Interpolation

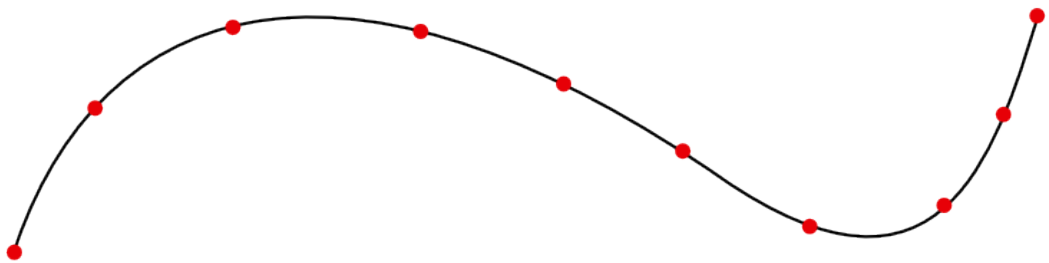
Pascal Romon

Problématique

L'interpolation est une opération mathématique permettant de construire une courbe à partir de la donnée d'un nombre fini de points.

Elle réalise l'intuition visuelle.

Problématique



→ Si on recherche une courbe polynomiale, quel sera son degré minimal?

Polynômes de Lagrange

Introduction :

La méthode de Lagrange permet de calculer un polynôme de degré n passant par $n+1$ points imposés.

Polynômes de Lagrange

Définition :

Soit un ensemble de points $(x_i, y_i),\ 0 \leqslant i \leqslant n$.

On définit les polynômes de Lagrange $L_i(x)$ associés à ces points : L_i s'annule en x_j $(j \neq i)$ et vaut 1 en x_i

$$L_i(x) = \frac{x - x_0}{x_i - x_0}\frac{x - x_1}{x_i - x_1} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}}\frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}$$
$$= \frac{\prod_{k \neq i}(x - x_k)}{\prod_{k \neq i}(x_i - x_k)}$$

Polynômes de Lagrange

Interpolation :

L'unique polynôme d'interpolation $P(x)$ de degré $\leqslant n$ passant par les points $\{(x_i, y_i)\}_{i=1\ldots n}$ est :

$$P(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x).$$

Propriété abstraite : la famille $(L_0, L_1, \ldots, L_n)$ est une base de l'espace vectoriel $\mathbb{R}_n[X]$ des polynômes de degré $\leqslant n-1$.
$\rightarrow$ Quelle est la dimension de cet espace ?

Polynômes de Lagrange

Exemple :

On cherche le polynôme $P(x)$ passant par les 3 points (x_0, y_0), (x_1, y_1), et (x_2, y_2)

$$P(x) = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Polynômes de Lagrange

Exemple :

On cherche $P(x)$ passant par les 3 points (x_1, y_1), (x_2, y_2) et (x_3, y_3)

$$P(x) = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$P(x_0) = y_0 \underbrace{\frac{(x_0 - x_1)(x_0 - x_2)}{(x_0 - x_1)(x_0 - x_2)}}_{1} + y_1 \underbrace{\frac{(x_0 - x_0)(x_0 - x_2)}{(x_1 - x_0)(x_1 - x_2)}}_{0} + y_2 \underbrace{\frac{(x_0 - x_0)(x_0 - x_1)}{(x_2 - x_0)(x_2 - x_1)}}_{0}$$

$$P(x_0) = y_0$$

Polynômes de Lagrange

Exemple :

On cherche $P(x)$ passant par les 3 points (x_0, y_0), (x_1, y_1) et (x_2, y_2)

$$P(x) = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$P(x_1) = y_0 \underbrace{\frac{(x_1 - x_1)(x_1 - x_2)}{(x_0 - x_1)(x_0 - x_2)}}_{0} + y_1 \underbrace{\frac{(x_1 - x_0)(x_1 - x_2)}{(x_1 - x_0)(x_1 - x_2)}}_{1} + y_2 \underbrace{\frac{(x_1 - x_0)(x_1 - x_1)}{(x_2 - x_0)(x_2 - x_1)}}_{0}$$

$$P(x_1) = y_1$$

Polynômes de Lagrange

Exemple :

On cherche $P(x)$ passant par les 3 points (x_0, y_0), (x_1, y_1) et (x_2, y_2)

$$P(x) = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$P(x_2) = y_0 \underbrace{\frac{(x_2 - x_1)(x_2 - x_2)}{(x_0 - x_1)(x_0 - x_2)}}_{0} + y_1 \underbrace{\frac{(x_2 - x_0)(x_2 - x_2)}{(x_1 - x_0)(x_1 - x_2)}}_{0} + y_2 \underbrace{\frac{(x_2 - x_0)(x_2 - x_1)}{(x_2 - x_0)(x_2 - x_1)}}_{1}$$

$$P(x_2) = y_2$$

Polynômes de Lagrange

Exemple :

Le polynôme $P(x)$ de degré $n = 2$ passant par les 3 points (x_0, y_0), (x_1, y_1) et (x_2, y_2) est bien :

$$P(x) = y_0 L_0(x) + y_1 L_1(x) + y_n L_2(x)$$

Un raisonnement abstrait garantit l'unicité de P ($n + 1 =$ nombre de polynômes $=$ dimension de $\mathbb{R}_n[X]$).

Interpolation polynômiale

Point de vue des systèmes linéaires :

On veut trouver le polynôme P de degré n passant par $n+1$ points (x_i, y_i) (tous les x_i sont distincts).

Le polynôme est de la forme :

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

et vérifie :

$$\forall i \in 0 \dots n, \quad P(x_i) = y_i$$

Interpolation polynomiale

Remarque :

L'évaluation d'un polynôme $P(x)$ en x_0 peut s'exprimer comme un produit scalaire

$$P(x_1) = \begin{pmatrix} x_i^n & \cdots & x_i^3 & x_i^2 & x_i & 1 \end{pmatrix} \begin{pmatrix} a_n \\ \vdots \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix}$$

Interpolation polynomiale

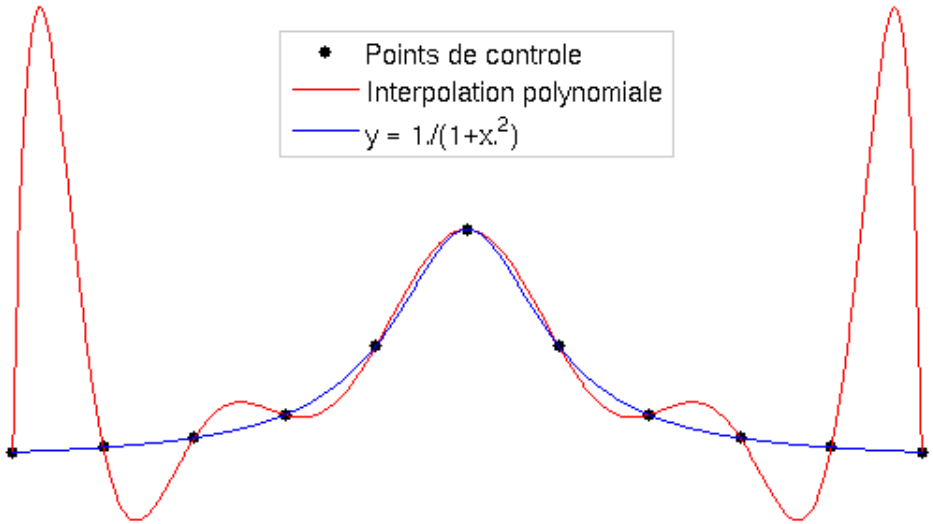
Pour $n+1$ points, on obtient donc le système suivant :

$$\begin{bmatrix} x_0^n & \cdots & x_0^3 & x_0^2 & x_0 & 1 \\ x_1^n & \cdots & x_1^3 & x_1^2 & x_1 & 1 \\ x_2^n & \cdots & x_2^3 & x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^n & \cdots & x_n^3 & x_n^2 & x_n & 1 \end{bmatrix} \begin{pmatrix} a_n \\ \vdots \\ \vdots \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}$$

(matrice carrée)
On montre que le déterminant de cette matrice n'est pas nul si les x_i sont distincts (déterminant de Vandermonde).

Interpolation polynomiale

Remarques :



En général, les méthodes polynômiales ne sont pas très stables entre les points de contrôle (phénomène de Runge).

Interpolation polynomiale

Stabilité :

- on diminue le degré du polynôme $\rightarrow$ problème surdéterminé, moindres carrés

  le polynôme ne passera pas nécessairement par tous les points de contrôle.
- on découpe l'intervalle d'interpolation en petits intervalles $\rightarrow$ pleins de polynômes de petit degré.

