Open Wifi Localizator

Rémy Detobel, Denis Hoornaert, Nathan Liccardo, Robin Petit
Université libre de Bruxelles, Département des sciences informatiques, Bruxelles

Résumé—blablabla...

Index Terms—Computer Society, IEEEtran, journal, LaTeX, paper, template.

TABLE DES MATIÈRES

1 INTRODUCTION

2 MODÉLISATION

2.1 Graphe

Un graphe (non-dirigé) est un couple $\Gamma = (V, E)$ où V est un ensemble de nœuds et $E \subset V \times V$ est un ensemble de couples de nœuds. Un graphe non-dirigé peut être pondéré (chaque arête a un poids donné) et est alors déterminé par un triplet $\Gamma_w = (V, E, \omega)$, pour $\omega \colon E \to \mathbb{R}$, une fonction de poids. Dans notre cas, cette fonction est assimilable à la métrique euclidienne car $V = \mathbb{R}^2$:

$$\omega \ : E \to \mathbb{R}^+ : (v_1, v_2) \mapsto \|v_1 - v_2\|_E \, . \tag{1}$$

Les graphes ont été largement étudiés, et des algorithmes efficaces existent. En particulier, la recherche de plus court chemin.

2.2 Recherche du plus court chemin

Selon les critères du graphe, il existe plusieurs algorithmes de recherche de plus court chemin. Les plus connus sont Dijkstra, son amélioration en complexité A*, Bellman-Ford, Floyd-Warshall.

Bellman-Ford a été déterminé pour gérer un graphe ayant des arêtes de poids positives, et Floyd-Warshall a pour objectif de déterminer *la valeur* du plus court chemin entre chaque paire de nœuds.

2.2.1 Dijkstra

L'algorithme de Dijkstra est bien connu et a une complexité très intéressante, à savoir $O(|E| + |V| \log |V|)$ et détermine un chemin minimisant le coût (somme des poids des arêtes). [1]

Le concept de cet algorithme est détaillé en annexe A.

*2.2.2 A**

L'algorithme A* est un descendant de l'algorithme de Dijkstra dans le sens où le concept dérive directement de Dijkstra à une différence près : l'algorithme A* utilise une *heuristique*, c-à-d une fonction $h : V \to \mathbb{R}$ donnant une estimation du coût entre chaque nœud et le nœud de destination, afin d'alléger la complexité de l'algorithme dans le pire des cas : l'algorithme A* est en $O(|E|)$. L'utilisation de cette heuristique permet également d'adopter plusieurs fonctions de coût différente, et donc permet d'adapter la notion de *plus court chemin* en fonction de ce qui est recherché.

L'utilisation de l'heuristique se fait comme suit : au moment de choisir le nœud à traiter dans une étape i, contrairement à l'algorithme de Dijkstra qui sélectionne le nœud ayant cumulé le plus petit coût dans les étapes $j < i$, l'algorithme A* sélectionne le nœud v minimisant la quantité $g(v)$ définie par :

$$g(v) = \text{coût}(v) + h(v), \tag{2}$$

avec h la fonction heuristique.

1. Dans le cas où plusieurs chemins de même poids sont possibles, Dijkstra fournit toujours un des chemins de manière déterministe selon l'implémentation.

