# Estimation de tempo par fenêtrage, autocorrélation et corrélation avec des trains d’impulsions.

## Fenêtrage

Connaissant la force d’onset du signal (OSS), sortie de la fonction d’onset, on commence par découper le signal en fenêtres rectangulaire d’une durée d’environ 5,5s et se recouvrant avec un décalage d’environ 0.5s. L’objectif sera de proposer pour chaque fenêtre, une estimation du tempo. Dans l’absolu, cet algorithme peut donc tenir compte de légère variation autour du tempo global. Dans sa version de base, ce fenêtrage permettra d’obtenir plusieurs tempos candidats ainsi qu’un « score » pour chaque candidat.

## Autocorrélation généralisée

Le tempo étant pouvant être décrit comme la périodicité dans l’apparition des notes, on utilise l’autocorrélation du signal OSS fenêtré pour extraire cette périodicité. L’auteur de la publication sur laquelle est basé cet algorithme propose d’utiliser l’autocorrélation généralisé, une variante de l’autocorrélation décrit par :

Où *m* correspond à l’indice de la fenêtre. Dans le cas particulier ou c=2, Am est égale à l’autocorrélation du signal. Dans notre cas, on choisit, comme proposé, c=0.5.

## Mise en valeur des harmoniques et sélection des candidats

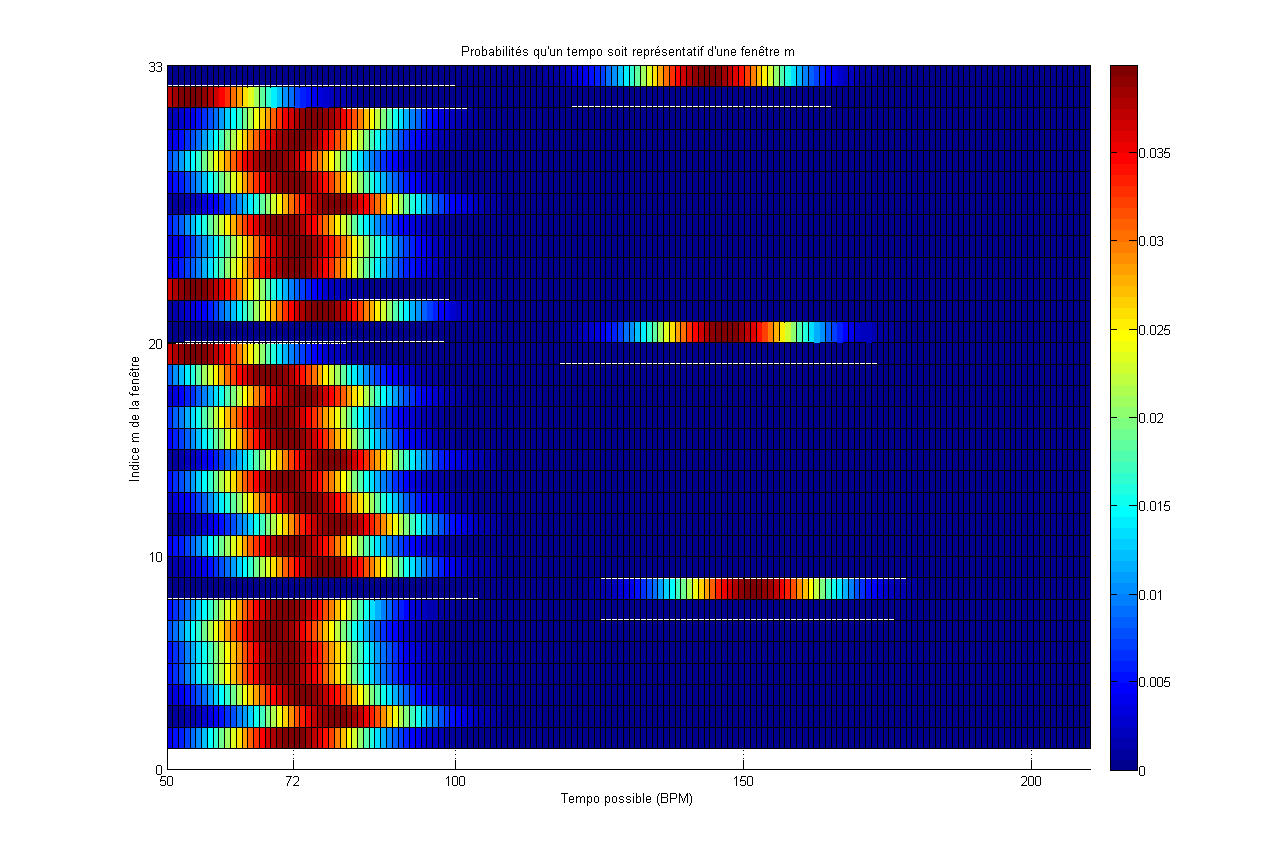
Dans un signal musical, les durées des doubles-croches, croches, noires, etc… sont liées par un rapport de 2 entre elles. On cherche donc à mettre en évidence ce lien en ajoutant au signal Am, son image sous-échantillonné dans un rapport de 2 et de 4, ce qui peut s’exprimé ainsi :