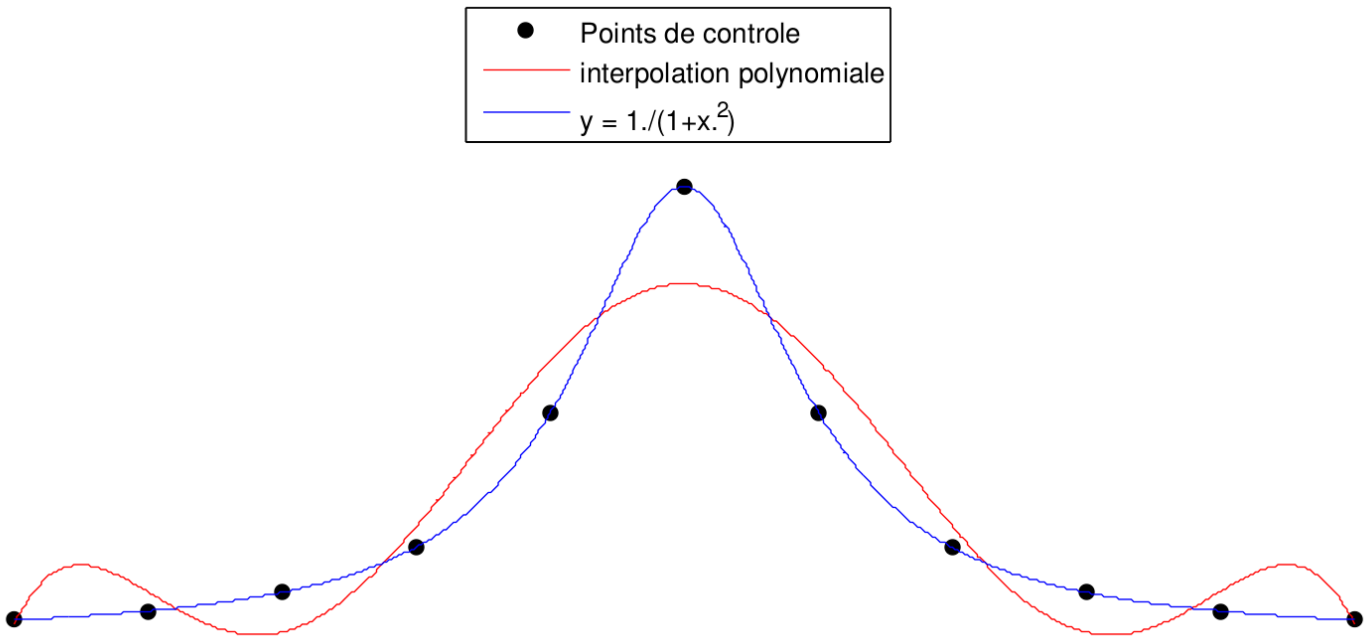
Interpolation polynomiale

Moindres carrés :

On choisit délibérément le degré m du polynôme inférieur au degré n prévu.
$\rightarrow$ système surdéterminé (moindres carrés).

$$\begin{bmatrix} x_0^m & x_0^{m-1} & x_0^{m-2} & \cdots & x_0 & 1 \\ x_1^m & x_1^{m-1} & x_1^{m-2} & \cdots & x_1 & 1 \\ x_2^m & x_2^{m-1} & x_2^{m-2} & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n-1}^m & x_{n-1}^{m-1} & x_{n-1}^{m-2} & \cdots & x_{n-1} & 1 \\ x_n^m & x_n^{m-1} & x_n^{m-2} & \cdots & x_n & 1 \end{bmatrix} \begin{pmatrix} a_m \\ a_{m-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}$$

Interpolation polynomiale



→ 11 points de contrôle

→ polynôme de degré 7

Spline cubique

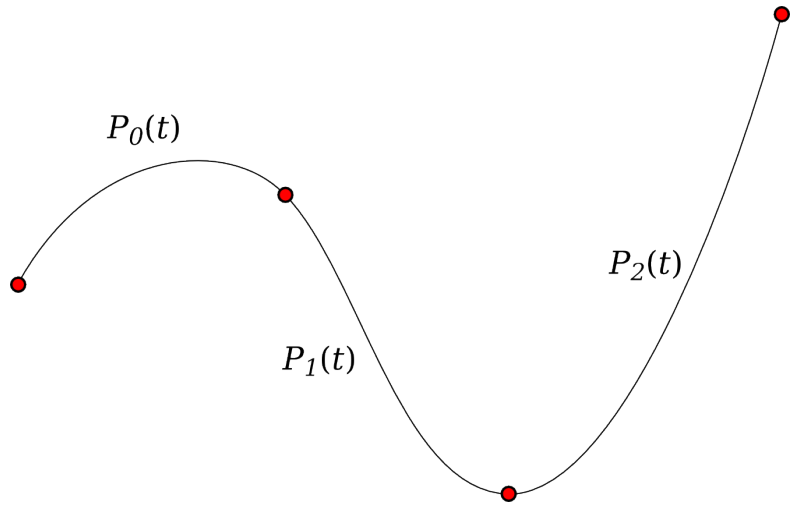
Introduction :

Pour éviter le phénomène de Runge, on peut choisir de découper l'intervalle d'interpolation en intervalles plus petits sur lesquels on utilise un polynôme de petit degré.

$\rightarrow$ Les splines cubiques sont un ensemble de polynômes de degré 3 passant par un ensemble de $n+1$ points de contrôle.
$\rightarrow$ On considère des points régulièrement espacés en abscisse : (i, x_i) avec $0 \leqslant i \leqslant n$.

Spline cubique 1D

Pour une spline 1D passant par $n+1$ points de contrôle $(x_0, x_1, ..., x_n)$, la i-ème partie de la spline est représenté par le polynôme $P_i(t)$:



$$P_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \text{ avec } t \in [0,1] \text{ et } i = 0, ..., n-1 .$$

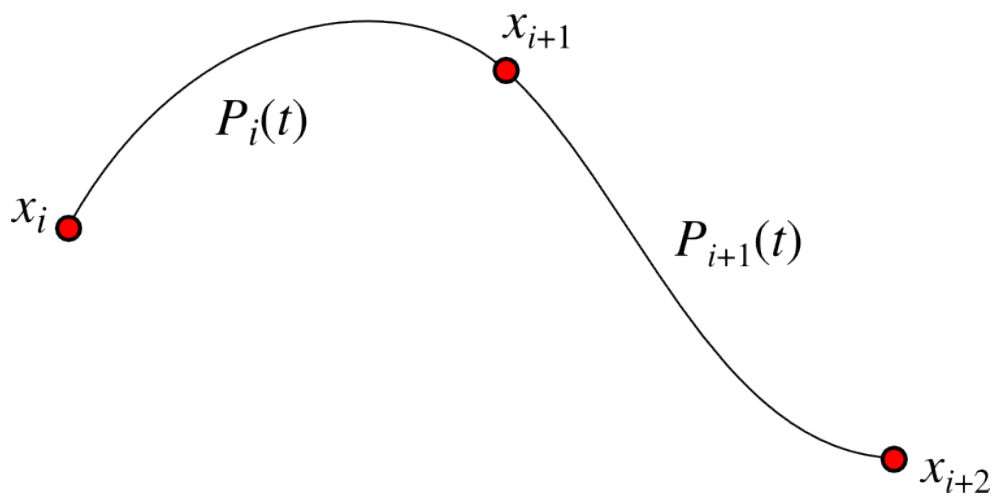
On constate qu'il y a priori trop de degrés de liberté $(4n)$ alors que le degré 1 suffit.

