Faculté des Sciences Appliquées
Département d'Électricité, Électronique et Informatique

Recherche d'une évaluation robuste de l'excentricité dans des systèmes binaires d'étoiles

Mémoire présenté en vue de l'obtention
du grade de Maître en Sciences Informatiques
par Ludovic Delchambre

Année académique 2009-2010

Promoteur : Pierre Geurts

Membres du jury : Eric Gosset
Louis Wehenkel
Pierre Wolper

Résumé

Ce document a pour sujet d'étude l'évaluation de l'excentricité de l'orbite des étoiles binaires spectrales dont seule une des composantes est visible. Cette évaluation est effectuée au moyen de méthodes d'intelligence artificielle sur la base des observations effectuées. Cette problèmatique est abordée dans le cadre d'un problème de régression et d'évaluation au sens des moindres carrés de l'excentricité.

Il est axé autour de deux aspects de l'intelligence artificielle qui sont d'un côté, les algorithmes génétiques et d'un autre côté les algorithmes d'apprentissage. Ces derniers sont abordés au moyen des algorithmes des plus proches voisins, des arbres extrêmement randomisés et des réseaux de neurones. Ces trois algorithmes sont utilisés via diverses techniques d'extrapolation de points nécessaires à la normalisation des observations et permettent d'optimiser la phase d'apprentissage proprement dite.

Ce document se voulant accessible, il présente l'ensemble des bases théoriques nécessaires à une bonne compréhension du problème de l'évaluation de l'excentricité dans des systèmes binaires d'étoiles ainsi que des différents algorithmes d'apprentissage, algorithmes génétiques et techniques d'extrapolation de points.

Finalement, une présentation exhaustive de l'ensemble des résultats obtenus en ajustant au mieux les différents paramètres des techniques utilisées est proposée.

Abstract

This thesis deals with the approximation of orbit eccentricity in single-lined spectroscopic binary stars. This approximation is performed by means of artificial intelligence based on recorded observations. This problem is viewed as a regression problem and as an eccentricity least squares approximation.

It focuses on two aspects of artificial intelligence : genetics algorithms on one hand and machine learning algorithms on the other hand. These are addressed by means of nearest neighbour algorithms, extremely randomized trees and neural networks. These algorithms are used through a variety of extrapolation methods that are required to normalize observation data and to optimize the actual learning stage.

This thesis aims to be accessible. We therefore introduce the theoretical basics needed to understand the evaluation of binary star systems eccentricity, machine learning algorithms, genetics algorithms, and extrapolation methods.

As a conclusion, we describe the entire set of data resulting from the different methods, adjusted with the best-fitting parameters.

Table des matières

Table des figures

Liste des tableaux

Avant-propos

Ce mémoire a été réalisé en collaboration avec le groupe d'astrophysique des hautes énergies (GAPHE) du Département d'astrophysique, géophysique et océanographie (AGO) de l'université de Liège dans le cadre de la mission Gaia de l'Agence Spatiale Européenne (ESA).

La mission Gaia a pour objectif principal[1] de créer la carte tridimensionnelle la plus précise de notre galaxie à travers l'observation d'environ un milliard d'étoiles. Cette carte tridimensionnelle nous permettra de mieux comprendre la formation et l'évolution de notre galaxie. Cette mission est programmée pour une durée de cinq ans durant lesquels le satellite enregistrera la couleur, la luminosité et la position de tous les objets célestes tombant dans son champ de vision. En répétant ces observations tout au long de sa mission, il permettra aux astronomes de calculer la position, la vitesse et la direction de chacun des objets célestes observés, de suivre leurs changements de luminosité et de déterminer s'ils possèdent un compagnon proche. De plus, suite à ces observations, et sur base des parallaxes annuelles mesurées, il sera possible de déterminer précisément la distance entre ces objets. Parmi ses nombreux autres objectifs, nous pouvons citer entre autres : la détection de nouvelles exoplanètes, de supernovæ, la découverte de nouveaux objets célestes dans notre système solaire, la vérification de certaines constantes de la relativité générale, ...

Dans ce document, nous nous intéresserons plus particulièrement à la détermination des paramètres orbitaux dans les systèmes binaires d'étoiles sur base des mesures effectuées par le détecteur RVS (Radial Velocity Spectrometer) du satellite Gaia. Une simulation de ces mesures sera employée afin d'obtenir une première évaluation optimale de l'excentricité au moyen de techniques issues de l'intelligence artificielle.

Nous organiserons ce document autour de trois grands axes. Premièrement une partie d'introduction à la problématique permettra au lecteur de cerner au mieux les principes astrophysiques liés aux étoiles binaires (chapitre 1), ceux-ci étant nécessaires à la comprèhension de l'objectif de ce travail (chapitre 2).

Deuxièmement, la partie théorique s'attachera à présenter les techniques d'ajustement de courbes (chapitre 3) qui seront utilisées de concert avec les algorithmes d'apprentissage (chapitre 4). Les algorithmes génétiques seront également abordés dans le cadre de cette partie théorique (chapitre 5).

Finalement, une partie centrée sur les développements pratiques cloturera ce document. Cette dernière mettra principalement en avant l'exploitation des données d'entrée afin de permettre leur utilisation par les algorithmes génétiques (chapitre 6) ainsi que l'optimisation des algorithmes génétiques (chapitre 7). Nous présenterons enfin les différents résultats obtenus (chapitre 8).

Première partie

Introduction à la problématique

Chapitre 1

Les étoiles binaires

Une étoile binaire est un système stellaire composé de deux étoiles orbitant autour de leur centre de masse commun. On estime actuellement que la moitié des étoiles de l'univers sont des systèmes binaires[2]. Dans un tel système, l'étoile la plus lumineuse est appelée la primaire et la moins lumineuse, la secondaire. L'orbite de chacune de ces étoiles est régie par les Lois de Kepler[3] :

1. Toutes les planètes décrivent un mouvement elliptique autour du soleil qui occupe un des foyers de l'ellipse
2. Une droite joignant une planète au soleil balaie des aires égales en des temps égaux
3. Le carré de la période de révolution d'une planète autour du soleil est proportionnel au cube de la distance moyenne qui la sépare de celui-ci

Précisons que cette énumération n'est pas tout à fait rigoureuse : elle reflète le fait que le soleil est tellement massif comparé aux planètes du système solaire que sa position se confond quasiment avec celle du centre de masse du système. Toutefois, tout comme dans le cadre des étoiles binaires, ce n'est pas le soleil qui sert de point de référence mais bien le centre de masse commun des deux corps. Notons également que les systèmes binaires ne sont pas nécessairement composés de deux étoiles et que l'une des composantes peut se révéler être une exoplanète, un trou noir ou tout autre corps de masse non-nulle.

Soit $M = (r, \theta)$ les coordonnées polaires d'un corps autour de son centre de masse f_0. Notons a et b respectivement le semi-grand axe et le semi-petit axe de l'ellipse de pôle f_0 et d'origine P tel que représenté figure 1.1. La formulation des lois de Kepler devient[4] :

1. $$\left\{ M(r,\theta) \mid r = \frac{a(1-e^2)}{1+e\cos(\theta)}, \forall\theta \right\} \tag{1.1}$$
   où $e = \frac{c}{a}$ est l'excentricité de l'orbite

2. $$\frac{r^2}{2}\dot{\theta} = \frac{\pi a^2\sqrt{1-e^2}}{p} \tag{1.2}$$
   où $\dot{\theta}$ est la dérivée temporelle de θ et p la période orbitale

3. $a^3 \propto p^2$, où $\propto$ signifie que a^3 est directement proportionnel à p^2.
   Plus particulièrement[5], $$\frac{a^3}{p^2} = \frac{G}{4\pi^2}\left(\mathcal{M}_1 + \mathcal{M}_2\right) \tag{1.3}$$

où, $\mathcal{M}_1$ et $\mathcal{M}_2$ sont les masses de, respectivement, la primaire et la secondaire et où G est la constante gravitationnelle de Newton. Nous définissons également T_p comme l'instant de passage au périastre et A comme étant l'apoastre.

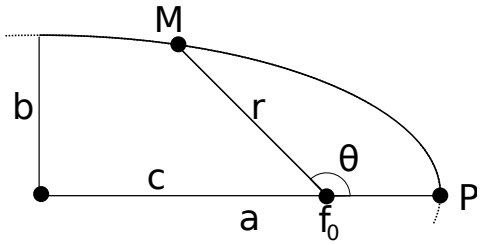

FIG. 1.1 – Représentation des coordonnées polaires de l'orbite Képlérienne

1.1 Anomalies de l'excentricité

En plus des paramètres décrits dans la section précédente, la première loi de Kepler nous permet de déterminer certains autres paramètres orbitaux de base. Soit une ellipse possédant un semi-grand axe a, un semi-petit axe b et deux foyers f_0 et f_1 où f_0 représente le centre de masse commun. Imaginons cette ellipse incluse dans un cercle de rayon a et de centre O ainsi qu'un corps céleste G à l'instant t. Ce schéma est représenté figure 1.2. Nous pouvons dès lors définir différentes anomalies de l'excentricité[4] (tableau 1.1).

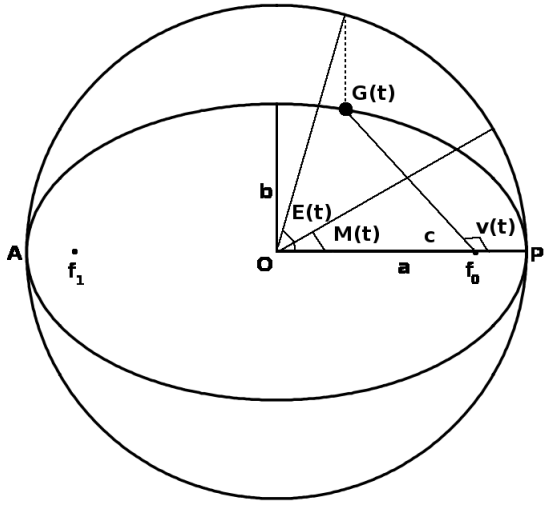

FIG. 1.2 – Géométrie des anomalies de l'excentricité dans les orbites Képlériennes

$M(t) = \dfrac{2\pi(t - T_p)}{p}$	:	L'anomalie moyenne
$E(t) = M(t) + e\sin(E(t))$	:	L'anomalie excentrique
$v(t) = 2\arctan\left(\sqrt{\dfrac{1+e}{1-e}}\tan\left(\dfrac{E(t)}{2}\right)\right)$	:	L'anomalie vraie

TAB. 1.1 – Formules des anomalies de l'excentricité de l'orbite Képlérienne à l'instant t

1.2 Paramètres de l'orbite apparente

Il est aisé de constater que les paramètres orbitaux de base sont insuffisants pour décrire une orbite telle que nous l'observons. Afin de tenir compte de l'observateur, il nous faut définir un plan et une direction de référence[4], soit P2 le plan tangent a la voute celeste. La projection de la direction Nord dans ce plan jour un role de reference. Définissons les noeuds orbitaux comme les points d'une orbite où le corps céleste traverse le plan de référence. Plus particulièrement, le nœud ascendant est le noœud orbital où le corps traverse le plan de référence tout en s'éloignant de l'observateur. Ce référentiel, illustré figure 1.3, nous permet de décrire les paramètres de l'orbite apparente donnés au tableau 1.2.

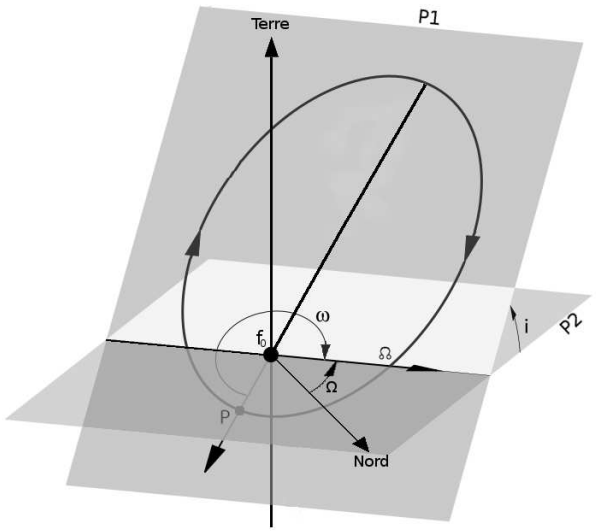

FIG. 1.3 – Géométrie des paramètres de l'orbite Képlérienne apparente

i	:	L'inclinaison du plan orbital P1 par rapport au plan de reference P2. En spectroscopie, l'angle i est compris entre 0 et 90 degres.
☊	:	Le nœud ascendant
Ω	:	La longitude du noeud ascendant, qui est l'angle entre la direction Nord dans P2 et le noeud ascendant ☊
ω	:	Argument du periastre, qui est l'angle mesure dans le plan de l'orbite, et dans le sens de parcours de celle-ci, entre la direction du noeud ascendant ☊ et le periastre P.
ϖ	:	La longitude du périastre, où $\varpi = \Omega + \omega$

TAB. 1.2 – Paramètres orbitaux apparents de l'orbite Képlérienne

Notons que la première loi de Kepler (formule 1.1) se doit d'être adaptée afin de tenir compte de l'inclinaison de l'orbite réelle par rapport au plan de référence et à la direction de référence, nous avons donc[2] :

1. $$\left\{ M(r,\theta) \mid r = \frac{a(1-e^2)}{1+e\cos(\theta-\varpi)}, \forall\theta \right\} \tag{1.4}$$

1.3 Méthodes de détection

Il existe plusieurs manières de détecter ces étoiles binaires :

- **Les binaires visuelles :** La séparation angulaire des étoiles est suffisante pour permettre une observation directe de la primaire et de la secondaire
- **Les binaires à éclipse :** Le plan de révolution des deux astres se trouve complètement ou en partie dans la ligne de vision de l'observateur. Un tel système procure une courbe de luminosité quasi-constante excepté en deux minima locaux et périodiques. Parmi ces minima, le minimum principal survient lorsque la secondaire éclipse la primaire et le minimum secondaire survient lorsque la primaire éclipse la secondaire. Une courbe théorique d'une étoile binaire à éclipse est illustrée à la figure 1.4.
- **Les binaires astrométriques :** La duplicité de l'étoile est révélée par le mouvement induit par la secondaire alors indétectable sur le photocentre au cours du temps. Le photocentre étant le barycentre de la luminosité du couple. Un exemple de binaire astrométrique est l'étoile Sirius A illustrée figure 1.5[6].
- **Les binaires spectroscopiques :** Le mouvement orbital est mis en évidence par la variation de vitesse radiale d'une ou de plusieurs composantes. Cette variation de la vitesse radiale est obtenue via l'effet Doppler-Fizeau des raies spectrales de la composante et mesurée à l'aide d'un spectromètre. Lorsque le spectre des deux composantes peut être mesuré, on parle d'étoile de type BS2. Lorsque seul le spectre de la primaire peut être mesuré, on parle d'étoile de type BS1. Dans le cadre de ce mémoire, nous nous intéresserons plus particulièrement aux équations du mouvement de ces dernières.