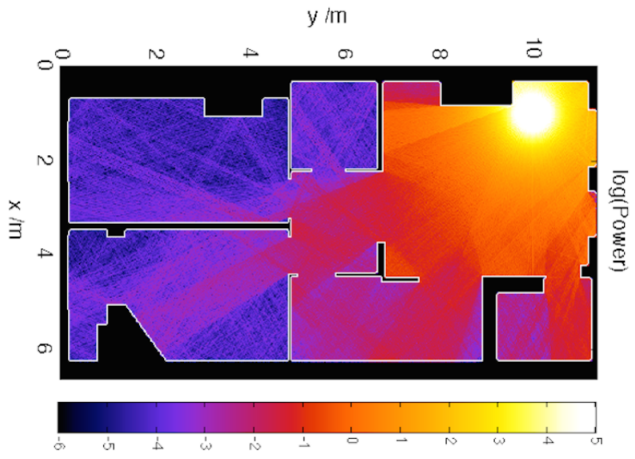
3 Localisation

3.1 Introduction aux Wifi

3.2 Problème rencontrer par l'utilisation des Wifi

Ce qui distingue la localisation via *Wifi* des autres types de localisation est le caractère imprévisible de la propagation des ondes. En effet, bien que la propagation dans un espace ouvert suit une fonction donnée, la propagation des ondes au sein d'un bâtiment est plus complexe à cause de la réflexion de l'atténuation des ondes survenant au contact des murs. De plus, l'environnement n'est pas la seule source d'altération du signal. En effet, un signal peut subir une détérioration de sa qualité de l'ordre de -3.5dBm si une personne se trouve entre l'émetteur et le récepteur [5]. Le signal peut aussi être détérioré par des interférences dont les sources peuvent être multiple. La difficulté repose dans ces deux derniers points car ceux-ci surviennent le plus souvent de manière inopinée.

Pour avoir un moyen efficace de se localiser sur base des *Wifi*, il faut pouvoir résoudre ces différents problèmes, ou trouver un moyen de les rendre moins significatifs.

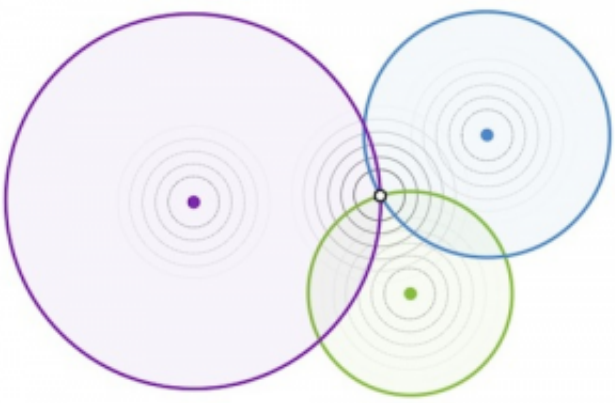


3.3 Méthodes

On peut diviser les méthodes de localisation en deux catégories bien distinctes.

Nous avons dans un premier temps, les méthodes dites de « propagation » qui se basent sur la connaissance préalable de l'ensemble des points d'accès *Wifi* qui seront utilisés ainsi que sur la qualité du signal que reçoit l'utilisateur pour pouvoir déterminer la position exacte de ce dernier.

Les méthodes de la deuxième catégorie parviennent, quand à elle, à localiser l'utilisateur en calculant la similarité entre une mesure (ensemble de *Wifi*) donnée et une mesure présente en base de données et qui a été faite ultérieurement.
Ces méthodes seront discutées et expliquées dans les deux sous-sections suivante.



3.3.1 *Trilatération*

La trilatération est une méthode mathématique permettant de trouver une coordonnée *consensus* qui sera contenu dans l'intersection de trois cercles. À la différence de la triangulation, la trilatération est basée sur l'utilisation de trois distances alors que la triangulation utilise trois angles donnés. Le schéma typique d'une trilatération est représenté par la figure 3.3.1.

Plus formellement, pour déterminer la position d'un utilisateur sur un plan cartésien, nous devons disposer des coordonnées cartésiennes du centre de chaque cercle ainsi que de la distance entre l'utilisateur et ces centres comme mentionné plus haut. L'obtention des coordonnées cartésiennes de l'utilisateur est réalisée via les formules suivante (voir annexe+NUMÉRO pour le développement détaillé) :

$$x = \frac{CE - BF}{EA - BD} \tag{3}$$

$$y = \frac{FA - DC}{EA - BD}, \tag{4}$$

où :
- $A = 2x_2 - 2x_1$
- $B = 2y_2 - 2y_1$
- $C = r_1^2 - r_2^2 - y_1^2 + y_2^2 - x_1^2 + x_2^2$
- $D = 2x_3 - 2x_2$
- $E = 2y_3 - 2y_2$
- $F = r_2^2 - r_3^2 - y_2^2 + y_3^2 - x_2^2 + x_3^2$

L'adaptation de la trilatération au problème de localisation se fait en déterminant la distance séparant l'utilisateur d'un point d'accès *Wifi* (centre de cercles). On obtient la distance entre un récepteur (utilisateur) et un émetteur via une fonction qui prend en paramètre la qualité de signal de l'émetteur perçu par le récepteur. La fonction est la suivante :

$$\alpha_f(s) = \frac{27,55 - 20\log_{10}(f) + |s|}{20} \tag{5}$$

$$d_f(s) = 10^{\alpha_f(s)}, \tag{6}$$

où :
- f est la fréquence (généralement 2.4 Ghz ou 5.0 GHz) ;
- s est la qualité du signal (mesuré en dBm).