Spline cubique 1D

Contraintes

Chaque bout de spline doit être cohérente avec la suivante :

- $\rightarrow$ $P_i(1) = P_{i+1}(0) = x_{i+1}$ (la fin de $P_i(t)$ = début de $P_{i+1}(t)$)
- $\rightarrow$ $P_i'(1) = P_{i+1}'(0)$ (idem pour la dérivée (continuité C^1))
- $\rightarrow$ $P_i''(1) = P_{i+1}''(0)$ (et pour la dérivée seconde (continuité C^2))

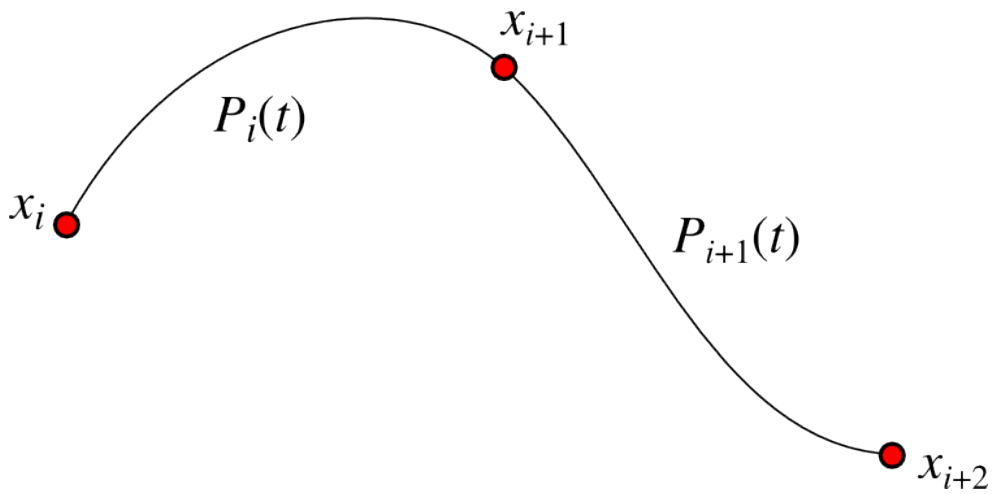


Spline cubique 1D

$$P_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$

Points de contrôle :

$$P_i(0) = x_i = a_i$$
$$P_i(1) = x_{i+1} = a_i + b_i + c_i + d_i$$

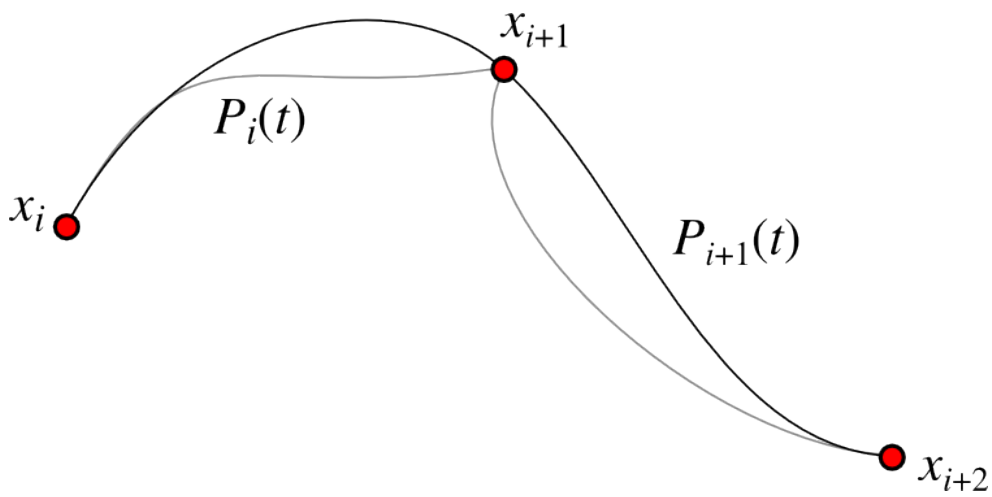


Cubic Spline 1D

$$P_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$

Dérivée première :

$$P_i'(0) \equiv D_i = b_i$$
$$P_i'(1) \equiv D_{i+1} = b_i + 2c_i + 3d_i$$



Cubic Spline 1D

$$P_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$

Dérivée seconde :

$$P_i''(0) \equiv S_i = 2c_i$$
$$P_i''(1) \equiv S_{i+1} = 2c_i + 6d_i$$

donc $P_i''(1) = P_{i+1}''(0)$ s'écrit

$$c_i + 3d_i = c_{i+1}$$

Cubic Spline 1D

Résumé :

- point de contrôle $\rightarrow$ $a_i = x_i$
- dérivée première $\rightarrow$ $b_i = D_i$
- point de contrôle $\rightarrow$ $x_{i+1} = a_i + b_i + c_i + d_i$
- dérivée première $\rightarrow$ $D_{i+1} = b_i + 2c_i + 3d_i$

En combinant ces 4 équations, on obtient pour $i = 0, \ldots, n-1$:

$$(\star) \begin{cases} a_i = x_i \\ b_i = D_i \\ c_i = 3(x_{i+1} - x_i) - 2D_i - D_{i+1} \\ d_i = 2(x_i - x_{i+1}) + D_i + D_{i+1} \end{cases}$$

auquel s'ajoute la continuité C^2.

Spline cubique 1D

On peut réécrire chaque polynôme en fonction des D_i et des x_i

$$\begin{cases} a_i = x_i \\ b_i = D_i \\ c_i = 3(x_{i+1} - x_i) - 2D_i - D_{i+1} \\ d_i = 2(x_i - x_{i+1}) + D_i + D_{i+1} \end{cases}$$

$$\begin{aligned} P_i(t) &= a_i + b_i t + c_i t^2 + d_i t^3 \\ &= x_i + D_i t + \big(3(x_{i+1} - x_i) - 2D_i - D_{i+1}\big) t^2 \\ &\quad + \big(2(x_i - x_{i+1}) + D_i + D_{i+1}\big) t^3 \end{aligned}$$

satisfait $(\star)$ par construction.
Noter que les $D_0, D_1, \ldots, D_n$ sont librement choisis à ce stade.

Spline cubique 1D

Contraintes C^2 :

$$P_i(t) = x_i + D_i t + \big(3(x_{i+1} - x_i) - 2D_i - D_{i+1}\big) t^2 + \big(2(x_i - x_{i+1}) + D_i + D_{i+1}\big) t^3$$

Pour $0 \leqslant i \leqslant n-2$, $c_{i+1} = c_i + 3d_i$ devient

$$\begin{aligned} &3(x_{i+2} - x_{i+1}) - 2D_{i+1} - D_{i+2} \\ &\quad = 3(x_{i+1} - x_i) - 2D_i - D_{i+1} + 6(x_i - x_{i+1}) + 3D_i + 3D_{i+1} \end{aligned}$$

soit, pour $i = 0, \ldots, n-2$

$$D_i - 4D_{i+1} + D_{i+2} = 3(x_{i+2} - x_i)$$

Spline cubique 1D

Contraintes C^2 (expression matricielle) :
Pour $i = 0, \ldots, n-2$, $D_i - 4D_{i+1} + D_{i+2} = 3(x_{i+2} - x_i)$

$$\begin{bmatrix} 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \end{bmatrix} \begin{pmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{n-1} \\ D_n \end{pmatrix} = \begin{pmatrix} 3(x_2 - x_0) \\ 3(x_3 - x_1) \\ \vdots \\ 3(x_{n-1} - x_{n-3}) \\ 3(x_n - x_{n-2}) \end{pmatrix}$$

Noter que la matrice est de taille $(n-1) \times (n+1)$

Spline cubique 1D

Conditions au bord

Trois types de conditions possibles :

- spline serrée = D_0 et D_n sont fixées par l'utilisateur
  (+2 équations)
- périodique (spline circulaire ou fermée) : $x_0 = x_n$; alors
  $D_0 = D_n$ et $P_0''(0) = P_{n-1}''(1)$
  (+2 équations)
- spline naturelle : $P_0''(0) = P_{n-1}''(1) = 0$
  (+2 équations)

Spline cubique 1D

Spline naturelle

On rajoute $P_0''(0) = P_{n-1}''(1) = 0$, c-à-d $2D_0 + D_1 = 3(x_1 - x_0)$ et $D_{n-1} + 2D_n = 3(x_n - x_{n-1})$

$$\begin{bmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 2 \end{bmatrix} = \begin{pmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{n-1} \\ D_n \end{pmatrix} = \begin{pmatrix} 3(x_1 - x_0) \\ 3(x_2 - x_0) \\ 3(x_3 - x_1) \\ \vdots \\ 3(x_{n-1} - x_{n-3}) \\ 3(x_n - x_{n-2}) \\ 3(x_n - x_{n-1}) \end{pmatrix}$$