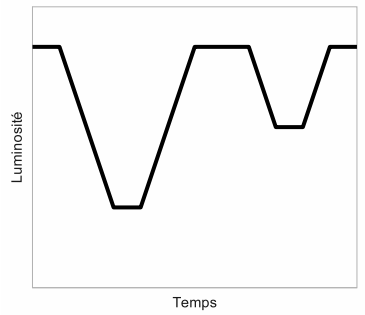

FIG. 1.4 – Courbe de la lumière théorique d'une étoile binaire à éclipse

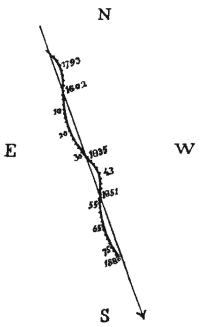

FIG. 1.5 – Sinuosités observées dans le mouvement propre de Sirius A

1.4 Vitesse radiale des étoiles binaires spectroscopiques

Soit l'équation de la vitesse radiale[7] d'une composante d'un système binaire sur son orbite

$$V_r(t) = V_0 + K\left[\cos\left(v\left(t\right) + \varpi\right) + e\cos(\varpi)\right] \tag{1.5}$$

où, V_0 est la vitesse du centre de masse et K la semi-amplitude de la courbe de vitesse radiale. Cette formule est obtenue au moyen des deux premières lois de Kepler. La courbe de vitesse radiale est donc définie au moyen de cinq paramètres (K, V_0, e, ϖ, p et T_p), dont deux dynamiques (K dépend de e et de p via la formule 1.6 et $v(t)$ dépend de e et T_p via la formule du tableau 1.1), ce qui la rend non-linéaire en les paramètres. Dans le cadre de ce mémoire nous supposerons que p est bien définie et ne doit pas être réajustée.

$$K = \frac{2\pi a \sin(i)}{p\sqrt{1-e^2}} \tag{1.6}$$

Analysons maintenant l'influence des différents paramètres sur l'allure de la courbe. L'influence des paramètres K et V_0 sur la courbe étant assez évidente à la vue de la formule 1.5, nous ne nous attarderons pas sur ces derniers. De même, sachant que T_p est l'instant de passage au périastre, l'ajout de ξ à T_p n'affectera la courbe finale qu'en un décalage de ξ de cette courbe le long de l'axe des abscisses. L'excentricité e contrôle quant à elle l'allure "pincée" de la courbe, allant de la simple sinusoïde lorsque l'orbite est circulaire à des pics de plus en plus abruptes en fonction de l'excentricité augmentante [4]. La longitude du périastre, ϖ, jouant quant à elle sur la symétrie de la courbe. Les influences des paramètres e et ϖ sur l'allure de la courbe peuvent être retrouvées sur les figures 1.6 et 1.7.

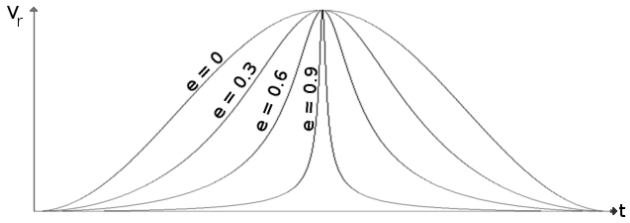

FIG. 1.6 – Influence de l'excentricité sur la courbe de vitesse radiale des binaires spectrales

1.4.1 Détermination de la vitesse radiale sur base de l'effet Doppler-Fizeau

Sur base de l'effet Doppler-Fizeau, nous pouvons déterminer la vitesse radiale de la composante primaire à l'instant t selon la formule[8]

$$V_r(t) = (\frac{\lambda(t)}{\lambda_0} - 1)c \tag{1.7}$$

où, $\lambda(t)$ est la longueur d'onde mesurée à l'instant t, λ_0 est la longueur d'onde qui serait obtenue si la composante primaire était au repos par rapport à l'observateur et c est la vitesse de la lumière.

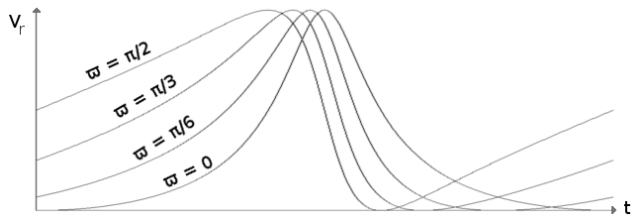

FIG. 1.7 – Influence de la longitude du périastre sur la courbe de vitesse radiale des binaires spectrales

1.4.2 Détermination des paramètres de l'équation de la vitesse radiale sur base de sa courbe

Analysons maintenant plus en détail la courbe produite par la formule 1.5 afin d'en extraire les paramètres à l'aide de la technique de Lehmann-Filhés[7]. Nous nous baserons pour cela sur la courbe représentée à la figure 1.8.

Par convention[2], la courbe est à son minimum et à son maximum respectivement lorsque $v(t) = \pi - \varpi$ et lorsque $v(t) = -\varpi$. Notons V_{min} et V_{max} la vitesse radiale minimale et maximale, nous pouvons déduire de la formule 1.5

$$K = \frac{V_{max} - V_{min}}{2} \tag{1.8}$$

Afin de déterminer la vitesse du centre de masse, il nous faut égaliser les aires situées au-dessus et en dessous de l'axe formé par V_0, autrement formulé

$$N_0 A N_1 + N_1 B N_2 = 0 \tag{1.9}$$

Notons que dans le cadre des étoiles BS2, nous obtenons deux courbes de vitesse radiale qui sont symétriques à un facteur d'échelle près[4] et que l'ordonnée de leurs intersections nous donne la valeur de V_0.

Une fois V_0 déterminé, nous pouvons déterminer e et ϖ grâce aux deux équations à deux inconnues obtenues en plaçant la vitesse soit au minimum, soit au maximum dans la formule 1.5[7], ce qui nous donne

$$e\cos(\varpi) = \frac{V'_{max} + V'_{min}}{V'_{max} - V'_{min}} \tag{1.10}$$

$$e\sin(\varpi) = \frac{2\sqrt{-V'_{max}V'_{min}}}{V'_{max} - V'_{min}} \frac{ACN_1 + DBN_2}{ACN_1 - DBN_2} \tag{1.11}$$

où

$$\begin{cases} V'_{max} = V_{max} - V_0 \\ V'_{min} = V_{min} - V_0 \end{cases}$$

Enfin, T_p peut être déterminé en prenant des points d'anomalie vraie connue, par exemple $v(t) = -\varpi$ et $v(t) = \pi - \varpi$ pour respectivement V_{max} et V_{min}, afin d'en déduire $M(t)$ et T_p en utilisant

$$\begin{cases} \cos(E(t)) &= \dfrac{\cos(v(t)) + e}{1 + e\cos(v(t)} \\ \sin(E(t)) &= \dfrac{\sqrt{1-e^2}\sin(v(t))}{1 + e\cos(v(t))} \\ M(t) &= E(t) - e\sin(E(t)) \end{cases} \tag{1.12}$$

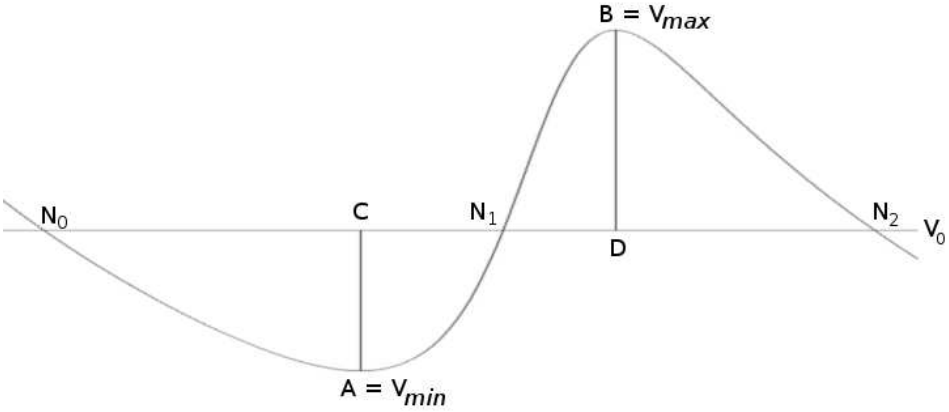

FIG. 1.8 – Courbe de vitesse radiale d'une étoile BS1 accompagnée de ses points importants

1.4.3 Détermination des contraintes des masses des composantes sur base des éléments orbitaux

Notre intérêt principal dans l'obtention des paramètres orbitaux est la détermination des masses des composantes du système binaire, ou dans une moindre mesure des contraintes sur ces masses. Sur base des lois Képlériennes, nous pouvons déterminer que[2]

$$\frac{a_1}{a_2} = \frac{\mathcal{M}_2}{\mathcal{M}_1} \tag{1.13}$$

où a_1 et a_2 sont les semi-grands axes de, respectivement, la primaire et de la secondaire et $\mathcal{M}_1$, $\mathcal{M}_2$ les masses de la primaire et de la secondaire. Si nous prenons p en jours et non plus en seconde, il vient de cette dernière équation et des formules 1.3 et 1.6

$$a\sin(i) = 1,375 \times 10^4 Kp\sqrt{1-e^2} \tag{1.14}$$

$$\mathcal{M}_{1,2}\sin(i)^3 = 1,0385 \times 10^{-7}(1-e^2)^{3/2}(K_1+K_2)^2 K_{2,1} p \tag{1.15}$$

où K_1 est la semi-amplitude de la primaire et K_2 celle de la secondaire.

Dans le cadre des étoiles BS1, l'information sur les masses se limite à une borne inférieure sur la masse de la primaire appelée *fonction de masse*[4] :

$$f_{\mathcal{M}} = 1,0385 \times 10^{-7}(1-e^2)^{3/2} Kp\mathcal{M}_{\odot} \tag{1.16}$$

où, $\mathcal{M}_{\odot}$ représente la masse solaire.

Si le spectre de la secondaire est également mesurable, nous pouvons, en plus de la formule 1.16, appliquer la formule 1.15. Notons néanmoins que cette dernière est tributaire de l'inclinaison i qui ne peut être déterminée par la courbe de vitesse radiale seule. Nous utiliserons dès lors une des autres techniques présentées section 1.3 afin de déterminer les masses des composantes de manière plus précise.

Chapitre 2

Objectifs du travail

La formule 1.7 nous permet de lier les observations réalisées à l'allure de la courbe de vitesse radiale. Cependant, malgré les progrès constants de la spectrométrie moderne, ces mesures sont entachées d'erreurs ; de telle sorte que même un astrophysicien expérimenté peut difficilement estimer les paramètres orbitaux sur base des seules mesures effectuées. L'origine de ce bruit est principalement à imputer à l'imprécision des instruments de mesure. En effet, selon la formule 1.7, une erreur de ξ sur la mesure de la longueur d'onde se traduit par une erreur de $\frac{\xi}{\lambda_0}c$ sur la mesure de la vitesse radiale (pour rappel $c \approx 300\ 000 km/s$).

Différentes méthodes ont vu le jour afin de déterminer les paramètres physiques de la courbe radiale, citons par exemple Heintz[9], Wilsing-Russel[7] et Lehmann-Filhès[7] que nous avons entrevue à la section 1.4.2. Ces méthodes sont utilisées comme solution approchée afin de les ajuster au mieux au moyen d'une méthode des moindres carrés telle que celle de Schlesinger[7]. Néanmoins, étant donné la non linéarité de la fonction 1.5 en ses paramètres, ces méthodes se basent sur des processus itératifs nécessitant une solution approchée assez fiable pour converger vers le minimum global. Les méthodes précédemment citées étant limitées, il est de coutume chez les astronomes de dériver ces solutions approchées de manière empirique, au cas par cas. Cependant, cette manière de procéder n'est pas viable dans le cadre du projet Gaia où le nombre d'étoiles traitées sera de l'ordre du million.

Nous chercherons donc dans le cadre de ce travail une procédure informatique basée sur les algorithmes d'apprentissage supervisé capable de déterminer une solution fiable en tant qu'alternative aux méthodes classiques existantes. Nous étudierons plus particulièrement le problème de la détermination de l'excentricité de l'orbite, celle-ci intervenant dans la détermination des contraintes des masses des composantes du système (cf. section 1.4.3) et dont l'évaluation robuste par les techniques actuelles fait défaut.

Afin de d'obtenir un éventail de résultats le plus large que possible, cette étude a été menée de manière parallèle au moyens de deux techniques issues de l'intelligence artificielle : les algorithmes d'apprentissages et les algorthmes génétiques. Chacune de ces deux techniques se révèle adaptée au problème de régression de l'évaluation de l'excentricité. Les algorithmes d'apprentissage grâce à leur capacité à mettre en évidence des modèles des données d'entrée afin d'extrapoler au mieux la sortie associée, et les algorithmes génétiques par leur capacité d'approximation des optima globaux.