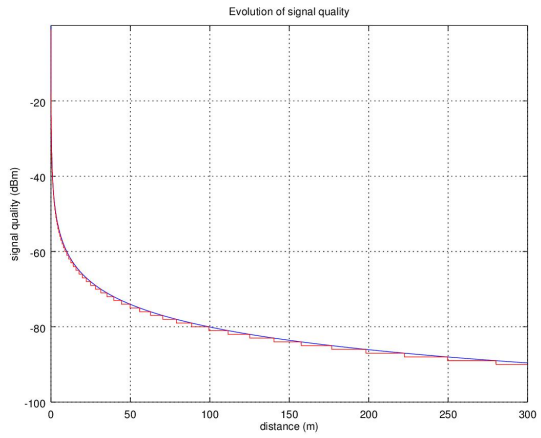


FIGURE 1. Répartition de la qualité de signal en fonction de la distance.

L'utilisation d'une telle fonction donne de bons résultats dans un environnement ouvert ou les obstacles sont rares. Comme démontré par [2], la précision de la localisation est situé entre 2 et 2.5 mètres. Néanmoins, dans le cas contraire, la fonction d'évaluation de la distance ne prenant pas en compte les obstacles, les estimations de distance s'en retrouvent biaisées et les coordonnées supposées de l'utilisateur aussi. De manière à résoudre ce problème, il est possible de remplacer la constante dans la formule (5) par une formule prenant en compte les éléments composant l'environnement. Cette formule est définie comme suit :

PLACER QQCH ICI (7)

Cette modification de la trilatération implique de disposer de plus de connaissance sur l'environnement. Collecter ces données demande du temps et même en consentant à les collecter, les estimations de distance laissent à désirer.

3.3.2 *Localisation via une Signal Strength Map*

Une *SSM* est défini comme étant une cartographie des régions couvertes par les signaux *Wifi*. La construction de cette carte se fait en regardant et enregistrant quels sont les points d'accès *Wifi* que l'on capte à une certaine position. On va donc enregistrer, dans une base de données, pour chaque position désirée, les identifiants (*BSS*) et les qualités de signal des points d'accès *Wifi* couvrant cette position.

3.3.2.1 Méthode utilisant les processus Gaussiens:

3.3.2.2 Méthode utilisant les maximum de vraisemblance: Comme mentionné plus haut, on va faire un ensemble de mesures pour chaque point du graphe et ainsi construire la base de données sur laquelle se basera notre méthode de localisation. Pour la méthode présente, on notera une mesure comme suit :

$$S^{(i)} = \{b_1, b_2, ..., b_n\} \qquad \forall j \in [1, k], \tag{8}$$

où :

— n est le nombre de points d'accès détectés lors de la mesure ;

— b_i, le i^{eme} point d'accès détecté, est défini comme un couple (a_i, S_i) décrivant respectivement l'identifiant d'un point d'accès et la qualité de signal perçue.

On prend plusieurs mesures pour un même point car il existe des variations de qualité de signal au cours du temps. En faisant de la sorte, on se donne une idée de la qualité d'un signal à un point du graphe. On notera ces « mesures moyennes » comme suit :

$$\bar{S} = \{(a_1, \bar{S}_1), ..., (a_n, \bar{S}_n)\}, \tag{9}$$

où :

— $\bar{S}_i = \frac{1}{k} \sum_{j=1}^{k} S_i^i \qquad \forall i \in [1, n]$

Cependant, considérer seulement la moyenne comme utile statistique n'est pas suffisant à cause du bruit et des interférences pouvant survenir (voir section précédente). Pour palier ce problème, on calcule aussi la variance (notée ν) pour chaque point du graphe. Ainsi, on obtiendra une nouvelle forme appelée ensemble caractéristique qui est définit comme suit :

$$S^* = \{(a_1, \bar{S}_1, \nu_1), ..., (a_n, \bar{S}_n, \nu_n)\} \tag{10}$$

Où :

— $\nu_i = \frac{1}{k-1} \sum_{j=1}^{k} (S_i^{(j)} - \bar{S}_i) \qquad \forall i \in [1, n].$