Spline cubique 1D

Spline fermée

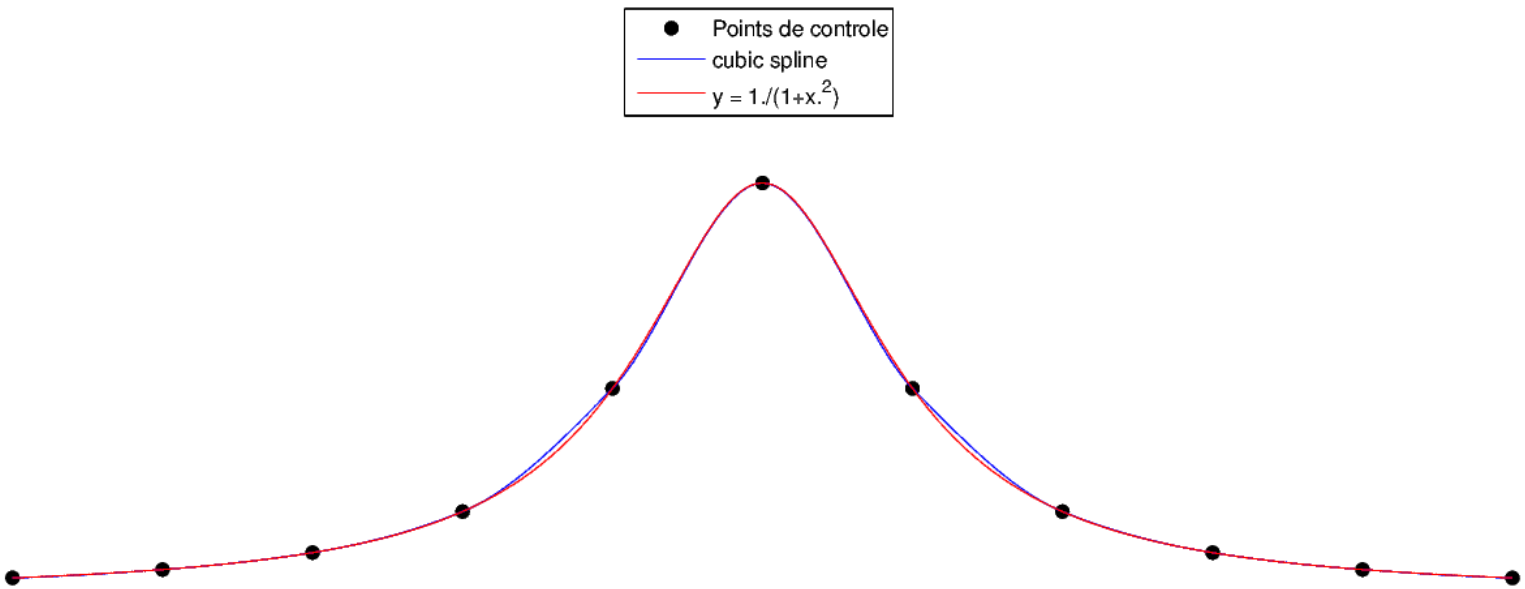
On rajoute $D_0 = D_n$ et $P_0''(0) = P_{n-1}''(1)$, c-à-d

$$\begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 4 & 1 & \\ 1 & & & & 1 & 4 & \end{bmatrix} \begin{pmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ D_{n-2} \\ D_{n-1} \end{pmatrix} = \begin{pmatrix} 3(x_1 - x_{n-1}) \\ 3(x_2 - x_0) \\ 3(x_3 - x_1) \\ \vdots \\ 3(x_{n-1} - x_{n-3}) \\ 3(x_n - x_{n-2}) \end{pmatrix}$$

On a simplifié le système en enlevant D_n.

Spline cubique 1D

En 1D :



Spline cubique paramétriques (2D et 3D)

En paramétrique :

En 2D, on veut autoriser des courbes qui ne sont pas des graphes au dessus de l'axe des abscisses.
En 3D, la spline passe par un ensemble de $n+1$ points de contrôle $\left\{(x_i, y_i, z_i)\right\}_{i\in 0\ldots n}$.

Chaque élément de spline est défini par 3 ensembles de polynômes $P_i^x(t)$, $P_i^y(t)$, $P_i^z(t)$ calculés en résolvant trois systèmes linéaires.

Spline cubique 3D

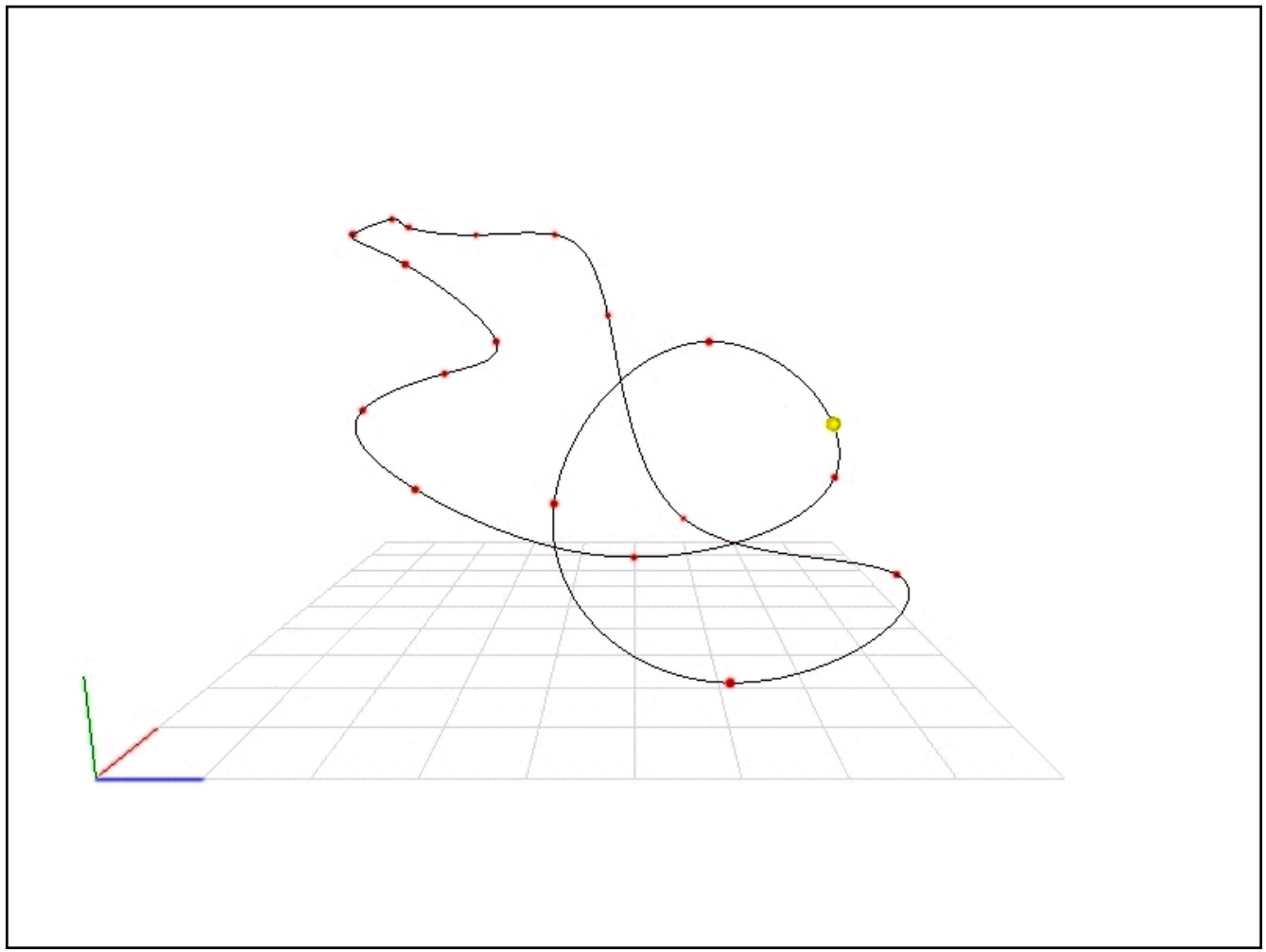
$$\begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ 1 & & & & & 1 & 4 \end{bmatrix} \begin{pmatrix} Dx_0 \\ Dx_1 \\ Dx_2 \\ Dx_3 \\ \vdots \\ Dx_{n-1} \\ Dx_n \end{pmatrix} = \begin{pmatrix} 3(x_1 - x_n) \\ 3(x_2 - x_0) \\ 3(x_3 - x_1) \\ \vdots \\ 3(x_{n-1} - x_{n-3}) \\ 3(x_n - x_{n-2}) \\ 3(x_0 - x_{n-1}) \end{pmatrix}$$

$$\begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ 1 & & & & & 1 & 4 \end{bmatrix} \begin{pmatrix} Dy_0 \\ Dy_1 \\ Dy_2 \\ Dy_3 \\ \vdots \\ Dy_{n-1} \\ Dy_n \end{pmatrix} = \begin{pmatrix} 3(y_1 - y_n) \\ 3(y_2 - y_0) \\ 3(y_3 - y_1) \\ \vdots \\ 3(y_{n-1} - y_{n-3}) \\ 3(y_n - y_{n-2}) \\ 3(y_0 - y_{n-1}) \end{pmatrix}$$

$$\begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ 1 & & & & & 1 & 4 \end{bmatrix} \begin{pmatrix} Dz_0 \\ Dz_1 \\ Dz_2 \\ Dz_3 \\ \vdots \\ Dz_{n-1} \\ Dz_n \end{pmatrix} = \begin{pmatrix} 3(z_1 - z_n) \\ 3(z_2 - z_0) \\ 3(z_3 - z_1) \\ \vdots \\ 3(z_{n-1} - z_{n-3}) \\ 3(z_n - z_{n-2}) \\ 3(z_0 - z_{n-1}) \end{pmatrix}$$