Comme nous le verrons chapitre 3, les algorithmes d'apprentissage nécessitent un pré-traitement des données d'entrée. Nous pouvons dès lors schématiser les étapes effectuées dans le cadre de ce travail par la figure 2.1. Les différents processus

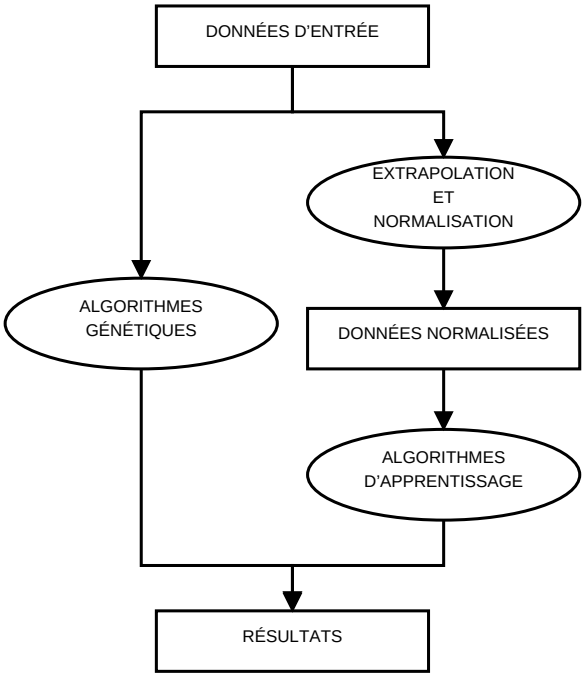

FIG. 2.1 – Diagramme de flux des différentes approches testées

illustrés faisant l'objet d'un développement théorique dans la suite de ce document.

Deuxième partie

Développements théoriques

Chapitre 3

Exploitation des données d'entrée

L'exploration des données et plus spécifiquement l'apprentissage supervisé, requiert des données d'entrée de qualité afin d'obtenir de bons résultats[10]. Or, les données réelles sont considérées comme impropres. En effet, elles peuvent être bruitées, inconsistantes et nous pouvons de surcroît avoir des données manquantes.

Dans le cadre du projet Gaia, le détecteur RVS nous fournira en moyenne 70 mesures par objet[1]. Ces mesures seront composées de la date d'observation t et de la vitesse radiale à l'instant t. Soit n le nombre d'observations effectuées sur l'un des objets, nous pouvons représenter ces observations par

$$P = \{(x_0, y_0), \ldots, (x_n, y_n)\} \tag{3.1}$$

où les x_i correspondent aux dates d'observation et les y_i aux vitesses radiales correspondantes pour $i = 0, \ldots, n$. Rappelons également que dans le cadre de ce mémoire, la période est supposée connue et n'a pas à être redéfinie. Nous pouvons dès lors effectuer une première normalisation afin que $0 \leq x_i \leq 1, i = 0, \ldots, n$. Nous parlerons dès lors de phase et non plus de date. Malgré cette première normalisation, nous pouvons constater que les données d'entrée souffrent encore d'imperfections, en effet :

- Elles sont **bruitées** (cf. chapitre 2)
- Elles sont **inconsistantes** : les observations ne sont pas effectuées à des phases identiques pour chacun des objets. Le nombre d'observations par objet peut varier et, dans le cas où seule l'excentricité de la courbe nous intéresse, certains paramètres de l'équation 1.5 nous apportent une information inutile qui, de plus, modifie l'allure de la courbe de vitesse radiale.

La nécessité d'une normalisation est donc bien présente afin de réduire la variabilité des différentes courbes d'entrée et par ce fait, de rendre l'extraction des données intéressantes plus aisée. Ce chapitre présente différents outils théoriques nous permettant d'y parvenir.

3.1 Méthodes d'interpolation de points

Le problème d'interpolation se définit comme la recherche d'une fonction $f(x)$ passant par un ensemble de points donnés. Soit $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$ cet ensemble de points, alors $f(x)$ est une interpolation de P si[11]

$$f(x_i) = y_i, \ \forall i = 0, \ldots, n \tag{3.2}$$

Notons que dans notre recherche d'une normalisation des données d'entrée, les techniques présentées dans cette section n'apportent aucune amélioration au niveau de la réduction du bruit. Précisons également que l'entièreté des techniques vues dans cette section sera basée sur les splines. Pour rappel[12], une spline est une fonction $S : [a, b] \to \mathbb{R}$ telle que pour $k + 1$ valeurs t appelés nœuds où

$$a = t_0 \leq t_1 \leq \ldots \leq t_{k-1} \leq t_k = b \tag{3.3}$$

on a k fonctions

$$S_i : [t_i, t_{i+1}] \to \mathbb{R}, i = 0, \ldots, k-1 \tag{3.4}$$

telles que

$$S(t) = \begin{cases} S_0(t), & t_0 \leq t < t_1 \\ \vdots & \\ S_i(t), & t_i \leq t < t_{i+1} \\ \vdots & \\ S_{k-1}(t), & t_{k-1} \leq t \leq t_k \end{cases} \tag{3.5}$$

Pour plus de clarté et de simplicité, nous supposerons dans la suite de ce document que les abscisses de P sont toutes distinctes, ordonnées et normalisées, plus particulièrement $0 = x_0 < x_1 < \ldots < x_i < \ldots < x_n = 1$ et $y_0 = y_n$. Nous avons également $t_i = x_i, \ \forall i = 0, \ldots, n$.

3.1.1 Interpolation des plus proches voisins

L'interpolation des plus proches voisins ou spline constante[13] représente la technique d'interpolation la plus simple que l'on puisse imaginer. Elle consiste en effet à extrapoler les valeurs manquantes par la valeur du point le plus proche.

Mathématiquement, soit $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, la spline associée aux noeuds x_i et x_{i+1} est définie par

$$S_i(x) = \begin{cases} y_i, & x - x_i < x_{i+1} - x \\ y_{i+1}, & \text{sinon} \end{cases} \tag{3.6}$$

3.1.2 Interpolation linéaire

Cette technique est également l'une des plus simples et consiste à relier les différents nœuds par des lignes droites[14].

Mathématiquement, soit $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, la spline associée aux noeuds x_i et x_{i+1} est définie par

$$S_i(x) = y_i + (x - x_i)\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{3.7}$$

Nous pouvons aisément constater que cette équation est bien celle d'une droite passant par les points (x_i, y_i) et (x_{i+1}, y_{i+1}). La spline ainsi créée est donc bien continue mais, est non différentiable en ses nœuds, elle est de continuité $\mathcal{C}^0$.

3.1.3 Interpolation cosinus

Comme nous venons de le voir, le principal défaut de l'interpolation linéaire est sa non différentiabilité au niveau de ses nœuds, ce qui lui donne un aspect "rugueux". Dans de nombreuses situations nous souhaitons obtenir une fonction d'interpolation plus souple. Afin d'obtenir une telle transition à moindre coût nous pouvons utiliser l'interpolation cosinus[15]. Une comparaison graphique de ces deux dernières méthodes est proposée figure 3.1.

Soit $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, la spline associée aux noeuds x_i et x_{i+1} est alors définie par

$$S_i(x) = y_i(1-\mu) + y_{i+1}\mu \tag{3.8}$$

où $\mu = \dfrac{1 - \cos\left(\dfrac{x - x_i}{x_{i+1} - x_i}\pi\right)}{2}$.

Nous pouvons constater que cette spline est de continuité $\mathcal{C}^1$. Plus particulièrement, $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}) = 0,\ \forall i = 0, \ldots, n-2$.

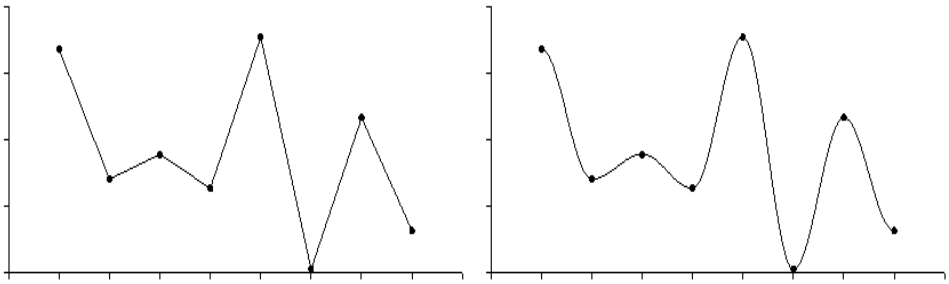

FIG. 3.1 – Comparaison de l'interpolation linéaire et de l'interpolation cosinus

3.1.4 Interpolation locale d'Akima périodique

Soit une spline polynomiale de degré 3, $S(x)$, et un ensemble de points $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$. La spline associée aux noeuds x_i et x_{i+1} est alors définie par

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{3.9}$$

où la connaissance de deux points accompagnés de leur tangente nous permet de déterminer de manière unique a_i, b_i, c_i et d_i. Plus particulièrement, des équations suivantes

$$\begin{aligned} S_i(x_i) &= y_i, \quad S_i(x_{i+1}) = y_{i+1} \\ S_i'(x_i) &= \delta_i, \quad S_i'(x_{i+1}) = \delta_{i+1} \end{aligned} \tag{3.10}$$

il vient

$$a_i = \frac{\delta_i + \delta_{i+1} - 2\mu}{(x_{i+1} - x_i)^2} \tag{3.11}$$

$$b_i = \frac{3\mu - 2\delta_i - \delta_{i+1}}{x_{i+1} - x_i} \tag{3.12}$$

$$c_i = \delta_i \tag{3.13}$$

$$d_i = y_i \tag{3.14}$$

où $\mu = \dfrac{y_{i+1} - y_i}{x_{i+1} - x_i}$

Hiroshi Akima[16] proposa dans les années 70 une méthode pour déterminer les tangentes δ_i et δ_{i+1} de manière à obtenir une courbe lisse et naturelle. Il détermina également, aux travers de nombreuses comparaisons, que celle-ci était celle qui se rapprochait le plus d'une courbe interpolante qui serait dessinée à la main[16].

Soit $Q = \{(x_{i-2}, y_{i-2}), (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2})\}$, la tangente δ_i en x_i peut être déterminée de la manière suivante :

$$\delta_i = \frac{m_1|m_3 - m_2| + m_2|m_1 - m_0|}{|m_3 - m_2| + |m_1 - m_0|} \tag{3.15}$$

où $m_i = \dfrac{y_{i-1} - y_{i-2}}{x_{i-1} - x_{i-2}}$, $i = 0, \ldots, 3$, sont les pentes des droites liant les différents points. La figure 3.1.4 illustre les différents points importants lors de la détermination de la tangente δ_i.

Nous pouvons constater que nous avons besoin d'évaluer deux autres points à chaque extrêmité de la courbe. Dans son papier, Hiroshi Akima résolvait ce problème en interpolant les trois premiers (derniers) points à l'aide d'une parabole[16]. Cependant, dans le cadre de ce mémoire, la fonction étudiée étant périodique et la période étant supposée connue et ayant été normalisée à 1, nous pouvons travailler avec l'ensemble de points suivant :

$$P' = \{(x_{n-1} - 1, y_{n-1}), (x_n - 1, y_n), P, (x_0 + 1, y_0), (x_1 + 1, y_1)\} \tag{3.16}$$

Notons qu'une attention toute particulière doit être prise en compte lorsque $x_0 = x_n - 1 = 0$. Comme cela est le cas dans le cadre de ce mémoire, nous travaillerons donc avec l'ensemble de points

$$P'' = \{(x_{n-2} - 1, y_{n-2}), (x_{n-1} - 1, y_{n-1}), P, (x_1 + 1, y_1), (x_2 + 1, y_2)\} \tag{3.17}$$

À la vue des formules 3.10, nous pouvons conclure que cette spline est également de continuité $\mathcal{C}^1$.

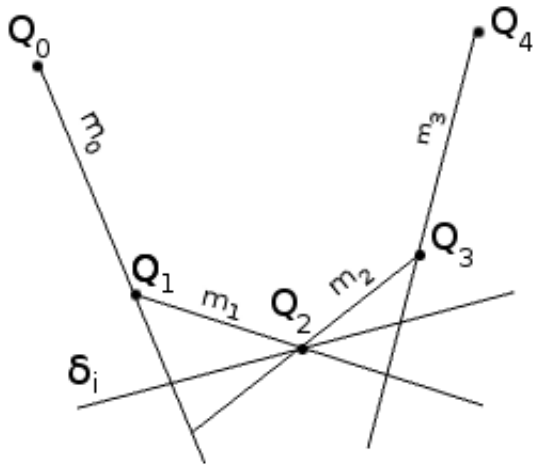

FIG. 3.2 – Points importants lors de la détermination de la tangente δ_i avec la méthode d'Akima

3.1.5 Interpolation par splines cubiques périodiques

Repartant de la formule 3.9, nous cherchons dans le cas des splines cubiques à minimiser l'énergie potentielle tout en conservant les contraintes de l'interpolation[15]. La spline ainsi créée aura de ce fait une continuité $\mathcal{C}^2$.