On peut donc définir notre base de données (notée $\mathcal{D}$) comme l'ensemble des points du graphe (noté L) pour lesquels il existe un ensemble caractéristique. Soit :

$$\mathcal{D} = \{L_1, L_2, ..., L_3\}, \tag{11}$$

où :

— $\dim(L_i)$, le nombre de points d'accès à cette position, est noté n_i

— S_i^* est l'ensemble caractéristique de L_i

Une fois la base de données remplie, on peut passer à l'étape dite d'exploitation. Il s'agit du processus qui sera utilisé par une personne pour pouvoir se localiser. Le processus va déduire la position de l'utilisateur sur base de la base de données préalablement créée (notée $\mathcal{D}$) et sur base d'une mesure qu'il aura effectuée (notée S'). La nature de cette mesure pouvant soit être de la forme (1) ou (2).

Le processus va déterminer une position $\hat{L}$ appartenant $\mathcal{D}$ pour laquelle la probabilité que l'utilisateur se trouve proche de ce point est la plus grande. Pour ce faire, on va calculer, pour chaque élément appartenant à $\mathcal{D}$, un score noté Z et considérer, comme solution à notre problème, le point correspondant au score Z minimal car le score Z minimal représente la vraisemblance maximale comme prouvé dans l'article (Mettre référence). Le score Z est définit comme suit :

$$Z_i = \sum_{j=1}^{n_i} \left(\frac{(\bar{S}_{ij} - S'_j)^2}{\nu_{ij}} \right), \tag{12}$$

où :

— $\bar{S}_{ij}$ est la moyenne de la qualité de signal du j^{eme} point d'accès ;
— ν_{ij} est la variance de la qualité de signal du j^{eme} point d'accès ;
— S'_j est la qualité de signal du j^{eme} point d'accès de la mesure effectué par l'utilisateur.

Mesures parasites

Dans cet article, on défini une mesure parasite comme la détection d'un point d'accès qui n'apparaitrait que très rarement lors de la prise de mesures. On qualifiera un point d'accès comme tel si le nombre de fois qu'il a été détecté est inférieur à un seuil fixé. Ce cas de figure survient pour des points d'accès situés à une distance telle qu'ils ne sont pas toujours perçus mais qui, dans le cas contraire, présentent des qualités de signal inférieur à $-90\mathrm{dBm}$.

3.3.2.3 Monte Carlo:

3.3.2.4 Amélioration par l'utilisation de filtres:

3.3.2.5 Amélioration par l'estimation de qualité de signal:

4 INTERPOLATION

4.1 Principe

Si $X = \{(x_i, y_i) \text{ t.q. } 0 \leqslant i \leqslant N\} \subset \mathbb{R}^2$ est un ensemble de points du plan euclidien, l'interpolation a pour objectif de déterminer une courbe passant par tous ces points. Il existe bien entendu une infinité de telles courbes. L'idée est donc de définir une forme générale et paramétrisée de la courbe, et de déterminer la valeur de ces paramètres en fonction de X.

4.2 Interpolation polynômiale

En supposant qu'il existe une fonction scalaire $F : \mathbb{R} \to \mathbb{R}$ telle que $\forall i \in \{0, \ldots, N\} : F(x_i) = y_i$, on peut chercher à déterminer une fonction interpolante sous forme polynômiale. Étant donné qu'un polynôme de degré $n \in \mathbb{N}$ est entièrement caractérisé par $n+1$ coefficients, on peut chercher à interpoler l'ensemble des couples (x_i, y_i) par une fonction polynômiale $P : \mathbb{R} \to \mathbb{R}$ de degré N. Les $N+1$ coefficients sont alors déterminés selon les $N+1$ équations $y_i = P(x_i)$. Pour ce faire, les méthodes des polynômes de Lagrange et de Newton peuvent être utilisées.

Cependant, un polynôme de degré élevé est soumis à de fortes oscillations entre les points de contrôle. De plus, pour $x \in \mathbb{R} \backslash [x_0, x_N]$, la valeur de $P(x)$ tend vite vers $+\infty$ en valeur absolue. C'est la raison pour laquelle une interpolation par morceaux est préférable si le nombre de points de contrôle est élevé, ou si de faibles variations sont attendues entre les points de contrôle.