On extrait ensuite, pour chaque fenêtre *m* les 10 pics les plus importants dont le retard (l’abscisse) sera considéré comme tempo candidat P.

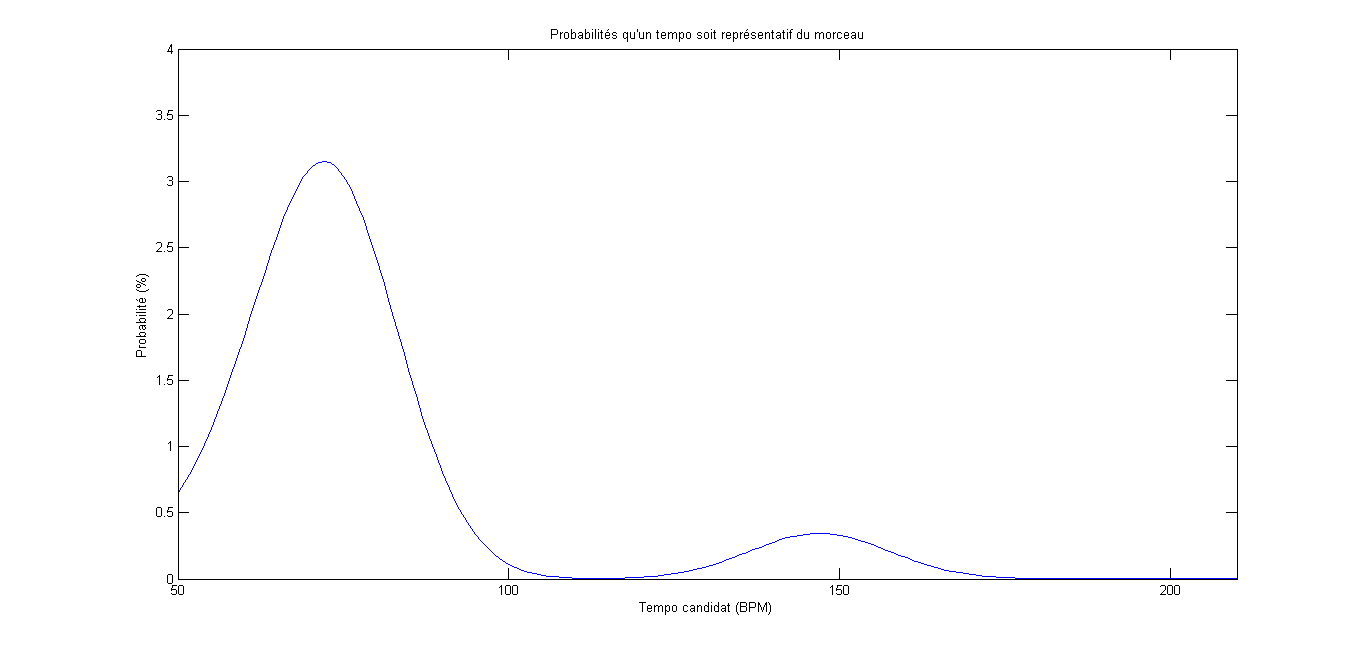
## Comparaison à des trains d’impulsions

On crée ensuite des trains d’impulsions pour chaque tempos candidat P. Un train d‘impulsion correspond à 4 impulsions unités dont la première ce produit à l’instant 0 et les suivantes sont espacées de 1 fois, 1,5 fois, ou 2 fois P donnant ainsi 3 trains d’impulsions que l’on somme entre eux dans le vecteur IP,m.