Soit $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, nous cherchons donc n fonctions

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad i = 0, \ldots, n-1 \tag{3.18}$$

sujettes aux contraintes

$$S_i(x_i) = S_{i-1}(x_i) = y_i, \quad i = 1, \ldots, n-1 \tag{3.19}$$

$$S_i'(x_i) = S_{i-1}'(x_i), \quad i = 1, \ldots, n-1 \tag{3.20}$$

$$S_i''(x_i) = S_{i-1}''(x_i), \quad i = 1, \ldots, n-1 \tag{3.21}$$

$$S_0(x_0) = y_0 \tag{3.22}$$

$$S_{n-1}(x_n) = y_n \tag{3.23}$$

Nous obtenons un système de $4n-2$ équations à $4n$ inconnues. Nous devons en effet déterminer les conditions qui prévalent aux extrémités. Différentes solutions ont vu le jour afin d'établir ces dernières. Si les dérivées premières aux extrémités peuvent aisément être trouvées, comme par exemple avec la méthode proposée par Hiroshi Akima (cf. section 3.1.4), nous parlerons de spline cubique à collier. Si par contre, nous laissons les conditions libres aux extrémités, alors les conditions $S_0''(x_0) = 0$ et $S_{n-1}''(x_n) = 0$ seront prévalentes[17] et nous parlerons dès lors de spline cubique naturelle. Néanmoins, dans le cadre de ce mémoire, la fonction étant périodique et les conditions $0 = x_0 < x_1 < \ldots < x_i < \ldots < x_n = 1$ et $y_0 = y_n$ ayant été fixées, nous nous baserons sur les conditions suivantes

$$S_0'(x_0) = S_{n-1}'(x_n) \tag{3.24}$$

$$S_0''(x_0) = S_{n-1}''(x_n) \tag{3.25}$$

Posons $b_n = b_0$, selon les contraintes 3.19, 3.21, 3.22, 3.23 et 3.25, les différents paramètres de la formule 3.18 peuvent dès lors s'écrire [1] :

$$a_i = \frac{b_{i+1} - b_i}{3h_i}, \quad i = 0, \ldots, n-1 \tag{3.26}$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{(b_{i+1} + 2b_i)h_i}{3}, \quad i = 0, \ldots, n-1 \tag{3.27}$$

$$d_i = y_i, \quad i = 0, \ldots, n-1 \tag{3.28}$$

avec $h_i = x_{i+1} - x_i$. En substituant dans les contraintes 3.20 et 3.24, nous obtenons

$$b_{i-1}h_{i-1} + 2b_i(h_{i-1} + h_i) + b_{i+1}h_i = \frac{3(y_{i+1} - y_i)}{h_i} - \frac{3(y_i - y_{i-1})}{h_{i-1}}, \quad i = 1, \ldots, n-1 \tag{3.29}$$

et

$$b_{n-1}h_{n-1} + 2b_0(h_{n-1} + h_0) + b_1h_0 = \frac{3(y_1 - y_0)}{h_0} - \frac{3(y_0 - y_{n-1})}{h_{n-1}} \tag{3.30}$$

[1] cf. D.S.G. POLLOCK, SMOOTHING WITH CUBIC SPLINES, p.4[17] pour la démonstration complète

ou sous forme matricielle

$$\begin{pmatrix} p_0 & h_0 & 0 & 0 & \dots & 0 & 0 & h_{n-1} \\ h_0 & p_1 & h_1 & 0 & \dots & 0 & 0 & 0 \\ 0 & h_1 & p_2 & h_2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & h_{n-3} & p_{n-2} & h_{n-2} \\ h_{n-1} & 0 & 0 & 0 & \dots & 0 & h_{n-2} & p_{n-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ \vdots \\ q_{n-2} \\ q_{n-1} \end{pmatrix} \tag{3.31}$$

avec

$$p_i = 2(h_{i-1} + h_i) = 2(x_{i+1} - x_{i-1}), \quad i = 1, \dots, n-1 \tag{3.32}$$

$$p_0 = 2(h_{n-1} + h_0) = 2(x_n - x_{n-1} + x_1 - x_0) \tag{3.33}$$

$$q_i = \frac{3(y_{i+1} - y_i)}{h_i} - \frac{3(y_i - y_{i-1})}{h_{i-1}}, \quad i = 1, \dots, n-1 \tag{3.34}$$

$$q_0 = \frac{3(y_1 - y_0)}{h_0} - \frac{3(y_0 - y_{n-1})}{h_{n-1}} \tag{3.35}$$

Une fois le vecteur de paramètres b connu, nous pouvons utiliser les formule 3.26 et 3.27 pour obtenir les vecteurs de paramètres a et c.

3.1.6 Interpolation par splines de Catmull-Rom périodiques

Les splines de Catmull-Rom sont des splines cubiques d'interpolation locale sous forme paramétrique $S : [0,1] \to \mathbb{R}^n$. Elles ont été développées dans le cadre de la modélisation géométrique et plus particulièrement dans le domaine de l'infographie et des films d'animation[18]. Elles se basent sur des arguments géométriques simples afin de déterminer le vecteur tangent, $\overrightarrow{m_i}$, au point de contrôle i.

Soit $P = \{(x_0, y_0), \dots, (x_n, y_n)\}$ et plus particulièrement $P_i = (x_i, y_i)$. Nous supposerons dans un premier temps les vecteurs tangents aux points P_i et P_{i+1} connus, soit $\overrightarrow{m_i}$ et $\overrightarrow{m_{i+1}}$. La fonction de base de la spline se définit alors comme $S_i : [0,1] \to \mathbb{R}^2$:

$$S_i(t) = a_i t^3 + b_i t^2 + c_i t + d_i \tag{3.36}$$

sujette aux contraintes

$$\begin{array}{ll} S_i(0) = P_i & S_i(1) = P_{i+1} \\ S_i'(0) = \overrightarrow{m_i} & S_i'(1) = \overrightarrow{m_{i+1}} \end{array} \tag{3.37}$$

en résolvant ces équations pour a_i, b_i, c_i et d_i, nous obtenons[18]

$$\begin{array}{l} a_i = 2(P_i - P_{i+1}) + \overrightarrow{m_i} + \overrightarrow{m_{i+1}} \\ b_i = -3(P_i - P_{i+1}) - 2\overrightarrow{m_i} - \overrightarrow{m_{i+1}} \\ c_i = \overrightarrow{m_i} \\ d_i = P_0 \end{array} \tag{3.38}$$

et en substituant dans la formule 3.36,

$$S_i(t) = h_{00}(t)P_i + h_{01}(t)P_{i+1} + h_{10}(t)\overrightarrow{m_i} + h_{11}(t)\overrightarrow{m_{i+1}} \tag{3.39}$$

où

$$\begin{array}{l} h_{00}(t) = 2t^3 - 3t^2 + 1 \\ h_{01}(t) = -2t^3 + 3t^2 \\ h_{10}(t) = t^3 - 2t^2 + t \\ h_{11}(t) = t^3 - t^2 \end{array} \tag{3.40}$$

sont les fonctions de base hermitienne.

Afin d'obtenir un contrôle local de la courbe, nous nous baserons uniquement sur le point précédent et suivant P_i afin de déterminer le vecteur $\overrightarrow{m_i}$, soit respectivement P_{i-1} et P_{i+1} le point suivant et précédent P_i. Nous pouvons également raisonnablement supposer qu'afin d'obtenir une interpolation lisse[18], les vecteurs $\overrightarrow{P_{i-1}P_{i+1}}$ et $\overrightarrow{m_i}$ doivent être colinéaires et de même sens. Mathématiquement

$$\overrightarrow{m_i} = k\overrightarrow{P_{i-1}P_{i+1}} \tag{3.41}$$

avec $k \geq 0$. Nous supposerons également que la norme du vecteur $\overrightarrow{m_i}$ doit être pondérée. De manière simple, nous prendrons

$$\|\overrightarrow{m_i}\| = \frac{\|\overrightarrow{P_{i-1}P_{i+1}}\|}{2} \tag{3.42}$$

dès lors, à la vue des formules 3.41 et 3.42, nous pouvons définir $\overrightarrow{m_i}$ comme

$$\overrightarrow{m_i} = \frac{P_{i+1} - P_{i-1}}{2} \tag{3.43}$$

Afin de tenir compte de la périodicité de la formule 1.5, nous définirons plus particulièrement

$$\begin{aligned} \overrightarrow{m_0} &= \overrightarrow{m_n} = \frac{P_1 - P_{n-1}}{2} \\ \overrightarrow{m_i} &= \frac{P_{i+1} - P_{i-1}}{2}, \qquad i = 1, \ldots, n-1 \end{aligned} \tag{3.44}$$

La spline de Catmull-Rom associée à l'ensemble de points P peut alors être définie par

$$S(t) = \begin{cases} S(t) = S_{\lfloor nt \rfloor}(nt - \lfloor nt \rfloor), & t < 1 \\ S(1) = P_n \end{cases} \tag{3.45}$$

où $\lfloor nt \rfloor \in \mathbb{Z}$ représente la partie entière de nt. Nous pouvons constater que cette spline est de continuité $\mathcal{C}^1$.

3.2 Méthodes d'ajustement de courbes

Comme nous l'avons vu dans la section précédente, les techniques d'interpolation ne permettent aucune réduction du bruit. Au contraire, certaines d'entres elles en amplifient même l'impact (cf. chapitre 8). Afin de pallier à ce défaut, nous pouvons utiliser des méthodes d'ajustement de courbes où, sur base d'un ensemble de points $\{(x_i, y_i) |\ i = 0, \ldots, n\}$, $x_i \in \mathbb{R}$ et $y_i \in \mathbb{R}$, nous cherchons à mininmiser une certaine méthode d'évaluation. Dans les problèmes de régression tel celui-ci nous utiliserons couramment la méthode des moindres carrés, où l'on cherche à minimiser

$$R^2 = \sum_{i=0}^{n} \left(y_i - \hat{f}(x_i) \right)^2 \tag{3.46}$$

3.2.1 Ajustement par un polynôme

Soit un ensemble de points $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$ et un polynôme $\hat{P}$ de degré k, ce polynôme peut être représenté sous la forme[19]

$$\hat{P}(x) = \sum_{i=0}^{k} a_i x^i \tag{3.47}$$

Selon la formule 3.46, nous cherchons ici à minimiser

$$R^2 = \sum_{i=0}^{n} \left(y_i - \hat{P}(x_i) \right)^2 \tag{3.48}$$

afin d'en déterminer le minimum, fixons les dérivées partielles à zéro,

$$\frac{\partial R^2}{\partial a_j} = -2 \sum_{i=0}^{n} \left(y_i - \hat{P}(x_i) \right) x^j = 0, \quad j = 0, \ldots, k \tag{3.49}$$

ou sous forme matricielle

$$M^T M \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{pmatrix} = M^T \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{3.50}$$

où

$$M = \begin{pmatrix} 1 & x_0 & \ldots & x_0^k \\ 1 & x_1 & \ldots & x_1^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^k \end{pmatrix} \tag{3.51}$$

Ce système linéaire peut être résolu à l'aide d'un algorithme de Gauss-Jordan classique[20]. Notons que dans le cadre de ce mémoire, le degré k du polynôme $\hat{P}$ est déterminé de manière itérative au moyen d'un test statistique de Fisher-Snedecor[7] pour $k = 0, \ldots, n-1$.

3.2.2 Ajustement par séries de Fourier

Soit un ensemble de points $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$ et une série de Fourier $\hat{F}$ composée de k harmoniques, cette série de Fourier peut être représentée sous la forme[21]

$$\hat{F}(x) = \sum_{i=0}^{k} a_i \cos(f\ i\ x) + b_i \sin(f\ i\ x) \tag{3.52}$$

où $f = \dfrac{2\pi}{T}$ représente la fréquence de la fonction à ajuster et T la période de la fonction. Nous pouvons également prouver que l'expansion des courbes de vitesse radiale des étoiles binaires spectrales en série de Fourier peut être représentée sous la forme suivante[22][7]

$$V_r(t) = V_0 + \sum_{n=1}^{\infty} \alpha_n \cos(nM_i) + \sum_{n=1}^{\infty} \beta_n \sin(nM_i) \tag{3.53}$$

où $M_i = 2\pi(t_i - T_p)$. Les séries de Fourier sont donc un outil de choix dans le cadre de notre étude. Selon le formule 3.46, nous cherchons ici à minimiser

$$R^2 = \sum_{i=0}^{n} \left(y_i - \hat{F}(x_i)\right)^2 \tag{3.54}$$

afin de déterminer le minimum, fixons les dérivées partielles à zéro,

$$\begin{aligned} \frac{\partial R^2}{\partial a_j} &= \textstyle\sum_{i=0}^{n} \left(y_i - \hat{F}(x_i)\right) \cos(f\ j\ x_i) = 0 \\ & \qquad\qquad\qquad\qquad\qquad\qquad j = 0, \ldots, k \\ \frac{\partial R^2}{\partial b_j} &= \textstyle\sum_{i=0}^{n} \left(y_i - \hat{F}(x_i)\right) \sin(f\ j\ x_i) = 0 \end{aligned} \tag{3.55}$$

ou sous forme matricielle

$$M^T M \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ \vdots \\ a_k \\ b_k \end{pmatrix} = M^T \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{3.56}$$

où

$$M = \begin{pmatrix} 1 & \cos(fx_0) & \sin(fx_0) & \ldots & \cos(fkx_0) & \sin(fkx_0) \\ 1 & \cos(fx_1) & \sin(fx_1) & \ldots & \cos(fkx_1) & \sin(fkx_1) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \cos(fx_n) & \sin(fx_n) & \ldots & \cos(fkx_n) & \sin(fkx_n) \end{pmatrix} \tag{3.57}$$

Le nombre d'harmoniques étant également déterminé itérativement au moyen d'un test de Fisher-Snedecor[7] pour $k = 0, \ldots, \dfrac{n-1}{2}$.

3.2.3 Ajustement par une courbe de Bézier

Les courbes de Bézier sont des courbes paramétriques étant définies par un ensemble de points appelé polygone de contrôle. Elles ont été inventées dans le cadre de la modélisation et la fabrication de pièces automobiles, mais trouvent actuellement de nombreuses applications dans le domaine des images de synthèse et plus généralement de l'infographie. Elles ne font pas partie des méthodes d'ajustement de courbes au sens des moindres carrés mais se révèlent intéressantes de par leur capacité à approximer leur polygone de contrôle tout en restant dans son enveloppe convexe[23]. Soit $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$ et plus particulièrement $P_i = (x_i, y_i)$, une courbe de Bézier $\hat{B}$ de degré n sera entièrement définie par ces $n+1$ points de contrôle et formulée comme

$$\hat{B}(t) = \sum_{i=0}^{n} P_i B_i^n(t) \tag{3.58}$$

où $B_i^n(t)$ est le $i^{\text{ème}}$ polynôme de Bernstein de degré n défini par

$$B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i} \tag{3.59}$$

où $\begin{pmatrix} n \\ i \end{pmatrix} = \frac{!n}{!i!(n-i)}$ sont les coefficients binomiaux. Nous supposerons également dans la définition précédente que $0^0 = 1$.

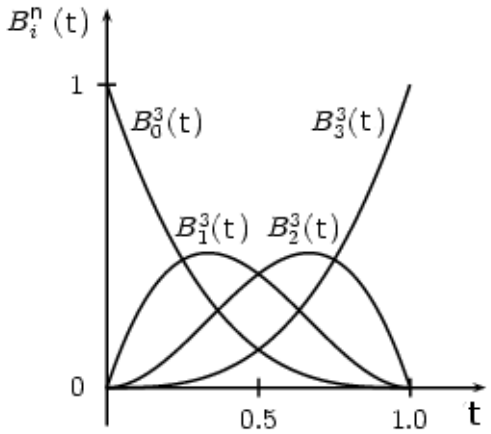

FIG. 3.3 – Les trois polynômes de Bernstein de degré 3

Si nous analysons plus en détail le comportement des courbes des polynômes de Bernstein, tel qu'illustré figure 3.3 pour $n = 3$, nous pouvons constater à la vue de la formule 3.58 que la courbe de Bézier passera par les points P_0 et P_n et que chaque point influencera de manière globale l'allure de la courbe. Nous pouvons également constater que $\sum_{i=0}^{n} B_i^n(t) = 1$ et que $B_i^n(t) \geq 0$[23], autrement formulé, la courbe de Bézier sera comprise dans l'enveloppe convexe définie par ses points de contrôle.