4.3 Interpolation par morceaux

L'interpolation par morceaux (*piece-wise interpolation* en anglais) est un moyen d'interpoler les $N+1$ points de contrôle par une courbe étant déterminée par une union de courbes définies chacune sur un segment $[x_j, x_{j+1}]$.

4.3.1 Interpolation par splines polynômiales

Si $f : [x_0, x_N] \to \mathbb{R}$ est la fonction d'interpolation, et si on considère que $f(x) = f_i(x)$ pour $x \in [x_i, x_{i+1})$, alors on cherche les $f_i : [x_i, x_{i+1}] \to \mathbb{R}$ sous forme de polynôme. Il faut alors introduire un certain nombre de contraintes afin de pouvoir déterminer les coefficients de ces N fonctions.

Il est commun d'imposer que ces fonctions soient cubiques, et d'imposer une continuité en C^2, i.e. que les dérivées secondes des f_i soient continues aux points de contrôle.

4.3.2 Interpolation par une courbe paramétrique

La sous-section précédente ne concerne que le cas où tous les x_i sont distincts deux à deux, et donc où il existe une fonction F telle que pour tout i, on ait $y_i = f(x_i)$. Dans le cas où les points de contrôle représentent une trajectoire dans $\mathbb{R}^2$, cette supposition n'est plus correcte, et l'interpolation proposée ci-dessus ne peut fonctionner (par définition d'une fonction, $f : D \to I$, on a pour tout $x \in D$ que $f(x)$ donne une unique valeur $y \in I$).

On choisit alors de considérer la courbe d'interpolation comme une courbe paramétrique :

$$\gamma : [0, N] \to \mathbb{R}^2 : t \mapsto (x(t), y(t)). \tag{13}$$

De tels problèmes ne se posent plus, et l'interpolation peut alors être faite de la sorte.

4.3.3 Splines cubiques hermitiennes

Les splines cubiques hermitiennes sont des splines polynômiales (de degré 3) d'interpolation par une courbe paramétrique. L'idée est de dire que l'on cherche N fonctions $\gamma_i : [0,1] \to \mathbb{R}^2 : t \mapsto (x_i(t), y_i(t))$ où $x_i, y_i : [0,1] \to \mathbb{R}$ sont deux fonction polynômiales cubiques.

x_i et y_i sont deux fonctions cubiques, donc chacune déterminées par 4 coefficients, et on cherche N telles fonctions. Il faut alors 4N contraintes pour déterminer ces coefficients. On impose donc $\gamma_i(0) = (x_i, y_i)$ et $\gamma_i(1) = x_{i+1}, y_{i+1})$ pour $i = 0, \ldots, N-1$, ce qui donne $2(N-1)$ contraintes. Il en reste $2(N+1)$ à déterminer. Pour cela, seule une continuité C^1 est demandée, et on impose :

$$\forall i \in \{1, \ldots, N-1\} : \frac{d\gamma_i}{dt}(0) = \frac{1}{2}(x_{i+1} - x_{i-1}, y_{i+1} - y_{i-1}). \tag{14}$$

Ainsi, il ne reste plus que deux contraintes qui sont données par $\frac{d\gamma_0}{dt}(0)$ et $\frac{d\gamma_{N-1}}{dt}(1)$ qui sont choisis arbitrairement. Par exemple en étant mis à 0. Selon ces contraintes, les courbes γ_i sont définies de manière univoque (voir Annexe B).

Les splines cubiques hermitiennes présentent également l'avantage suivant : le déplacement d'un point de contrôle (x_j, y_j) n'a d'influence que sur les courbes γ_k pour $|k-j| \leqslant 1$. Ainsi, modifier un point de contrôle ne donne qu'une modification locale de la courbe, ce qui n'est pas le cas en général.

5 Discussion et limitations

6 Conclusion

Références

[1] Matteo Cypriani, Frédéric Lassabe, Philippe Canalda, François Spies, *Wi-Fi-Based Indoor Positioning : Basic Techniques, Hybrid Algorithms and Open Software Platform*. 2010.

[2] Bianca BOBESCU, Marian ALEXANDRU *Mobile indoor positioning using Wi-fi localisation*. Transilvania University, Brasov, Romania, 2015.