## Détermination du meilleur candidat.



Accumulation de toutes les gaussiennes dans un seul vecteur C.



## Méthode originale décrite dans la publication

Détection des 3 pics principaux (même s’il en existe moins…). Le pic ayant la plus forte amplitude correspond au tempo candidat L. Les deux autres pics sont les deuxième et troisième meilleur candidats, L2 et L3.

Utilisation d’une Support Vector Machin (SVM) : en entrée de la SVM, il est demandé plusieurs paramètres. Le tempo sera détecté avec une certaine incertitude σ. Dans la publication, cette incertitude varie avec le tempo candidat L : . Dans notre implémentation elle est fixe à 4 BPM. Les paramètres sont les suivants (tempo = tempo réel)

1. La probabilité que le tempo soit inférieur à L :
2. La probabilité que le tempo soit supérieur à L
3. La probabilité que le tempo soit proche de L
4. La probabilité que le tempo soir proche de L/2 (rapport de ½ )
5. La probabilité que le tempo soir proche de 2L (rapport de 2 )
6. La somme de ces 3 dernières probabilités
7. Les rapports entre L et le deuxième meilleur candidat
8. Le rapport entre L et le 3ème meilleur candidat
9. Le nombre de tempo pour lequel la densité de probabilité n’est pas nulle (en mode discret)
10. Le tempo candidat lui-même, L.

Ces paramètres sont ensuite normalisés entre 0 et 1 et on applique à ces 10 paramètres, les poids de la SVM déterminés au préalable par apprentissage. Si le produit scalaire entre le vecteur 10x1 des paramètres et le vecteur 10x1 des poids de la SVM est positif, il faut doubler le tempo candidat.

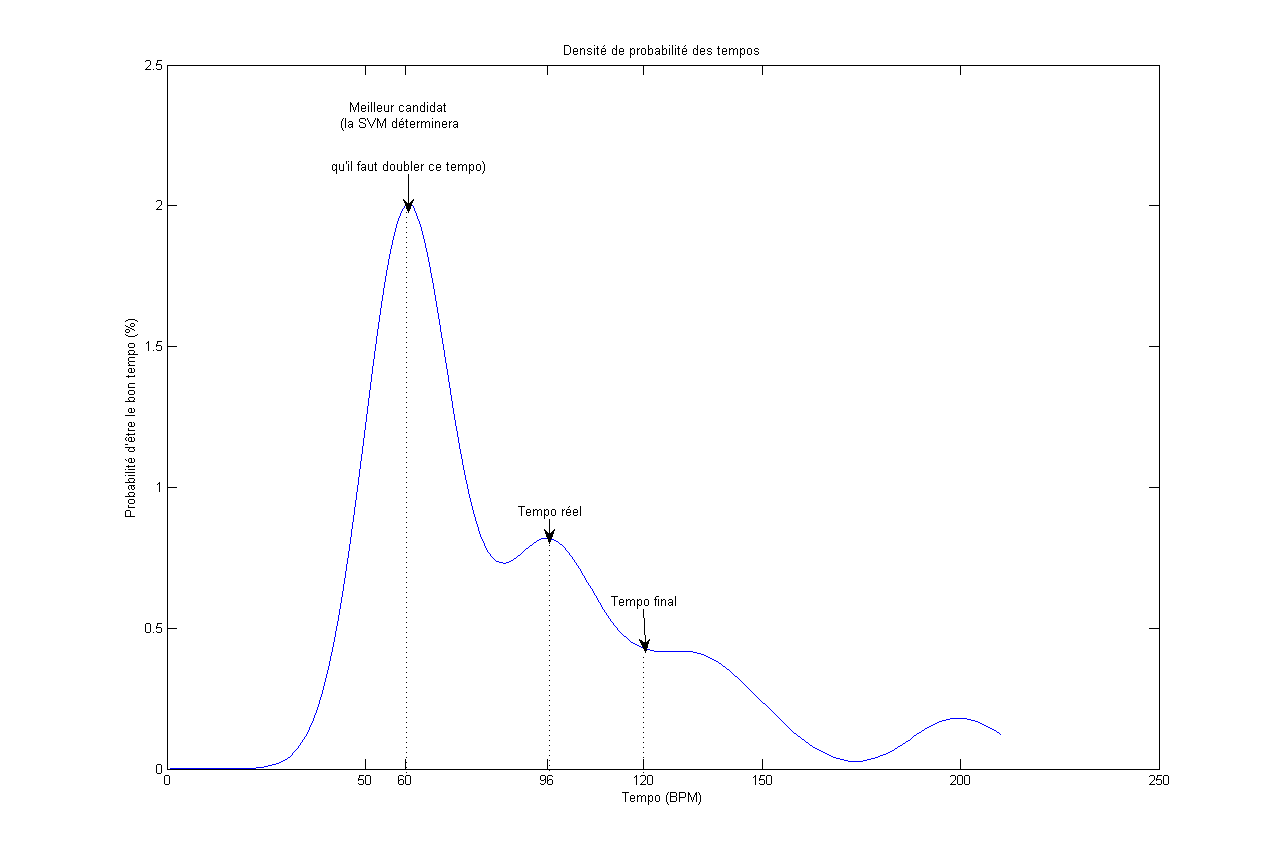
Remarque : Cette méthode bien qu’elle donne des résultats convenables, nous semble assez étrange dans ses détails : la publication ne cite que les paramètres 1, 4 et 10 alors que dans l’implémentation Matlab donnée en référence apparaissent ces 10 paramètres différents. Nous ne comprenons pas exactement la pertinence de certains paramètres comme les paramètres 1, 2 et 9.