Spline cubique 3D

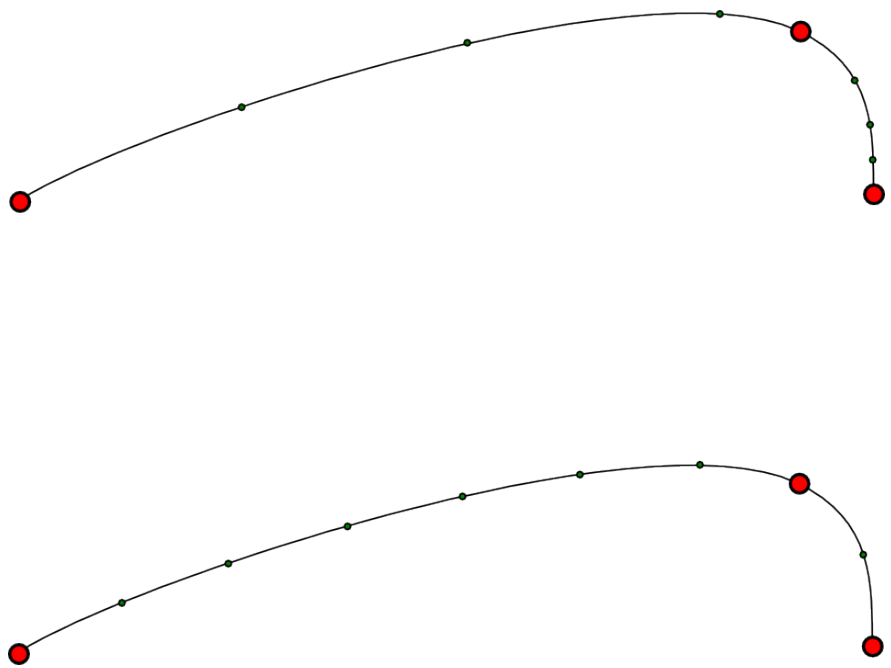
En 3D :



Spline cubique 3D

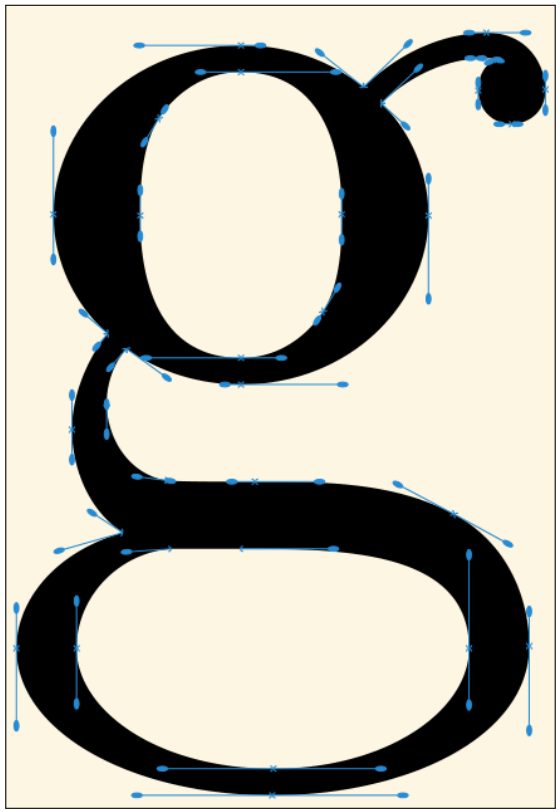
Inconvénient :

Il faut noter que t varie de 0 à 1 sur une portion de spline de façon non uniforme, ce qui complique l'utilisation de l'interpolation.



Courbes de Bézier

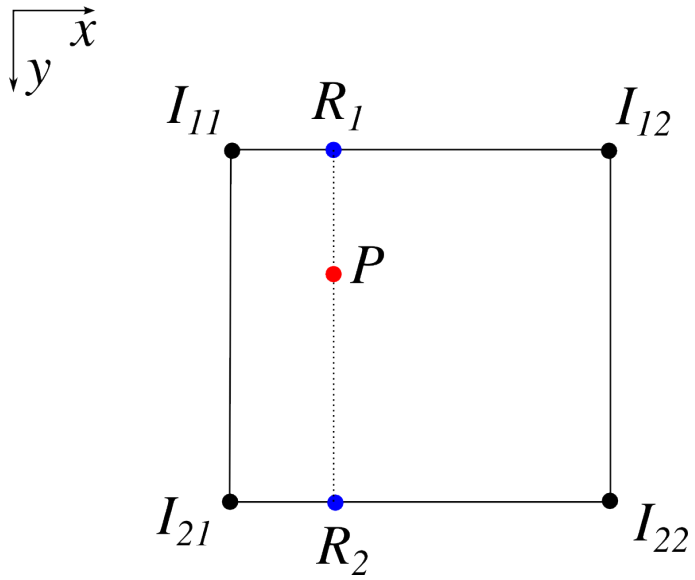
En informatique graphique, les courbes de Bézier sont utilisées pour définir des objets courbés et lisses: par exemple, les polices PostScript, TrueType.



Lettre g, police Computer Modern, par Sebastian, stackexchange.com

Interpolation bilinéaire

Pour interpoler en dimension 2 sur un carré (au lieu d'un intervalle)



souvent utilisée pour faire de l'antialiasing Interpolation linéaire pour R_1 et R_2 :

$$t = \frac{x_2 - x_p}{x_2 - x_1} \quad \rightarrow \quad \begin{array}{l} f(R_1) = tf(I_{11}) + (1-t)f(I_{12}) \\ f(R_2) = tf(I_{21}) + (1-t)f(I_{22}) \end{array}$$

Interpolation bilinéaire

Soit :

$$t = \frac{x_2 - x_p}{x_2 - x_1} \quad \text{et} \quad s = \frac{y_2 - y_p}{y_2 - y_1}$$

$$f(P) = (1-t)(1-s)f(I_{11}) + t(1-s)f(I_{12}) + s(1-t)f(I_{21}) + stf(I_{22})$$

Interpolation bilinéaire

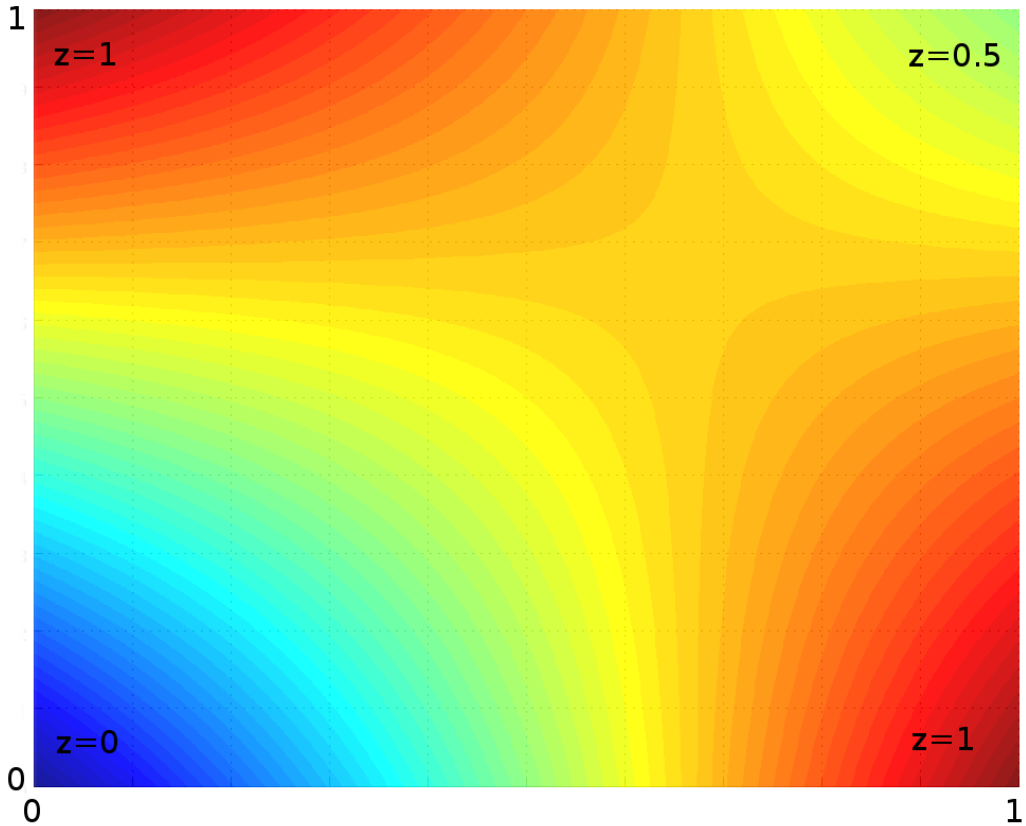
coordonnées unitaires :

$$f(x,y) = f(0,0)(1-x)(1-y)+f(1,0)x(1-y)+f(0,1)(1-x)y+f(1,1)xy$$

forme matricielle :