Algorithme de De Casteljau

Les courbes de Bézier sous forme de Bernstein souffrent d'une faible stabilité numérique. En effet, les calculs des polynômes de Bernstein $B_n^n(t)$ où $n > 20$, se révèlent plus qu'approximatifs[20]. L'algorithme de De Casteljau apporte une

solution à ce problème en se basant sur des arguments géométriques simples (cf. figure 3.4). Ce dernier permet, si nous reprenons l'ensemble de points P, d'estimer la valeur de la courbe de Bézier en un point donné t par $\hat{B}(t) = P_0^n(t)$ où

$$P_i^j(t) = \begin{cases} P_i, & j = 0 \\ (1-t)P_i^{j-1} + tP_{i+1}^{j-1} \end{cases} \tag{3.60}$$

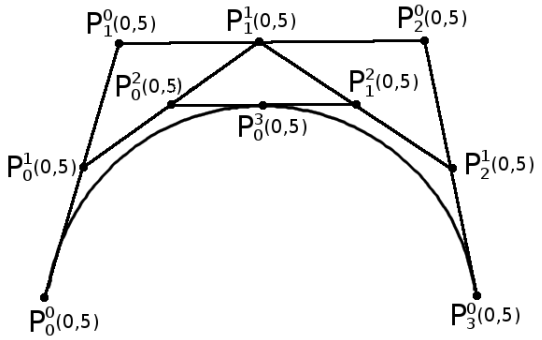

FIG. 3.4 – Illustration de la géométrie de l'algorithme de De Casteljau pour $n = 3$ et $t = 0,5$

3.2.4 Ajustement par une B-spline cubique ouverte uniforme

Comme nous l'avons vu dans la section précédente, les courbes de Bézier ne possèdent aucun contrôle local de la courbe et qui plus est, leur degré est directement défini sur base du nombre de points du polygône de contrôle. Les B-splines surpassent ces défauts en généralisant la technique des courbes de Bézier. Elles sont, tout comme les différentes splines vues section 3.1, définies par parties mais, au contraire de ces dernières, le vecteur des nœuds K n'est pas directement dépendant des points à approximer. En effet, dans ce cas, le vecteur K est composé de nombres réels et non de points.

Il existe de nombreuses variantes des B-splines, citons entre autres les B-splines uniformes où les nœuds sont répartis de manière uniforme et où toutes les fonctions de poids ont la même forme, les B-splines non-uniformes où les nœuds ne sont pas nécessairement équidistants et où les fonctions de poids n'ont pas spécialement la même forme et les B-splines non-uniformes rationnelles (NURBS) où la fonction de poids est spécifiée comme un ratio de polynômes en t[23]. Notons finalement que les NURBS sont la plus haute généralisation possible des B-splines et donc, des courbes de Bézier.

Nous nous attarderons plus particulièrement sur les splines ouvertes uniformes qui sont une forme particulière des B-splines non-uniformes. Nous étudierons donc ces dernières au travers des B-splines non-uniformes afin d'ensuite en définir la spécificité. Une B-spline non-uniforme de degré k est définie au moyen d'un ensemble de points $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$ et plus particulièrement $P_i = (x_i, y_i)$ et un vecteur de $n + k + 1$ réels $K = \{t_0, \ldots, t_{n+k}\}$, elle peut être formulée comme[23]

$$\hat{S}(t) = \sum_{i=0}^{n} P_i N_i^k(t), \quad t_0 \leq t \leq t_n \tag{3.61}$$

où $N_{ik}(t)$, la fonction de poids, peut être définie de manière récursive par

$$N_i^k(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_i^{k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1}^{k-1}(t) \tag{3.62}$$

avec $N_i^1(t) = \begin{cases} 1, t \in [t_i, t_{i+1}[\\ 0, \text{ sinon} \end{cases}$

où $0/0 = 1$. Si nous analysons plus en détail le comportement des fonctions de poids N_i^k par rapport à la formule ci-dessus(cf. figure 3.5), nous pouvons constater plusieurs choses. Tout d'abord, chaque fonction de poids de degré k est en fait une spline composée de segments polynomiaux de degré $k-1$. Ensuite, elles sont des versions décalées l'une de l'autre, ce qui donne lieu à une optimisation du calcul dans le cadre des B-splines uniformes. Nous pouvons également constater que lorsque les nœuds sont équidistants, la courbe ne passe pas par P_0, ni par P_n. Afin d'obtenir une courbe fermée, nous pouvons modifier le vecteur des nœuds afin de tirer la courbe vers les points extrêmes, de manière simple pour une B-spline de degré k :

$$K = \{t_0 = t_1 = \ldots = t_{k-1}, K', t_{n+1} = \ldots = t_{n+k}\} \tag{3.63}$$

Si de plus les nœuds $K'' = \{t_{k-1}, K', t_{n+1}\}$ sont équidistants, nous sommes dès lors en présence d'une B-spline ouverte uniforme. Une manière simple de générer les différents nœuds afin d'obtenir ce type de B-spline est la suivante

$$t_i = \begin{cases} 0, & 0 \leq i < k, \\ i - k + 1, & k \leq i \leq n, \\ n - k + 2, & n < i \leq n + k \end{cases}, \quad 0 \leq i \leq n + k \tag{3.64}$$

où l'équation de la B-spline devient alors

$$\hat{S}(t) = \sum_{i=0}^{n} P_i N_i^k(nt), \quad 0 \leq t \leq 1 \tag{3.65}$$

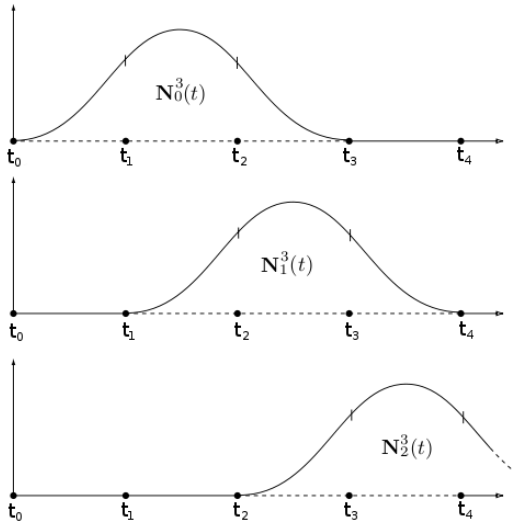

FIG. 3.5 – Illustration des fonctions de poids d'une B-spline uniforme de degré 3

3.3 Normalisation de l'échantillonnage

Comme nous l'avons vu précédemment, l'excentricité e de la courbe des vitesses radiales est notre principale préoccupation et à la vue de la formule 1.5, nous pouvons constater que les paramètres V_0, K, ϖ et T_p n'influencent pas le paramètre e et donc, nous apportent des informations supplémentaires inutiles sur l'allure de la courbe. Rappelons également que les données d'entrée ont déjà subi une première normalisation par rapport à la période p afin que $0 \leq x_i \leq 1, i = 0, \ldots, n$.

De plus, toujours à la vue de la formule 1.5, nous pouvons constater que la normalisation de certains paramètres, V_0, K et T_p pour ne pas les citer, se révèle plus qu'aisée. Le paramètre ϖ étant quant à lui plus délicat à déterminer et devant lui même faire l'objet d'une phase d'apprentissage afin d'être approximé, il ne sera tout simplement pas normalisé.

Soit $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, un ensemble de données d'entrée, $\hat{f}(t)$ une fonction d'ajustement de courbe basée sur P telle que présentée section 3.1 et 3.2. Nous pouvons définir l'échantillon de $\hat{f}(t)$ comme

$$P' = \left\{ \left(x'_0 = t_0, y'_0 = \hat{f}(t_0) \right), \ldots, \left(x'_{n'} = t_{n'}, y'_{n'} = \hat{f}(t_{n'}) \right) \right\} \tag{3.66}$$

pour les fonctions cartésiennes et

$$P' = \left\{ P'_0 = \hat{f}(t_0), \ldots, P'_{n'} = \hat{f}(t_{n'}) \right\}, \quad \text{pour les fonctions paramétriques} \tag{3.67}$$

où $T = \{t_0, \ldots, t_{n'}\}$ contient des valeurs échantillonnées de manière uniforme sur l'intervale $[0, 1]$, et où $P'_i = \{(x'_i, y'_i)\}, i = 0, \ldots, n'$. Nous verrons dans la partie pratique de ce document (section 6.1), une manière de déterminer n' de façon optimale, nous le supposerons actuellement connu et fixé.

Si nous analysons plus particulièrement la formule 1.5 de la courbe des vitesses radiales par rapport à l'instant de passage au périastre T_p, nous pouvons constater que ce dernier n'intervient que dans la formulation de l'anomalie moyenne $\left(M(t) = \dfrac{2\pi(t - T_p)}{p} \right)$. L'effet de T_p sur l'allure de la courbe se traduit donc par un décalage de cette dernière le long de l'axe des abscisses[7]. Fixons de manière arbitraire l'estimation de T_p à l'abscisse correspondant à la vélocité minimale trouvée par l'ajustement des courbes. Soit $\hat{T_p}$, cette estimation, nous pouvons écrire

$$\hat{T_p} = x'_i \quad | \quad y'_i = \min y'_j, \quad i, j \in 0, \ldots, n' \tag{3.68}$$

L'estimation des paramètres K et V_0 est basée sur la méthode de Lehmann-Filhés entrevue à la section 1.4.2[7]. Selon la formule 1.8 nous avons

$$K = \frac{V_{max} - V_{min}}{2} \tag{3.69}$$

où V_{max} et V_{min} sont respectivement la vitesse radiale maximale et minimale. Si notre but est, à ce niveau, d'uniformiser au mieux l'ensemble des courbes possédant les mêmes excentricités et longitudes du périastre, il nous faut garder à l'esprit qu'une certaine variabilité entre courbes possédant des paramètres e et ϖ différents est nécessaire afin de permettre à l'algorithme d'apprentissage supervisé de différencier au mieux ces courbes, et donc d'approximer au mieux leur excentricité. Sachant que K est dépendant de e (cf. section 1.4) et que les différentes techniques d'ajustement de courbes nous fournissent des amplitudes de courbes différentes, nous pouvons dès lors affirmer que les amplitudes calculées sur base des ajustements de courbes contiennent une information importante qui ne doit pas être perdue. Nous

nous baserons dès lors directement sur les données d'entrée afin de déterminer la vitesse radiale minimale et maximale. Mathématiquement,

$$\hat{K} = \frac{\max y_i - \min y_j}{2}, \quad i, j \in 0, \ldots, n \tag{3.70}$$

L'approximation de la vitesse du centre de masse V_0 est déterminée par la formule 1.9. Pour rappel, l'aire située au-dessus de l'axe formé par V_0 doit être égale à l'aire située en dessous de cet axe. La technique de Lehmann-Filhés propose de calculer ces aires sur base de l'interpolation linéaire des points d'entrée. Pour rappel, nous avons $x_0 = 0$, $x_n = 1$ et $y_0 = y_n$, cette duplication au niveau des points extrêmes nous permet de tenir compte de l'entièreté des aires définies sur la période. La robustesse de cette technique ayant été prouvée[7], nous pouvons directement travailler avec les données d'entrée P et estimer $\hat{V_0}$ comme[7]

$$\hat{V_0} = \frac{\sum_{i=0}^{n-1}(y_{i+1} + y_i)(x_{i+1} - x_i)}{2(x_n - x_0)} \tag{3.71}$$

Une fois les paramètres $\hat{K}$, $\hat{V_0}$ et $\hat{T_p}$ estimés, nous pouvons normaliser l'échantillon P' de $\hat{f}(t)$ par

$$P_i'' = \begin{cases} \left(x_i'' = x_i' - \hat{T_p}, y_i'' = \dfrac{y_i' - \hat{V_0}}{\hat{K}}\right), x_i' \geq \hat{T_p} \\ \left(x_i'' = 1 + x_i' - \hat{T_p}, y_i'' = \dfrac{y_i' - \hat{V_0}}{\hat{K}}\right), x_i' < \hat{T_p} \end{cases} , i = 0, \ldots, n' \tag{3.72}$$

Chapitre 4

Algorithmes d'apprentissage supervisés

L'apprentissage automatique a pour domaine d'étude, le développement, l'analyse et l'implémentation de méthodes permettant à des ordinateurs, et plus généralement à des machines, d'améliorer leur comportement face à un problème à travers des processus d'apprentissage. Il est un savant mélange de statistiques, d'exploration de données et d'intelligence artificielle. Ses applications sont nombreuses, citons par exemple le pilotage autonome de véhicules tel le "DARPA grand Challenge" ou encore le contrôle de processus industriels en passant par le diagnostic médical. Il constitue un domaine d'étude relativement récent et en perpétuelle évolution[24].

Dans le cadre de notre étude de l'excentricité des étoiles BS1, nous nous intéresserons plus particulièrement à un sous-domaine de l'apprentissage automatique qu'est l'apprentissage supervisé où, sur base d'un ensemble d'apprentissage constitué d'entrées-sorties, nous cherchons à déterminer un modèle des entrées nous permettant d'extrapoler au mieux la sortie de nouveaux cas se présentant à nous.

Avant de nous aventurer plus loin, définissons quelques notions nécessaires à la compréhension des différents algorithmes présentés. Comme nous l'avons vu, les algorithmes d'apprentissage supervisés se basent sur un ensemble d'apprentissage, soit LS cet ensemble. Nous savons également que LS est constitué d'un ensemble d'attributs, soit $x = \{x_0, \ldots, x_{p-1}\}$ ses attributs d'entrées et $y = \{y_0, \ldots, y_{q-1}\}$ ses sorties. Dans le cadre de notre étude, nous aurons $q = 1$ et nous représenterons donc les sorties de LS simplement par $y_i, i = 0, \ldots, N-1$. Nous savons également que la sortie est numérique, nous nous trouvons donc dans un problème de régression. Enfin, l'accès aux valeurs d'un attribut i pour une observation o_j, sera représenté par $x_i(o_j)$, nous pouvons dès lors exprimer les vecteurs d'attributs d'entrée comme $x(o_i) = \{x_0(o_i), \ldots, x_{p-1}(o_i)\}, i = 0, \ldots, N-1$. Définissons finalement N comme la taille de l'ensemble d'apprentissage et p comme sa dimension.