D’autre part, nous avons pu récupérer les valeurs déterminées issues de l’apprentissage mais ces valeurs donnent de mauvais résultats sur notre jeu de donnée. Il a fallu les modifier manuellement et neutralisé certains paramètres pour obtenir des résultats corrects.

Enfin, la SVM de cette algorithme ne permet de déterminé que s’il faut doubler ou non le candidat final et non s’il faut le diviser par deux ou tout simplement choisir un autre candidat.

## Amélioration proposée

Ce qui motive notre volonté de modifier l’algorithme est notamment le fait qu’il ne peut choisir qu’entre 2 tempo possibles, le meilleur candidat et son double. Dans bien des cas, le tempo réel correspondra à la moitié du meilleur candidat, ou bien à un double/moitié d’un autre candidat :



La différence entre notre système et celui de la publication est que nous nous servons par la suite du tempo pour déterminer la durée musicale des notes (croche, noire, blanche…). Nous avons donc besoin d’un tempo fiable pour réaliser correctement cette étape. Une erreur sur le tempo à ce moment entraîne des répercussions par la suite. Dans notre exemple, une noire jouée à 90BPM sera déterminée comme une croche pointée à 120 BPM, ce qui est déjà en soi une erreur. Cette erreur aura également un impact au moment du découpage en mesure 4 :4 prévu par notre algorithme.

Or cette étape de détermination des durées est également un atout que nous pouvons utiliser. En effet, l’objectif de l’algorithme de détermination des durées est d’assigner à chaque note jouée une durée musicale ainsi qu’un coefficient de certitude que la durée à bien été déterminée. Ce coefficient correspond à la probabilité que la note jouée soit d’une certaine durée musicale, sachant le tempo estimé. Cela entraîne que si le tempo est mal estimé, cette probabilité attribuée à chaque note va être faible en moyenne, alors que si le tempo est correctement estimé, la moyenne des probabilités sera maximisée. Nous proposons donc de calculer ces probabilités pour tous les candidats (il peut y avoir 2 à 6 candidats environ).

Remarque : cette logique ne permet pas de déterminer s’il faut doubler ou diviser le tempo retenu. Il pourra toujours y avoir cette erreur d’un facteur 2 ou bien, on pourrait avoir 2 tempos candidat ex-aequo à la première place. On utilisera pour ce problème une méthode de machine learning basée sur la SVM de la publication.

On choisit donc de garder tous les maximums locaux présents dans la densité de probabilités C. Il y en a obligatoirement au moins 1 mais il peut y en avoir plusieurs, sans limite. On exécute alors la détermination des durées pour chaque tempo τ et on calcule à chaque fois la moyenne µ(τ) des probabilités p des N notes :

On considère alors que le meilleur candidat est le τ qui maximise µ(τ). On descend également la limite supérieur de tempo acceptable de 210BPM à 180BPM.