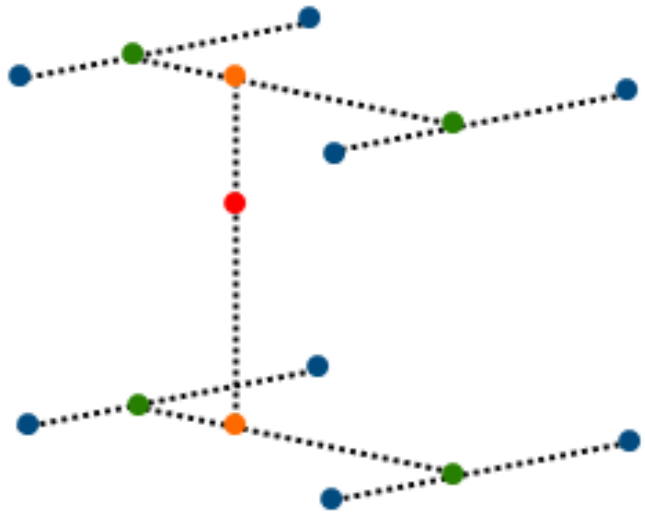
$$f(x,y) = \begin{pmatrix} 1-x & x \end{pmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{pmatrix} 1-y \\ y \end{pmatrix}$$

Interpolation bilinéaire



Interpolation trilinéaire

Extension naturelle en 3D :



Interpolation nD

Comment fait-on en dimension n?

$\rightarrow$ Radial Basis Function Interpolation

Radial Basis Function Interpolation

Introduction :

Cette méthode permet d'interpoler une fonction :

$$u = f(x_1, x_2, \ldots, x_n)$$

Applications :

- interpoler une couleur dans une image 2D
- interpoler une température dans un espace 3D
- interpoler une vitesse dans un espace 3D + t
- ...

Radial Basis Function Interpolation

Méthode :

entrée :

- un ensemble de k points nD $\left\{\mathbf{x}_i = (x_1^i, x_2^i, \ldots, x_n^i)\right\}_{i=1\ldots k}$
- leur valeur $u_i = f(\mathbf{x}_i)$

sortie : On cherche une fonction $g(\mathbf{x})$ telle que $g(\mathbf{x}_i) = u_i$, $\forall i$

Radial Basis Function Interpolation

On définit $g(\mathbf{x})$ telle que :

$$g(\mathbf{x}) = \sum_{i=1}^{k} \omega_i . \phi(|\mathbf{x} - \mathbf{x}_i|)$$

avec :

- ω_i des coefficients à déterminer.
- $\phi(|\mathbf{a} - \mathbf{b}|)$ une fonction "radiale" dont la valeur dépend de la distance entre $\mathbf{a}$ et $\mathbf{b}$.

Radial Basis Function Interpolation

Fonction ϕ :

Soit $d = \|\mathbf{a}-\mathbf{b}\|$ la norme L_2 du vecteur $\mathbf{a}-\mathbf{b}$, $\phi(d)$ est une fonction "radiale" dont la valeur dépend uniquement de d (et éventuellement de quelques constantes).

$\rightarrow$ Il existe une multitude de fonctions radiales telles que :

- $\phi(d) = d$
- une *multiquadrique* $\phi(d) = \sqrt{1 + (\varepsilon d)^2}$

ou bien (radiales décroissantes)

- une *inverse quadric* $\phi(d) = \frac{1}{1+(\varepsilon d)^2}$
- une gaussienne $\phi(d) = e^{-\varepsilon d^2}$

Radial Basis Function Interpolation

$$g(\mathbf{x}) = \sum_{i=1}^{k} \omega_i . \phi(|\mathbf{x} - \mathbf{x}_i|)$$

- $\phi(|\mathbf{x} - \mathbf{x_i}|)$: on peut le calculer $\forall \mathbf{x_i} \rightarrow$ OK.
- ω_i : on les connait pas, mais on sait que $g(\mathbf{x}_i) = u_i$, $\forall i$.

On obtient k contraintes sur les ω_i :

$$\sum_{i=1}^{k} \omega_i . \phi(|\mathbf{x}_m - \mathbf{x}_i|) = u_m \quad \forall m \in 1...k$$

Radial Basis Function Interpolation

$$\sum_{i=1}^{k} \omega_i . \phi(|\mathbf{x}_m - \mathbf{x}_i|) = u_m \quad \forall m \in 1...k$$

On peut l'écrire matriciellement sous la forme :

$$\begin{bmatrix} \phi(|\mathbf{x}_1 - \mathbf{x}_1|) & \phi(|\mathbf{x}_1 - \mathbf{x}_2|) & \cdots & \phi(|\mathbf{x}_1 - \mathbf{x}_k|) \\ \phi(|\mathbf{x}_2 - \mathbf{x}_1|) & \phi(|\mathbf{x}_2 - \mathbf{x}_2|) & \cdots & \phi(|\mathbf{x}_2 - \mathbf{x}_k|) \\ \vdots & \vdots & \vdots & \vdots \\ \phi(|\mathbf{x}_k - \mathbf{x}_1|) & \phi(|\mathbf{x}_k - \mathbf{x}_2|) & \cdots & \phi(|\mathbf{x}_k - \mathbf{x}_k|) \end{bmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_k \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{pmatrix}$$

$\rightarrow$ on peut résoudre et trouver les ω_i.

Radial Basis Function Interpolation

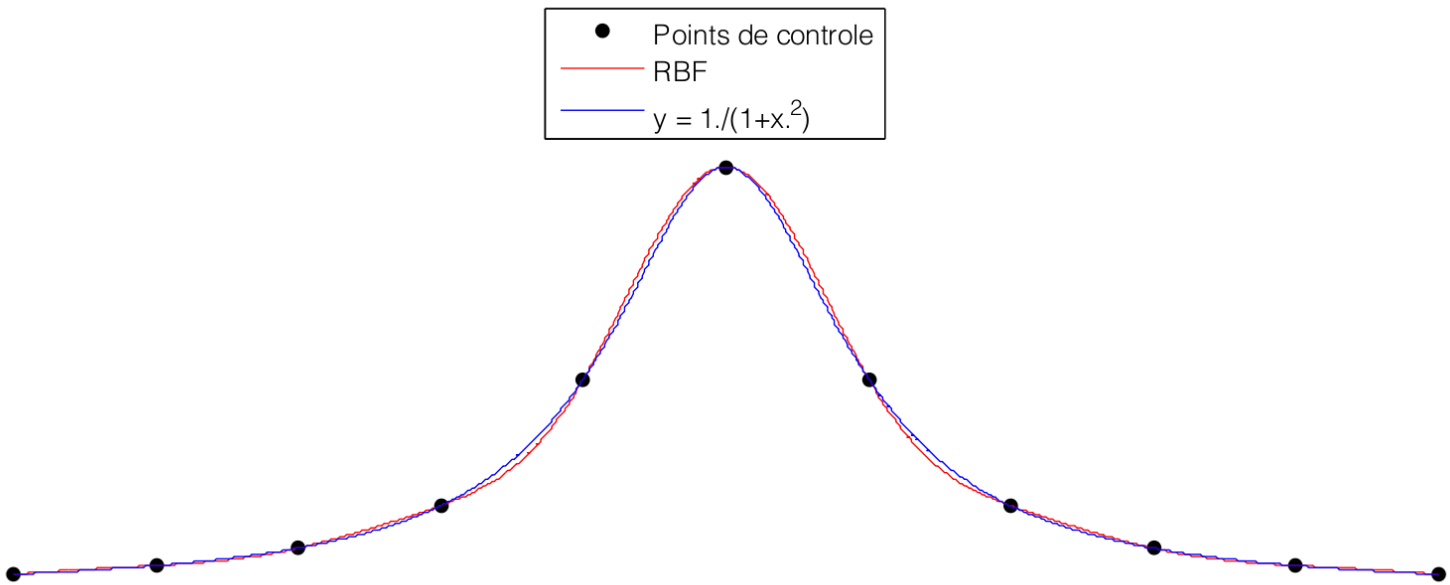
Évaluation d'un point :