Nous pouvons dès lors définir le problème d'apprentissage supervisé comme[24] : sur base d'un ensemble d'apprentissage LS constitué de paires d'entrée-sortie, $LS = \{(x_i, y_i) | i = 0, \ldots, N-1\}$ où $x_i \in \mathbb{R}^p$ et où $y_i \in \mathbb{R}$. Le but de l'apprentissage en régression est alors de trouver une fonction $\hat{f} : \mathbb{R}^p \to \mathbb{R}$ qui minimise l'erreur quadratique en généralisation $\mathbb{E}_{X,Y}(\hat{f}(x) - y)^2$.

Notons finalement que l'erreur sur l'ensemble d'apprentissage est souvent biaisée positivement. On estime dès lors l'erreur d'un modèle sur un ensemble de test indépendant de l'ensemble d'apprentissage : $\sum_{o \in TS} \left(\hat{f}(x(o) - y(o)\right)^2$, où TS est l'ensemble de test indépendant. L'erreur obtenue est appelée l'erreur en généralisation. Le compromis entre l'erreur sur l'ensemble d'apprentissage et l'erreur en généralisation est appelé la régularisation.

4.1 Algorithme des plus proches voisins pour la régression

L'algorithme des plus proches voisins est basé sur l'intuition que des objets avec des entrées similaires doivent avoir des sorties similaires[24]. Cette distance est toute relative par rapport au problème traîté. Si nos entrées sont des chaînes de caractères nous pourrions par exemple prendre la distance de Levenshtein[25]. Nous supposerons actuellement que cette mesure est la distance euclidienne $d(o, o') = \sqrt{(X(o) - X(o'))(X(o) - X(o'))^T}$, où $X(o)$, $X(o')$ sont les vecteurs des valeurs des attributs de o et o'. Une fois cette mesure de distance déterminée, nous pouvons formuler l'algorithme des plus proches voisins comme[24]

$$\hat{Y}(o) = \frac{1}{k} \sum_{o_i \in N_k(o, LS)} y_i \tag{4.1}$$

où y_i est la sortie associée à l'observation o_i et où $N_k(o, LS)$ représente les k plus proches voisins de o dans LS selon la distance précédemment définie. Nous pouvons aisément voir que la paramétrisation de k nous permet d'obtenir une régularisation du problème.

Comme nous l'avons vu section 3.3, les fonctions d'ajustement de courbes sont échantillonnées de manière uniforme, nous nous attendons donc à ce que chaque attribut soit pondéré afin d'obtenir des poids similaires lors du calcul de la distance. Nous supposerons que chaque attribut doit avoir une moyenne nulle et une variance unitaire sur l'ensemble d'apprentissage. Soit les moyennes des vecteurs d'attributs $\overline{x_i}$,

$$\overline{x_i} = \frac{\sum_{j=0}^{N-1} x_i(o_j)}{N}, \quad i = 0, \ldots, p-1 \tag{4.2}$$

et σ_{x_i} les écarts-types de ces vecteurs

$$\sigma_{x_i} = \sqrt{\frac{\sum_{j=0}^{N-1} \left(x_i\left(o_j\right) - \overline{x_i}\right)^2}{N}} \tag{4.3}$$

nous pouvons formuler la mesure de distance pondérée entre deux observations o et o' comme

$$d^w(o, o') = \sqrt{\sum_{i=0}^{p-1} \left(\frac{x_i(o) - \overline{x_i}}{\sigma_{x_i}} - \frac{x_i(o') - \overline{x_i}}{\sigma_{x_i}} \right)^2} \tag{4.4}$$

ou plus simplement

$$d^w(o, o') = \sqrt{\sum_{i=0}^{p-1} \frac{1}{\sigma_{x_i}^2} \left(x_i(o) - x_i(o')\right)^2} \tag{4.5}$$

Afin de gagner en efficacité, nous pouvons également considérer que la solution est une moyenne pondérée de l'inverse des distances

$$\hat{Y}(o) = \frac{\sum_{o_i \in N_k^w(o, LS)} \frac{y_i}{d^w(o, o_i)}}{\sum_{o_i \in N_k^w(o, LS)} \frac{1}{d^w(o, o_i)}} \tag{4.6}$$

Notons finalement que l'entièreté de l'ensemble d'apprentissage doit être stocké en mémoire et que toute nouvelle observation doit être comparée à chaque observation de l'ensemble d'apprentissage. Sa complexité est donc de l'ordre de $\theta(pN)$ pour l'apprentissage et pour la prédiction.

4.2 Algorithmes basés sur les arbres pour la régression

L'idée des arbres de régression est de partitionner l'espace des attributs d'entrée et d'associer chacune de ces partitions à un modèle simple[26]. Nous introduirons ces derniers sur base des arbres binaires à entrées numériques. Plus particulièrement, un arbre binaire est un arbre où

- Chaque nœud interne est composé d'un test d'inégalité basé sur un attribut x_i pour une valeur s_i.
- Chaque branche correspond à un des deux résultats du test du nœud parent.
- Chaque feuille est associée à un modèle simple, dans notre cas nous prendrons la moyenne des sorties de la partition afin de minimiser l'erreur quadratique.

La figure 4.1 illustre le résultat d'un arbre de régression pour des attributs numériques où Ri représente la moyenne des sorties de la partition associée à la feuille i.

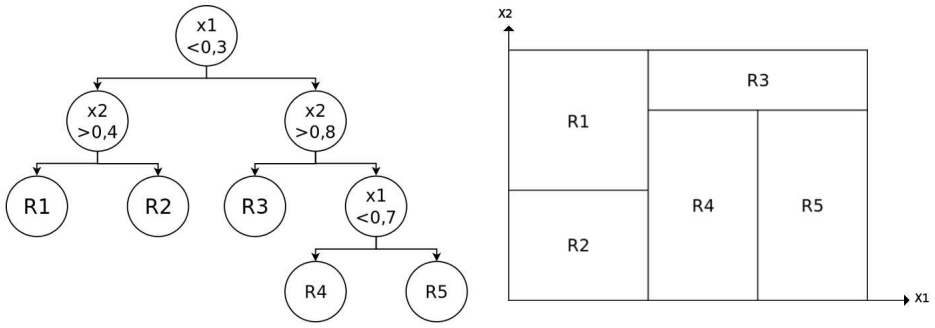

FIG. 4.1 – Illustration d'un arbre de régression binaire pour des attributs numériques

Initialement, notre partition est l'ensemble d'apprentissage complet. Sur base de ce dernier, il nous faut trouver un attribut x_i et une valeur s_i maximisant une mesure de réduction de l'impureté. Pour cela, définissons les deux sous-ensembles d'apprentissage générés par un test de l'attribut x_i et de la valeur s_i comme

$$R_{inf}(E, x_i, s_i) = \{o_j | x_i(o_j) < s_i\}, \quad o_j \in E \tag{4.7}$$

et

$$R_{sup}(E, x_i, s_i) = \{o_j | x_i(o_j) \geq s_i\}, \quad o_j \in E, \tag{4.8}$$

la réduction de l'impureté introduite par cette séparation peut être exprimée comme

$$\Delta I(E, x_i, s_i) = \sigma^2_{y|E} - \frac{|R_{inf}(E, x_i, s_i)|}{|E|}\sigma^2_{y|R_{inf}} - \frac{|R_{sup}(E, x_i, s_i)|}{|E|}\sigma^2_{y|R_{sup}} \tag{4.9}$$

où $|E|$, $|R_{inf}|$ et $|R_{sup}|$ représentent la taille des différents ensembles et où $\sigma^2_{y|E}$, $\sigma^2_{y|R_{inf}}$ et $\sigma^2_{y|R_{sup}}$ représentent la variance de la sortie y au sein de ces ensembles. Nous pouvons donc déterminer x_i et s_i en testant de manière exhaustive chaque attribut et chaque valeur dichotomique au sein de ces attributs. Une fois les ensembles R_{inf} et R_{sup} déterminés, nous pouvons appliquer la méthode de manière récursive jusqu'à ce que $\sigma^2_{y|E} = 0$, ou qu'aucune valeur dichotomique ne puisse être trouvée parmi les attributs, ou encore qu'un critère d'arrêt soit remplit, par exemple $|E| < N_{min}$.

La construction d'un arbre entièrement développé, comme nous pouvons nous en douter, possède un faible biais mais une très haute variance, plus particulièrement dans le cadre d'attributs numériques[24]. Une première solution qui s'offre à nous est de stopper le développement de cet arbre selon un critère de coût-complexité défini, soit l'information fournie par un arbre $\mathcal{T}$

$$\mathcal{I}_{\mathcal{C}}^{\mathcal{T}}(LS) = \sum_{i=1}^{\mathcal{C}^{\mathcal{T}}} \frac{|LS_i|}{|LS|} \Delta I(LS_i, x_i, s_i) \tag{4.10}$$

où $\mathcal{C}^{\mathcal{T}}$ est le nombre de nœuds tests de $\mathcal{T}$, où LS_i est le sous-ensemble d'apprentissage associé au nœud i et où x_i et s_i sont respectivement l'attribut et la valeur de test au nœud i. Nous pouvons dès lors définir la qualité de l'arbre comme

$$Q(\mathcal{T}, LS) = |LS| \mathcal{I}_{\mathcal{C}}^{\mathcal{T}}(LS) - \alpha \mathcal{C}^{\mathcal{T}}, \quad \alpha \geq 0 \tag{4.11}$$

Cette qualité est dès lors évaluée durant le développement de l'arbre où la diminution de la qualité de l'arbre pour un α donné formera le critère d'arrêt. Notons que α est ici un terme de régularisation. En effet $\alpha = 0$ nous donnera un arbre entièrement développé tandis qu'un α élevé nous donnera un arbre élagué de manière précoce.

Sur base d'un arbre entièrement développé, nous pouvons également construire une séquence d'arbres $\left\{\mathcal{T}^q \subset \mathcal{T}^{q-1} \subset \ldots \subset \mathcal{T}^0 \subset \mathcal{T}\right\}$, où $\mathcal{T}^{q-1}$ est obtenu en supprimant un nœud de test de $\mathcal{T}^q$[24]. Ce dernier peut être le nœud qui, lorsqu'il est enlevé, dimininue le plus l'erreur sur un ensemble de test indépendant VS. Nous pouvons également supprimer le nœud qui maximise la qualité de l'arbre sur l'ensemble VS selon la formule 4.11 pour un α croissant. Finalement, nous sélectionnerons l'arbre minimisant l'erreur sur l'ensemble de test VS. Le principal défaut de cette approche est la nécessité d'allouer une partie des données à l'élagage de l'arbre.

Notons finalement que le développement des arbres de régression est un processus rapide. Sa complexité est de l'ordre de $\theta(pN \log N)$ pour le développement d'un arbre complet et de l'ordre du plus long chemin reliant une feuille au nœud racine pour le test.

4.2.1 Les arbres extrêmement randomisés

Comme nous l'avons vu, les arbres de régression possèdent une haute variance. L'élagage y apporte une solution partielle mais insuffisante afin de les rendre compétitifs avec d'autres méthodes d'apprentissage. Nous pouvons également constater que cette variance est fortement influencée par le caractère déterministe dont le point de coupure (selon un attribut x_i et pour une valeur s_i) est déterminé. Ce point de coupure dépend largement de l'ensemble d'apprentissage utilisé[27]. De nombreuses méthodes ont émergées afin de réduire le biais ou la variance. Nous nous intéresserons plus particulièrement aux méthodes d'ensemble pour les arbres. Ces dernières se basent sur des versions modifiées de l'ensemble d'apprentissage et/ou des perturbations introduites dans son développement afin de construire une série d'arbres dont les sorties seront agrégées afin de produire un modèle qui, typiquement, possède une plus faible variance[27].

Ces deux constatations nous permettent d'introduire les arbres extrêmement randomisés. Le principe est de construire M arbres sur base de l'ensemble d'apprentissage et de perturber le développement de ces arbres en introduisant une sélection aléatoire des attributs couplés à une valeur aléatoire du point de scission. Plus précisément, lors de la scission d'un nœud, nous sélectionnerons aléatoirement

K attributs sans remplacement et pour chacun de ses attributs x_i, nous lui attribuerons une valeur de scission aléatoire s_i tel que $x_{imin} \leq s_i \leq x_{imax}$ où x_{imin} et x_{imax} sont respectivement les valeurs minimale et maximale observées au sein de l'attribut x_i. Chacun des points de coupure $(x_i, s_i), i = 0, \ldots, K-1$ est dès lors évalué selon la formule 4.9 et celui possédant la plus haute réduction de l'impureté est sélectionné. Le processus continue de manière récursive pour chaque sous-ensemble avec comme critère d'arrêt $|LS_i| < N_{min}$ où $|LS_i|$ est la taille de l'ensemble d'apprenstissage associé au nœud i.

Si nous nous attardons sur les différents paramètres, nous pouvons voir que K est dépendant du problème à traîter, allant de l'arbre entièrement aléatoire dont la structure ne dépend pas des sorties lorsque $K = 1$ à l'arbre dont seulement la valeur du point de coupure s_i est aléatoire lorsque $K = p$. La paramétrisation de K est particulièrement cruciale lors de la présence d'attributs non pertinents. Le paramètre N_{min} est un terme de régularisation, une valeur de $N_{min} = 1$ nous donnera des arbres complètement développés, avec donc un faible biais et une haute variance tandis que $N_{min} > 1$ nous donnera des arbres de plus petites tailles possédant une plus faible variance, mais un plus grand biais. Enfin, le comportement de l'erreur de prédiction étant une fonction décroissante de M, nous choisirons donc M grand[27].

Finalement, l'algorithme présenté se révèle rapide (de l'orbre de $\theta(N \log N)$) tout en gardant une précision comparable aux autres méthodes d'ensemble appliquées à des arbres[27].

