbf{len}(L)-1} L[i]X^i$.
Créer ensuite une fonction `parabole(A, B, p)` qui trace la parabole cherchée dans le préambule (entre les points A et B). A, et B ont la même signification que précédemment. On utilisera 30 points en tout pour effectuer le tracé.

II Splines quadratiques

2.1 Présentation du problème

On se donne maintenant une liste de points `Pts =` $[A_0, \dots, A_n]$ (ou un point est toujours une liste de longueur 2) ainsi qu'un nombre p. On suppose que les points sont triés par abscisses strictement croissantes.
On souhaite tracé la courbe suivante :

- Entre A_0 et A_1 il s'agit de la parabole de la partie précédente.
- Pour $i \in [\![1, n-1]\!]$, la courbe entre A_i et A_{i+1} est une parabole passant par A_i et A_{i+1} telle que la pente de la tangente en A_i soit la même que la pente en A_i de la courbe précédente.



2.2 Tracé

Il nous manque un outil technique. Créer une fonction `pente_tangente(L, x)` qui retourne la valeur de $P'(x)$ où P est encore représenté par la liste L. On veut ici une fonction générale, c'est à dire que l'on ne suppose pas que le degré de P est 2.

Créer maintenant la fonction `spline(Pts, p)`.
Pour créer la liste de points `Pts` pour le test, on prendra n (à choisir) points sur une courbe représentative de référence, équirépartis (par exemple $\sin, x \mapsto \arctan(x) + \sin(x)$...). On pourra faire de ce processus de création de `Pts` une fonction, afin de changer facilement la valeur de n.
Tracer ensuite la courbe demandée pour différente valeurs de p. On pourra essayer d'imposer la couleur de tracé des arcs de paraboles.

III Splines cubiques

L'idée est exactement la même que précédemment, mais cette fois on cherche des courbes d'équation $y = a+bx+cx^2+dx^3$ et on impose en plus la valeur de la dérivée seconde en chaque point. Plus précisément, les valeurs des deux premières dérivées sont imposées au premier point, puis reportées pas à pas comme pour les splines quadratiques.
Par exemple, la fonction qui remplace `coefficients` pourrait être `coeff_deg3(A, B, p1, p2)` où $p1$ est la pente de la tangente en A et $p2$ la valeur de la dérivée seconde en A.
Pour le tracé, on imposera que la dérivée seconde au premier point est nulle. Par essais successifs (en changeant la pente à l'origine), on tentera d'obtenir une dérivée seconde au dernier point (A_n, pas le dernier point de raccord) la plus proche possible de 0.
Proposer un algorithme de calcul approché de la pente à l'origine pour que la dérivée seconde finale soit nulle.