Pour un point **y** quelconque que l'on souhaite interpoler, on obtient :

$$u_{\mathbf{y}} = \sum_{i=1}^{k} \omega_i . \phi(|\mathbf{y} - \mathbf{x}_i|)$$

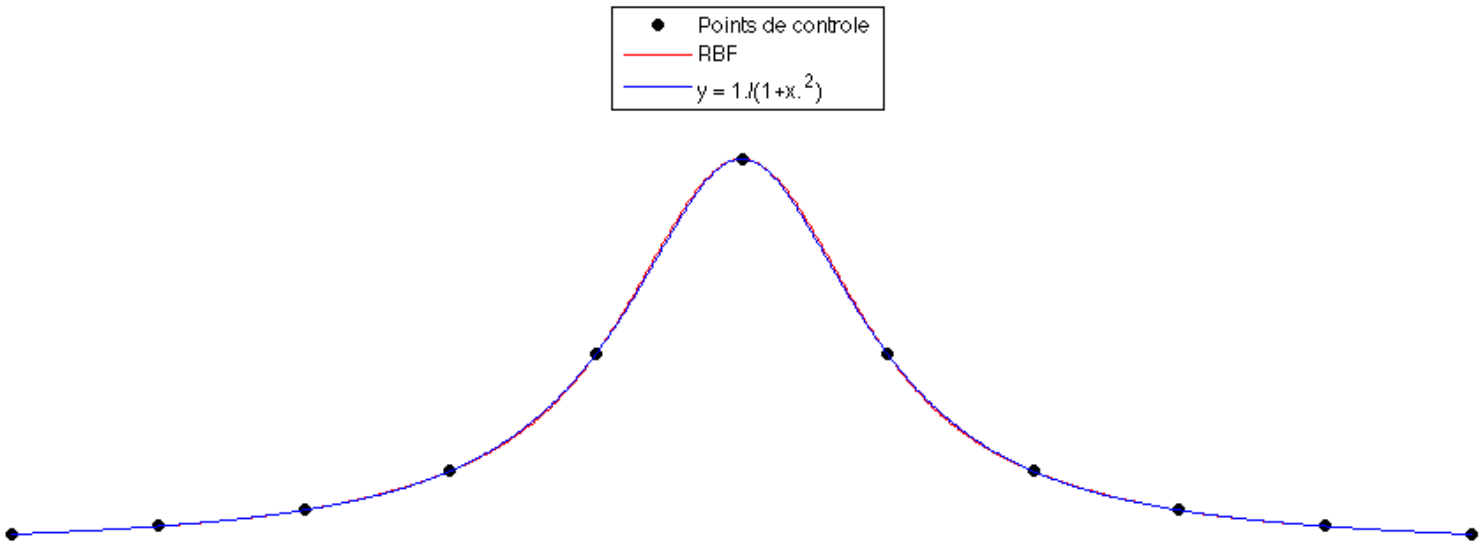
Radial Basis Function Interpolation

En 1D : $\phi(d) = e^{-d^2}$



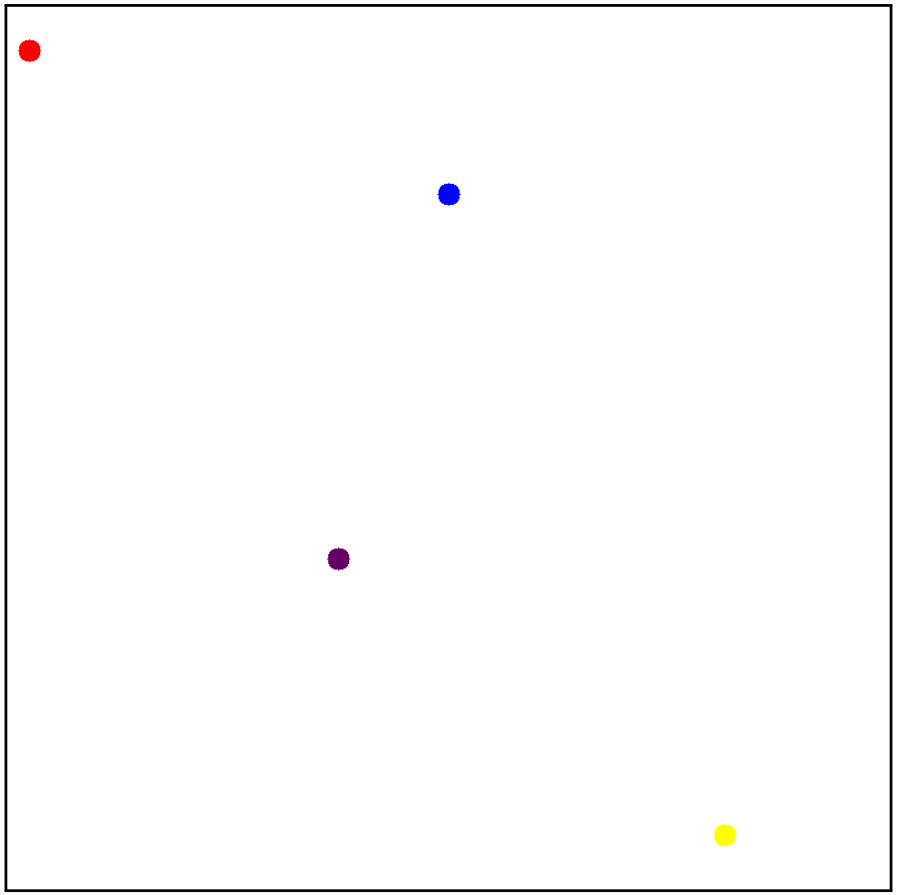
Radial Basis Function Interpolation

En 1D : $\phi(d) = \dfrac{1}{1+d^2}$



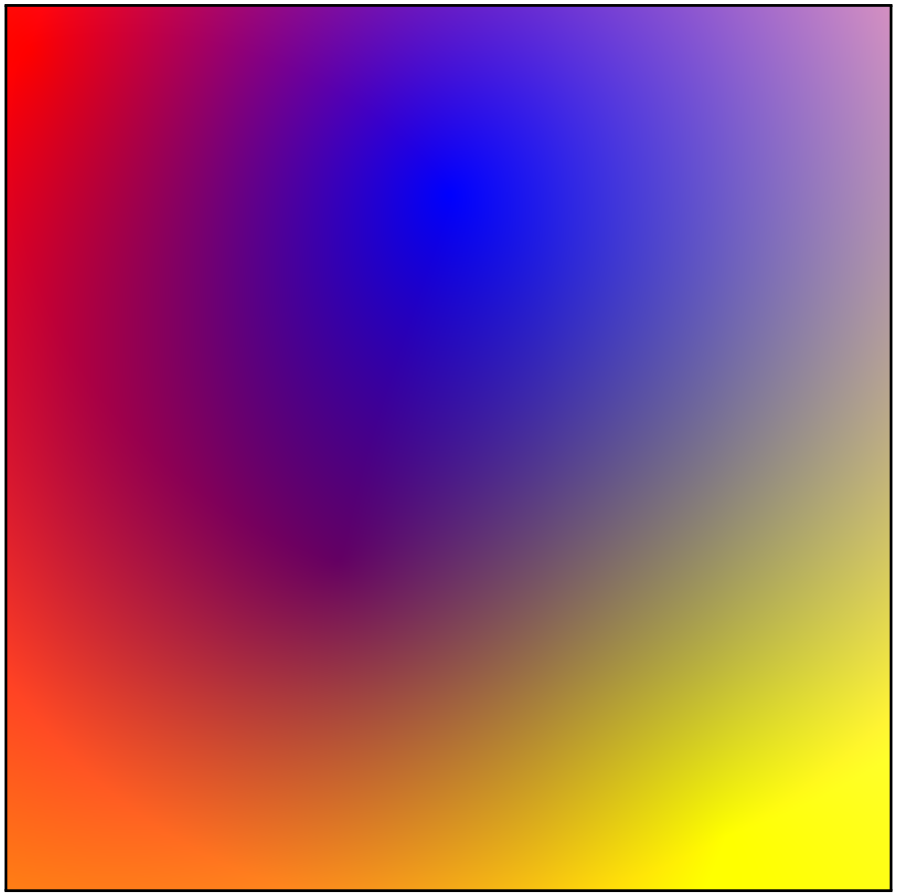
Radial Basis Function Interpolation

Exemple : interpolation 2D de couleur