4.3 Algorithme de réseaux de neurones pour la régression

Les réseaux de neurones sont des algorithmes d'apprentissage inspirés du fonctionnement d'un cerveau biologique. Ils sont, tout comme ces derniers, composés d'unités de bases interconnectées appelées neurones artificiels[26]. Nous pouvons représenter un tel neurone par une fonction mathématique qui a m valeurs réelles, associe une sortie $y \in \mathbb{R}$ et qui peut être formulée comme

$$\varphi\left(x(o)\right) = \phi\left(w_0 + \sum_{i=1}^{m} w_i x_{i-1}(o)\right) \tag{4.12}$$

où $x(o) \in \mathbb{R}^m$, où $w = \{w_0, \ldots, w_m\}$ représentent les contributions des différentes entrées au neurone (poids) et où ϕ représente la fonction de sortie. Notons que w_0 permet de fixer le décalage potentiel dans les données entrantes, il est appelé le paramètre de biais. En posant $x'(o) = \{1, x_0(o), \ldots, x_{m-1}(o)\}$, nous pouvons plus simplement représenter un neurone artificiel par

$$\varphi\left(x(o)\right) = \phi\left(x'(o)w^T\right) \tag{4.13}$$

Si nous nous positionnons maintenant dans un problème de régression constitué de m entrées numériques et de N observations, nous cherchons dès lors à minimiser

$$\begin{aligned} R^2(w) &= \sum_{i=0}^{N-1} R_i^2(w) \\ &= \sum_{i=0}^{N-1} \left(y_i - \varphi\left(x(o_i)\right)\right)^2 \end{aligned} \tag{4.14}$$

En supposant que ϕ est une fonction différentiable et que le vecteur w est initialisé, nous pouvons résoudre ce problème au moyen d'une descente de gradient[26]. Le gradient de $R_i^2(w)$ nous est donné par

$$\frac{\partial R_i^2(w)}{\partial w_j} = -2\left(y_i - \varphi\left(x(o_i)\right)\right)\phi'\left(x'(o_i)w^T\right)x'_j(o_i), \quad j = 0, \ldots, m \tag{4.15}$$

où ϕ' est la dérivée de la fonction ϕ. Connaissant le gradient, la mise à jour du vecteur de poids à l'itération r peut être formulée comme

$$w_i^{r+1} = w_i^r - \gamma \sum_{j=0}^{N-1} \frac{\partial R_j^2(w)}{\partial w_i}, \quad i = 0, \ldots, m \tag{4.16}$$

où γ est appelé le taux d'apprentissage. Nous voyons dans cette introduction l'intérêt d'avoir une fonction de sortie différentiable. Imaginons maintenant que les entrées de notre neurone soient elles-mêmes les sorties de K neurones artificiels. Nous introduisons donc une couche supplémentaire de neurones. La formule 4.12 devient dès lors

$$\varphi\left(x(o)\right) = \phi\left(\Phi\left(x(o)\right) w^T\right) \tag{4.17}$$

où $\Phi\left(x(o)\right) = \left\{1, \varphi_0\left(x(o)\right), \ldots, \varphi_{K-1}\left(x(o)\right)\right\}$ et où

$$\varphi_k\left(x(o)\right) = \phi_k\left(w_0^k + \sum_{j=1}^{m} w_j^k x_{j-1}(o)\right) = \phi_k\left(x'(o) w^{kT}\right) \tag{4.18}$$

Son gradient nous est donné par

$$\begin{aligned}
\frac{\partial R_i^2(w)}{\partial w_j} &= -2\left(y_i - \varphi\left(x(o_i)\right)\right)\phi'\left(\Phi\left(x(o_i)\right) w^T\right)\Phi_j\left(x(o_i)\right) && (4.19)\\
&= \delta_i \Phi_j\left(x(o_i)\right), \quad j = 0, \ldots, k && (4.20)\\
\frac{\partial R_i^2(w)}{\partial w_j^k} &= \delta_i w_k \Phi_k'\left(\left(x(o_i)\right)\right) x_j'(o_i) && (4.21)\\
&= \delta_{ij} x_j'(o_i), \quad j = 0, \ldots, m && (4.22)
\end{aligned}$$

Comme nous le voyons, il existe une forte corrélation entre $\frac{\partial R_i^2(w)}{\partial w_j}$ et $\frac{\partial R_i^2(w)}{\partial w_j^k}$. Afin d'exploiter au mieux cette corrélation, nous pouvons maintenir la valeur de sortie de chaque neurone $\Phi_j\left(x(o)\right)$, cette opération est appelée la passe avant. Dans un second temps sur base des $\Phi_j\left(x(o)\right)$ et des entrées, nous pouvons calculer δ_i et δ_{ij} et donc, le gradient des vecteurs de poids. Cette passe est appelée la propagation arrière des erreurs. Les valeurs δ pouvant être vues comme les erreurs commises par une certaine couche par rapport à la couche précédente. Il va sans dire que l'exemple illustré ci-dessus s'applique également à des réseaux de neurones avec un nombre de couches arbitrairement grand.

Dans la pratique, les fonctions de sortie souvent utilisées[26] sont la sigmoïde

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}} \tag{4.23}$$

et la tangente hyperbolique

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.24}$$

ceci s'explique principalement par leur non-linéarité, leur continuité $\mathcal{C}^\infty$, la simplicité de calcul de leurs dérivées et leurs courbes en "S". Nous utiliserons également la fonction identité $\mathrm{id}(x) = x$ afin de pouvoir décrire des hyperplans au sein du réseau. Notons finalement que la sigmoïde et la tangente hyperbolique ont un comportement quasi-linéaire aux alentours de zéro. Nous choisirons donc des poids initiaux aléatoires et proches de zéro, ce qui aura pour effet de rendre le modèle quasi-linéaire au départ de l'algorithme et de le rendre non-linéaire au fur et à mesure de

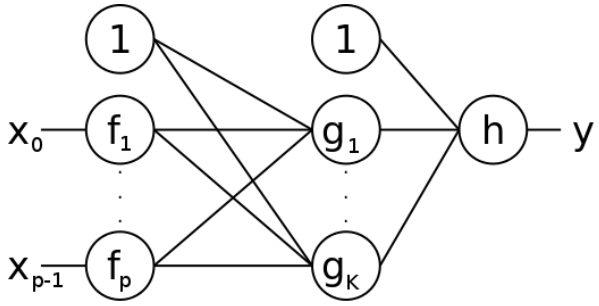

FIG. 4.2 – Réseau de neurones à p entrées, 1 couche cachée et K neurones dans la couche cachée

l'augmentation des poids. Si nous nous intéressons à des structures de réseaux de neurones plus complexes telle que celle représenté figure 4.3, nous nous attendons à obtenir une grande souplesse par rapport au modèle généré. De surcroît, nous pouvons prouver[26] que lorsque $f(x) = \mathrm{id}(x)$, $h(x) = \mathrm{id}(x)$, K arbitrairement grand et que $g(x)$ est choisie de manière appropriée, le modèle peut approximer n'importe quelle fonction continue définie sur $\mathbb{R}^p$. Un tel modèle est appelé un approximateur universel. Nous pouvons constater à l'opposé qu'un modèle avec un K trop petit n'aura pas assez de flexibilité pour captruer les propriétés importantes de la fonction à approximer et que lorsque le modèle est plus complexe que la fonction à approximer, pour un nombre d'itérations de la descente de gradient suffisamment important, le modèle surapprendra les données. Différentes solutions existent afin d'éviter ce surapprentissage, nous pouvons par exemple stopper l'apprentissage de manière précoce lorsque l'erreur du modèle sur un ensemble de données indépendantes de l'ensemble d'apprentissage commence à augmenter[26]. Nous pouvons également modifier la fonction d'erreur $R^2(w)$ de manière à tenir compte des poids, mathématiquement

$$R'^2(w) = R^2(w) + \lambda \|w\|^2 \tag{4.25}$$

où λ est alors un terme de régularisation.

Chapitre 5

Algorithmes génétiques

Les algorithmes génétiques ne sont pas des algorithmes d'apprentissage, ils font partie d'une famille de méthodes parallèles aux algorithmes d'apprentissage : les algorithmes évolutionnaires. À un niveau d'abstraction assez élevé, nous pouvons voir les algorithmes d'apprentissage comme l'imitation des facultés cognitives du cerveau alors que les algorithmes évolutionnaires sont l'imitation de la théorie de l'évolution des espèces[28]. Selon cette théorie, l'évolution tend à produire des organismes adaptés à leur environnement au travers de ses mécanismes. Elle se base pour cela sur plusieurs postulats :

- Les caractéristiques d'un **individu** sont codées dans son ou ses **chromosomes**, eux-mêmes constitués de **gènes**.
- Une **population** d'organismes est constituée d'individus tous différents.
- Les **différences** entre individus leur confèrent une plus ou moins grande **adaptation** à un milieu selon les principes de la **sélection** naturelle.
- Une partie des caractéristiques des individus est **héréditaire**, ces caractéristiques pouvant être altérées soit par **mutation**, soit par dérive génétique.
- Les individus les plus adaptés ont tendance à se **reproduire** davantage.

Nous étudierons plus particulièrement, dans le cadre de ce mémoire, la capacité des algorithmes génétiques à résoudre des problèmes d'optimisation. L'étude de ces derniers ayant commencé très tôt dans l'histoire de l'informatique[28], l'envergure de leur étude dépasse largement le cadre de ce document. Nous ne nous intéresserons donc qu'aux facettes ayant un lien fort avec notre étude. Néanmoins, le lecteur intéressé pourra trouver dans la bibliographie de nombreuses références dans le domaine. Citons plus particulièrement M. MITCHELL, *"An Introduction to Genetic Algorithms"* et R. HAUPT, *"Practical Genetic Algorithms"*.

Afin de passer d'une représentation naturelle de la théorie de l'évolution à une implémentation informatique, il nous faut établir une représentation la plus fidèle possible des processus à l'œuvre lors du passage d'une génération d'individus à la génération suivante. À la vue des différents postulats de cette théorie, nous pouvons représenter sans trop de risque une simplification d'un tel processus par le diagramme illustré figure 5.1[29].

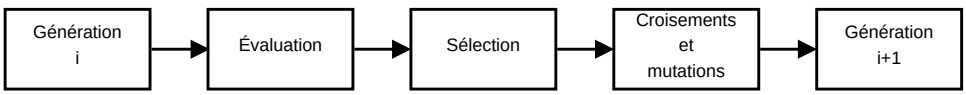

FIG. 5.1 – Diagramme simplifié des processus à l'œuvre selon la théorie de l'évolution lors du passage d'une génération d'individus à la génération suivante

Nous définirons également quelques concepts nécessaires à la compréhension de ce chapitre :

- Un **individu** est la représentation d'une solution candidate au problème d'optimisation, il est constitué d'un chromosome et d'un taux d'adaptation.
- Un **chromosome** est l'ensemble des paramètres à ajuster lors du problème d'optimisation, il est consituté d'un ou plusieurs gènes.
- Un **gène** est une représentation efficace, au sens de l'algorithme génétique, d'un paramètre du problème d'optimisation. Notons que ce dernier correspond la plupart du temps à une représentation binaire du paramètre.
- Une **population** est un ensemble d'individus ayant le même environnement (structure du chromosome identique, fonction d'adaptation identique et problème de base identique) et évoluant de manière synchrone.
- La **fonction d'adaptation** est une fonction quantifiant l'adaptation d'un individu à un environnement. Dans le cadre des problèmes d'optimisation, il s'agira pour la plupart du temps de versions adaptées des fonctions des moindres carrés.

Nous pouvons d'ores et dèjà constater que très peu de paramètres sont dépendants de la fonction à optimiser, les gènes et la fonction d'adaptation pour ne pas les citer. Cette constatation nous permet de faire abstraction des différentes dérivées et restrictions inhérentes à la plupart des méthodes d'optimisation "classiques"[29], la classe des problèmes traîtés se révèle dès lors assez variée et importante.

Si nous nous attardons sur les différentes étapes d'une théorie de l'évolution "informatisée" comme représenté figure 5.1, nous pouvons extraire pour chaque étape des caractéristiques globales qui sont indépendantes de l'implémentation choisie. Parmi lesquelles

- La génération d'une **population initiale** : Nous supposerons que nous ne possédons pas initialement un ensemble de solutions candidates et que ces dernières doivent être générées. De nombreuses techniques existent afin de générer cette polulation initiale, citons par exemple l'algorithme génétique de Hillis[28]. Nous nous contenterons ici de générer une population d'individus dont le chromosome est généré de manière purement aléatoire.
- L'**évaluation** des individus : Chaque individu se voit affecter un taux d'adaptation suivant la fonction d'adaptation.
- La **sélection** des individus : Certains individus sont sélectionnés sur base de leur taux d'adaptation afin de pouvoir se reproduire.
- Le **croisement** : Les individus ayant été sélectionnés pour la reproduction, voient chacun leur chromosome croisé avec celui d'un autre individu sélectionné, afin de produire un ou plusieurs nouveaux individus. Ce croisement étant probabiliste, il se peut dès lors que des individus survivent à une génération.
- La **mutation** : Un certain pourcentage du chromosome de la nouvelle population se voit altéré de manière plus ou moins aléatoire.

Nous pouvons voir que ce processus est stochastique, les algorithmes génétiques sont donc des algorithmes métaheuristiques. Précisons également que les différents concepts présentés dans cette introduction seront analysés plus en profondeur dans la suite de ce document. Cette base nous permet néanmoins d'introduire des concepts plus avancés des algorithmes génétiques.

5.1 Bases théoriques du fonctionnement des algorithmes génétiques

Bien que les algorithmes génétiques soient aisés à décrire et à programmer, leur comportement est complexe et de nombreuses questions à propos de leur fonctionnement restent encore ouvertes à l'heure actuelle. Et ce, malgré l'ensemble des travaux qui ont été réalisés sur leurs fondements théoriques. Nous nous baserons dans ce document sur la théorie traditionnellement admise de ces algorithmes, cette dernière ayant été formulée par John Holland qui est également l'inventeur des algorithmes génétiques[28].

En prenant un certain niveau d'abstraction, cette théorie stipule que les algorithmes génétiques fonctionnent en découvrant, en recombinant et en accentuant l'effet de bons "blocs de constructions" des solutions d'une manière hautement parallèle. Afin de formaliser la notion de "blocs de construction", introduisons celle de schéma. Un schéma est une chaîne de bits composée de 0, de 1 et d'astérisques, cette dernière représentant soit un 0, soit un 1 (ex : $H = 1{*}{*}0 = \{1000, 1010, 1100, 1110\}$). Une chaîne de bits correspondant à un schéma est appelée une instance du schéma. Nous pouvons également définir l'ordre du schéma, $o(H)$, qui est le nombre de bits fixé dans ce dernier et $d(H)$ comme la longueur de définition de H qui est la plus grande distance entre deux bits définis.