[3] OnkarPathak, Pratik Palaskar, Rajesh Palkar, Mayur Tawari, *Wi-Fi Indoor Positioning System based on RSSI Measurements from Wi-Fi Access Points A Trilateration Approach*. International Journal of Scientific & Engineering Research, 2014.

[4] Brian Ferris, Dirk Hähnel, Dieter Fox, *Gaussian Processes for Signal Strength-Based Location Estimation*. University of Washington, Department of Computer Science & Engineering, Seattle, WA Intel Research Seattle, Seattle, WA.

[5] Thomas Locher, Roger Wattenhofer, Aaron Zollinger, *Received-Signal-Strength-Based Logical Positioning Resilient to Signal Fluctuation*. Computer Engineering and networks Laboratory, ETH zurich, Switzerland.

Annexes

A. Algorithme de Dijkstra

L'idée de l'algorithme de Dijkstra est de conserver un vecteur des nœuds à traiter contenant initialement tous les nœuds, et duquel on enlève chaque nœud lorsqu'il est traité. Afin de choisir le nœud à traiter à chaque étape, il faut regarder lequel de ces nœuds nécessite un coût minimal pour être atteint depuis la source. C'est ainsi que l'algorithme de Dijkstra assure de déterminer un chemin optimal dans le graphe. L'algorithme peut être schématisé comme suit :

```
set all distances to infty
set src distance to 0
Q <- priority queue
Q.addAll(nodes)
WHILE NOT Q.empty()
  N <- Q.pop()
  IF N == dest
    RETURN the path from src to dest
  ELSE
    FOR EACH N' IN N.neighbours()
      update distance to N'
      update closest node leading to N'
    END FOR EACH
  END IF
END WHILE
```

B. Splines cubiques hermitiennes

Comme détaillé dans la sous-sous-section 4.3.3, en posant :

$$x_i(t) = \sum_{k=0}^{3} \alpha_{ik} t^k \tag{15}$$

$$y_i(t) = \sum_{k=0}^{3} \beta_{ik} t^k \tag{16}$$

$$p_i = (x_i, y_i) \tag{17}$$

pour $\mathbb{R}^2 \ni \eta_{ik} \coloneqq (\alpha_{ik}, \beta_{ik})$, on trouve de manière plus générale :

$$\gamma_i(t) = \sum_{k=0}^{3} \eta_{ik} t^k, \tag{18}$$

ce qui donne :

$$\gamma_i(0) = \eta_{i0} \quad \text{et} \quad \gamma_i(1) = \sum_{k=0}^{3} \eta_{ik}. \tag{19}$$

Pour $\mathbb{R}^2 \ni \Delta_i \coloneqq \frac{1}{2}(x_{i+1} - x_{i-1}, y_{i+1} - y_{i-1})$ $(i = 1, \ldots, N-1)$ et $\Delta_0 = 0 = \Delta_N$, on a alors pour $0 \leqslant i \lneqq N$:

$$\sum_{k=0}^{2} (k+1)\eta_{i(k+1)} t^k = \frac{d\gamma_i}{dt}(t), \tag{20}$$

et :

$$\eta_{i1} = \tfrac{d\gamma_i}{dt}(0) = \Delta_i \tag{21}$$

$$\sum_{k=0}^{2} (k+1)\eta_{i(k+1)} = \tfrac{d\gamma_i}{dt}(1) = \Delta_{i+1}. \tag{22}$$

Cela se réécrit sous forme de système matriciel qui donne :

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix}}_{\coloneqq M} \underbrace{\begin{bmatrix} \eta_{i0} \\ \eta_{i1} \\ \eta_{i2} \\ \eta_{i3} \end{bmatrix}}_{\coloneqq \eta} = \begin{bmatrix} p_i \\ p_{i+1} \\ \Delta_i \\ \Delta_{i+1} \end{bmatrix}. \tag{23}$$

En inversant la matrice M, on trouve la solution :

$$\eta = M^{-1} \begin{bmatrix} p_i \\ p_{i+1} \\ \Delta_i \\ \Delta_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_i \\ p_{i+1} \\ \Delta_i \\ \Delta_{i+1} \end{bmatrix} = \eta. \tag{24}$$

On remarque en effet qu'en changeant un des points de contrôle p_j, la modification de la courbe d'interpolation sera locale étant donné que les coefficients de la courbe γ_i ne dépendent que des points de contrôles p_{i-1}, p_i, et p_{i+1}.