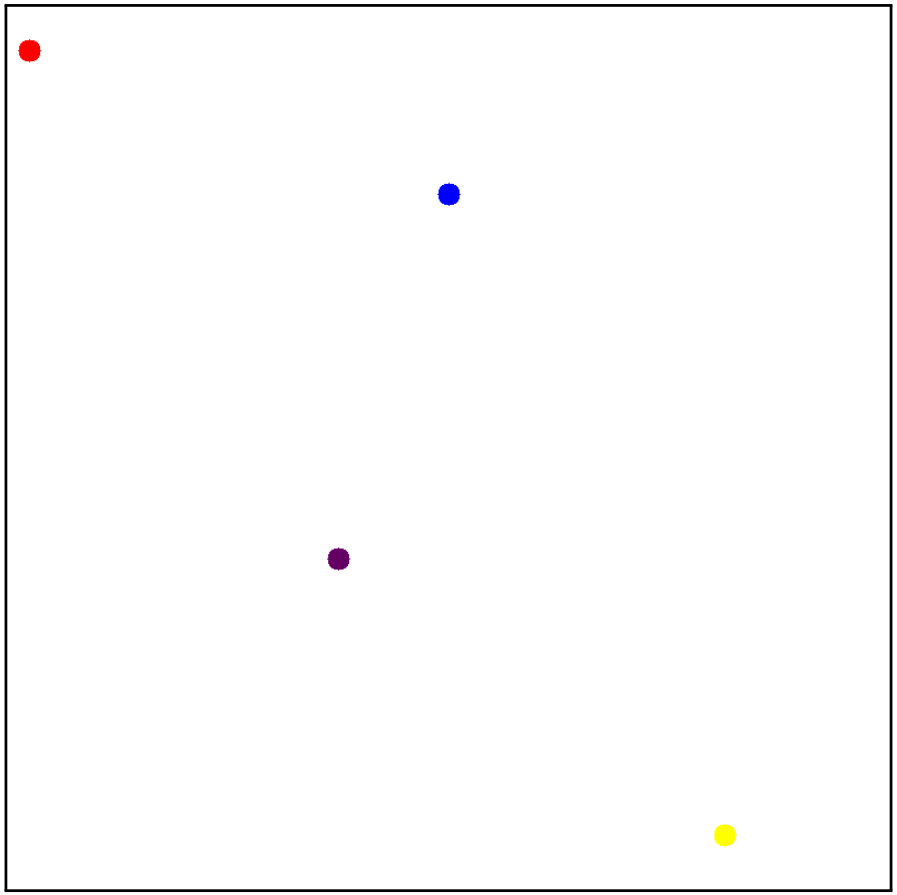
Radial Basis Function Interpolation

Exemple : interpolation 2D de couleur : $\phi(d) = d$



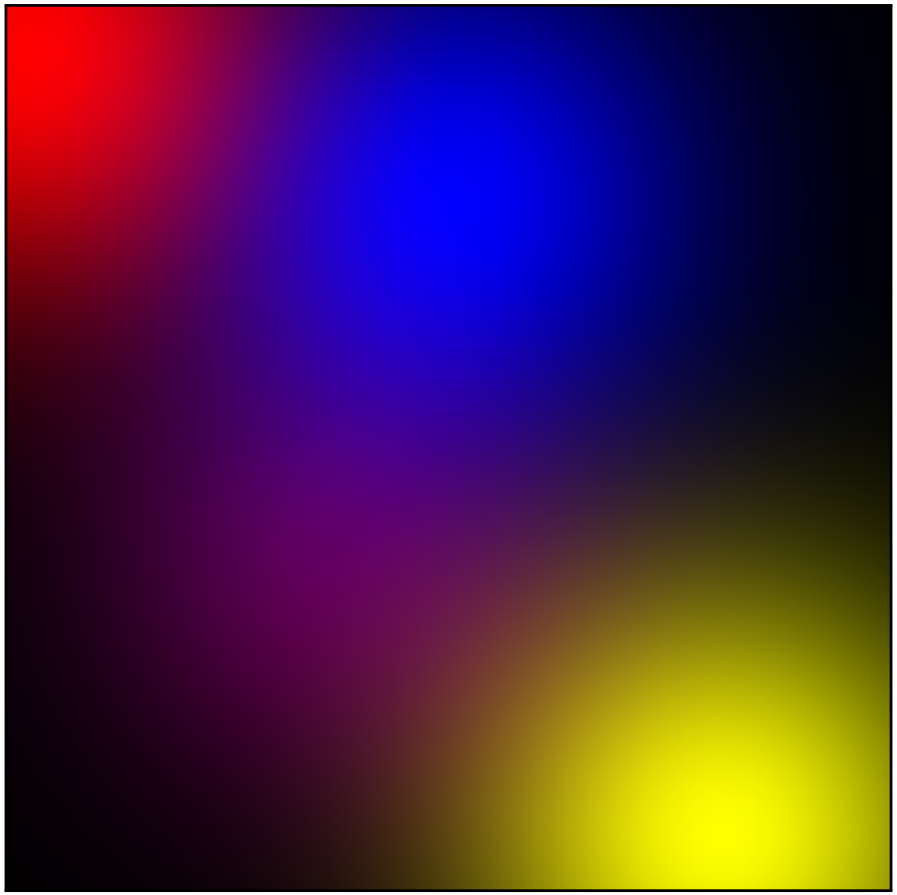
Radial Basis Function Interpolation

Exemple : interpolation 2D de couleur



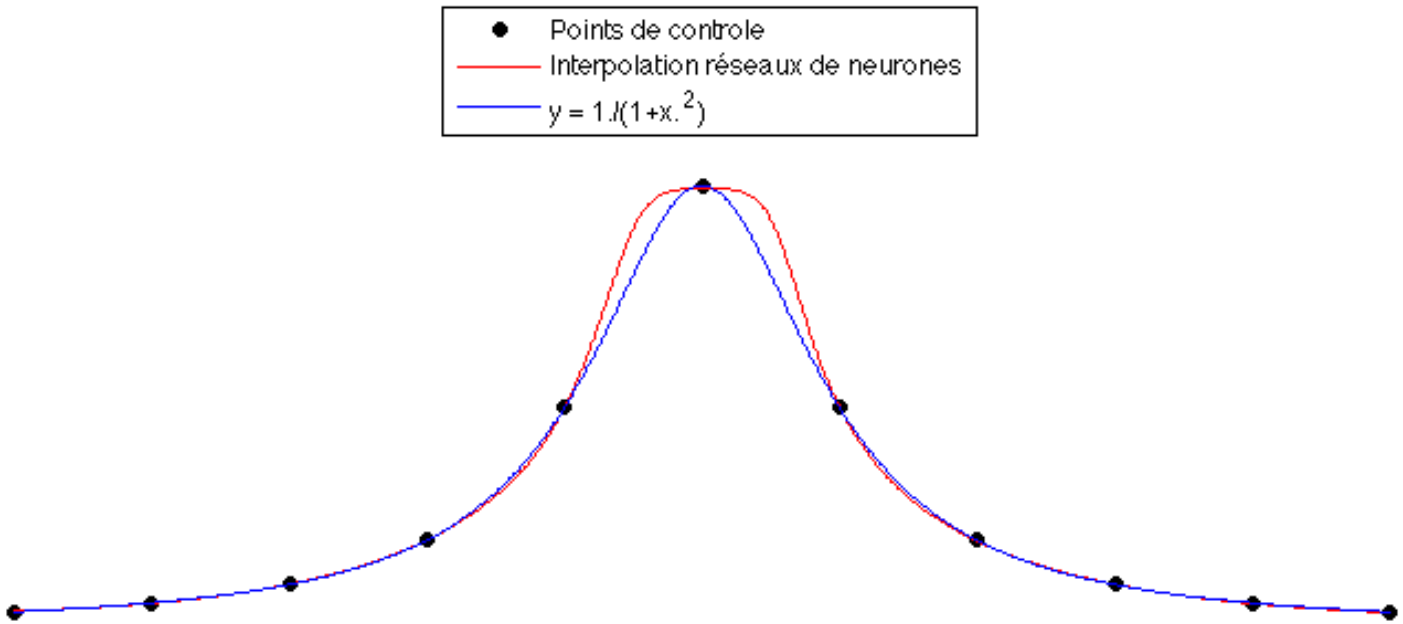
Radial Basis Function Interpolation

Exemple : interpolation 2D de couleur : $\phi(d) = e^{-\varepsilon d^2}$



Divers

Réseaux de neurones :



→ Topologie : 1-3-3-1

→ Apprentissage : 10 000 itérations