Pour toute chaîne de bits donnée de longeur l, il existe 3^l schémas possibles et 2^{2^l} sous-ensembles de bits. Dès lors, les schémas ne peuvent pas être utilisés afin de représenter toutes les populations de chaînes de bits possibles de taille l. Toute chaîne de longeur l est également une instance de 2^l schémas différents, donc toute population de taille n est composée d'un nombre d'instances de schémas différents compris entre 2^l et $n2^l$. Nous pouvons donc voir la phase d'évaluation des individus comme une évaluation implicite de la moyenne du taux d'adaptation d'un nombre important de schémas.

Nous nous intéressons maintenant au nombre d'instances d'un schéma H attendu d'une génération t à la génération suivante $t + 1$. Si nous supposons pour l'instant qu'il n'y a pas de croisements ni de mutations, nous pouvons représenter le taux d'adaptation moyen à un instant t comme

$$\bar{f}(t) = \frac{\sum_{x \in S} f(x)}{n} \tag{5.1}$$

où $f(x)$ est le taux d'adaptation de x, où S est l'ensemble des chaînes de bits de longueur l et où $n = |S|$. Si $m(H, t)$ est le nombre d'instances de H à l'instant t, nous pouvons représenter le taux d'adaptation moyen des instances de H à un instant t comme

$$\hat{u}(H, t) = \frac{\sum_{x \in H} f(x)}{m(H, t)} \tag{5.2}$$

dès lors, nous pouvons calculer le nombre d'instances du schéma H d'une génération à une autre comme

$$E(m(H, t + 1)) = n \frac{\sum_{x \in H} f(x)}{\sum_{x \in S} f(x)} = \frac{\sum_{x \in H} f(x)}{\bar{f}(t)} = \frac{\hat{u}(H, t) m(H, t)}{\bar{f}(t)} \tag{5.3}$$

Les croisements et les mutations peuvent tous les deux créer ou détruire des instances de H. Considérons uniquement leurs aspects destructeurs afin de fixer une borne inférieure sur $E(m(H, t + 1))$. Nous pouvons définir $S_c(H)$ comme la probabilité qu'un croisement en un point unique se produise en dehors de l'intervalle des bits définissant H. Soit p_c, la probabilité de croisement, alors

$$S_c(H) \geq 1 - p_c \left(\frac{d(H)}{l - 1} \right) \tag{5.4}$$

Nous pouvons constater que la probabilité de survie est supérieure pour des schémas courts. Prenons maintenant p_m, la probabilité qu'a chaque bit de la chaîne d'être muté et $S_m(H)$, la probabilité de survie d'une instance de H lors d'une mutation, alors

$$S_m(H) = (1 - p_m)^{o(H)} \tag{5.5}$$

Ici aussi, la probabilité de survie à une mutation est plus élevée pour des schémas de petits ordres. Nous pouvons maintenant intégrer les formules 5.4 et 5.5 à la formule 5.3 afin d'obtenir une borne inférieure sur le nombre d'instances de H entre deux générations[30] :

$$E(m(H, t+1)) \geq \frac{\hat{u}(H,t)}{\bar{f}(t)} m(H,t) \left(1 - p_c \frac{d(H)}{l-1}\right) (1 - p_m)^{o(H)} \tag{5.6}$$

ce résultat est connu comme le théorème des schémas. Il peut être interprété de la manière suivante : si $\hat{u}(H,t) > \bar{f}(t)$ alors il y aura un nombre exponentiel d'instances de ce schéma ayant des longueurs de définition courtes et de petits ordres. Ce théorème nous permet de justifier l'hypothèse des blocs de construction selon laquelle les algorithmes génétiques suivent un modèle qui optimise les schémas de petits ordres et de petites longueurs de définition et par la suite, les recombine afin d'obtenir des schémas de plus grands ordres et de plus grandes longueurs de définition ayant des taux d'adaptation élevés.

La mutation dans le cadre de l'hypothèse des blocs de construction et du théorème du schéma peut être vue comme le moyen d'éviter une perte de diversité pour la position d'un bit donné, ce qui serait par exemple le cas si l'entièreté de la population est une instance de $H = 1 * **$. Nous avons ici exploré le croisement en un point, néanmoins ces principes ont également été démontrés pour l'ensemble des techniques de croisement qui seront explorées ici[28].

5.2 Représentation des gènes

Comme nous l'avons vu, les algorithmes génétiques traîtent les différents paramètres à ajuster lors du problème d'optimisation de manière abstraite au moyen de leur représentation binaire. Les paramètres de la courbe des vitesses radiales prenant leurs valeurs dans $\mathbb{R}$, leur représentation informatique est donc basée sur la norme IEEE 754[31]. Pour rappel, cette dernière définit un bit de signe b_s, une mantisse B_m et un exposant B_e afin de représenter un nombre réel r. Nous pouvons représenter r comme

$$r = (-1)^{b_s} B_m 2^{B_e - 2^{e-1}+1} \tag{5.7}$$

où e est le nombre de bits représentant B_e. Nous pouvons aisément voir que le changement d'un seul bit de B_e peut nous mener à une nouvelle valeur r' extrêmement éloignée de r. Ce dernier argument rend cette représentation inadaptée aux mutations et aux croisements des algorithmes génétiques. Il nous faut dès lors quantifier et borner chaque paramètre x de la fonction étudiée afin que $x' \in \mathbb{Z}_0^+$ et $x' < 2^l$ où l est le nombre de bits que l'on désire utiliser pour la représentation du gène associé au paramètre x. Nous avons donc

$$x' = \left\lfloor \frac{x - x_{min}}{x_{max} - x_{min}} (2^l - 1) \right\rfloor \tag{5.8}$$

où x_{min} et x_{max} sont respectivement les valeurs minimale (comprise) et maximale (non comprise) que x peut prendre dans le cadre du problème étudié.

Cette transformation induit une erreur inhérente qui peut néanmoins être contrebalancée par le choix d'un l adapté au problème[29]. Avec un niveau de sophistication supplémentaire et afin de minimiser l'impact du changement d'un seul bit, nous pouvons travailler avec un codage binaire de Gray[29]. Ce type d'encodage a la particularité de posséder une distance de Hamming de 1 entre deux nombres consécutifs. À titre d'exemple, la représentation du code binaire de Gray pour les nombres de 0 à 7 est illustrée au tableau 5.1. Notons finalement, qu'une transformation inverse est nécessaire afin d'effectuer la phase d'évaluation et que la détermination de la représentation des gènes est une phase cruciale (si ce n'est la plus cruciale) des algorithmes génétiques[29].

Nombre	0	1	2	3	4	5	6	7
Code binaire classique	000	001	010	011	100	101	110	111
Code binaire de Gray	000	001	011	010	110	111	101	100

TAB. 5.1 – Comparaison du code binaire de Gray et du code binaire classique pour les nombres de 0 à 7

5.3 Choix de la fonction d'adaptation et évaluation

Comme nous l'avons vu précédemment, la fonction d'adaptation est utilisée afin de déterminer le taux d'adaptation d'un individu à un environnement. Les individus ayant une forte adaptation à cet environnement se verront affecter un taux d'adaptation haut et inversément, un individu peu adapté se verra affecter un taux d'adaptation bas. La notion d'adaptation étant bien entendu relative au problème traîté. Si nous nous situons dans un problème d'optimisation, et plus particulièrement d'ajustement de courbes selon les points $P = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, nous pouvons formuler la fonction d'adaptation $f_a(\hat{\theta})$ comme

$$f_a(\hat{\theta}) = \frac{1}{\sum_{i=0}^{n} \left(y_i - f(\hat{\theta}, x_i)\right)^2} \tag{5.9}$$

où $\hat{\theta}$ sont les paramètres estimés par l'algorithme génétique et où $f(\theta, x)$ est la fonction cible à ajuster. La phase d'évaluation consistant dès lors à assigner un taux d'adaptation à l'ensemble des individus de la population.

5.4 Sélection des individus et population d'accouplement

Sur base des taux d'adaptation appliqués à la section précédente, et pour une population de taille n, nous pouvons effectuer la sélection de $\lceil n/2 \rceil$ couples d'individus, chaque couple produisant deux nouveaux descendants afin de conserver les schémas présents dans leurs chromosomes. L'ensemble de ces $\lceil n/2 \rceil$ couples est appelé la population d'accouplement. Introduisons également le principe d'élitisme, qui consiste à recopier l'individu (ou les premiers individus) possédant le plus haut taux d'adaptation directement dans la population de la génération suivante. Ce mécanisme nous assure qu'une solution qui serait optimale ne sera pas perdue lors de la phase de croisement et/ou de mutation[29].

Une des premières idées venant à l'esprit afin d'effectuer la sélection de la population d'accouplement est de ne retenir uniquement que les meilleurs individus

et d'en effectuer diverses combinaisons. Néanmoins, nous pouvons prouver[32] que cela mène à une convergence précoce de l'algorithme génétique dûe à une perte de diversité des individus. De nombreuses méthodes stochastiques ont donc vu le jour afin de pallier à ce problème, la suite de ce document présente un certain nombre de techniques de sélection parmi les plus populaires. Définissons d'abord la pression de sélection[32] comme le degré par rapport auquel les individus avec un haut taux d'adaptation sont favorisés. Une haute pression de sélection signifiant un haut favoritisme envers les individus à haut taux d'adaptation. Le choix d'une technique de sélection est basé sur ce critère et est fortement dépendant du problème à traiter.

- **Sélection par roulette :** L'idée de la sélection par roulette est d'assigner une probabilité de sélection directement proportionnelle au taux d'adaptation de l'individu par rapport à la population[32]. Soit $p_s(i)$, la probabilité de sélection de l'individu i pour une population de taille n. Nous pouvons définir cette probabilité de sélection comme

$$p_s(i) = \frac{f_a(\theta_i)}{\sum_{j=0}^{n} f_a(\theta_j)} \tag{5.10}$$

  où $f_a(\theta_i)$ est le taux d'adaptation de l'individu i. Nous pouvons dès lors définir n intervalles associés aux différents individus. Plus particulièrement pour un individu i, nous avons

$$I_i = \left[\sum_{j=0}^{i-1} p_s(j), \quad \sum_{j=0}^{i} p_s(j) \right[\tag{5.11}$$

  La sélection d'une valeur aléatoire dans l'ensemble de ces intervalles nous permet alors de déterminer l'individu sélectionné.
- **Sélection par tournoi et sélection par rang :** Cette technique consiste à sélectionner k individus de manière aléatoire parmi la population et ensuite de sélectionner celle qui possède le taux d'adaptation le plus haut. Lorsque $k = 2$ nous appellerons cette technique la sélection par rang[32].
- **Sélection par échantillonnage déterministe :** Le principe de cette technique est de créer une population de sélection S de taille n égale à celle de la population de base avant d'y tirer aléatoirement l'individu sélectionné. La construction de cette population de sélection est effectuée en deux temps
  1. Évaluer $k_i = \left\lfloor n\frac{f_a(\theta_i)}{F} \right\rfloor$, où $F = \sum_{i=0}^{n} f(\theta_i)$ et ensuite dupliquer k_i fois l'individu i dans S.
  2. Trier l'ensemble de la population sur base de la partie décimale de $k'_i = n\frac{f_a(\theta_i)}{F}$ et recopier les premiers individus afin que $|S| = n$.
- **Sélection par échantillonnage des résidus :** Cette technique est une combinaison de la sélection par échantillonnage déterministe, pour sa première phase et d'une modification de la sélection par roulette pour les parties décimales $k'_i = n\frac{f_a(\theta_i)}{F}$. Cette modification consiste à exclure un individus de la roulette si il est sélectionné, nous parlerons alors de sélection par échantillonnage des résidus sans remplacement. Dans le cas contraire, nous nous trouvons dans le schéma classique de la sélection par roulette et nous parlerons alors de sélection par échantillonnage des résidus avec remplacement.

5.5 Croisements et mutations

Une fois la population d'accouplement construite, nous pouvons en sélectionner les différents couples afin d'effectuer un éventuel croisement. Ce croisement s'effec-

tuera si pour une variable aléatoire $0 \leq x \leq 1$, on a la condition $x \leq p_c$. Dans le cas contraire, les parents subissent uniquement une mutation et sont copiés dans la nouvelle population créée. Ce croisement peut être effectué de différentes manières :

- **Croisements par points :** Dans ce type de croisement des parties des chromosomes sont échangées entre individus selon un ou plusieurs points de coupure. Plus précisément, pour un seul point de coupure dont l'indice est $0 \leq c \leq l$, et en supposant que les chromosomes des deux parents sont de la forme

  $$C_1 = \{b_0^1, b_1^1, \ldots, b_l^1\} \qquad C_2 = \{b_0^2, b_1^2, \ldots, b_l^2\}$$

  nous obtiendrons

  $$C_1 = \{b_0^2, b_1^2, \ldots, b_{c-1}^2, b_c^2, b_{c+1}^1, \ldots, b_l^1\}$$
  $$C_2 = \{b_0^1, b_1^1, \ldots, b_{c-1}^1, b_c^1, b_{c+1}^2, \ldots, b_l^2\}$$

  Connaissant le croisement en un point, il est aisé d'extrapoler le concept de croisement en k points.
- **Croisement uniforme :** Le croisement est ici effectué de manière uniforme sur l'entièreté de la longueur du chromosome. Une manière de procéder est de générer un chromosome aléatoire de longueur l, soit C_u ce chromosome, et d'effectuer les opérations binaires suivantes

  $$C_1' = (\overline{C_u}.C_1) + (C_u.C_2)$$
  $$C_2' = (\overline{C_u}.C_2) + (C_u.C_1)$$

Enfin, comme nous l'avons vu, la mutation permet de garder une certaine diversité au sein de la population. Elle joue également un rôle important lors de la finalisation de l'algorithme. En effet, une fois un schéma fort répandu dans la population, et donc supposé de bonne qualité, la mutation consiste dès lors à affiner les différents chromosomes trouvés. Nous pouvons alors voir les algorithmes génétiques comme ayant une convergence rapide due aux opérations de croisement à leurs débuts, pour ensuite avoir une convergence plus lente où la solution est affinée par les opérations de mutation[33].

Soit un chromosome C' de longueur l, l'opération de mutation sujette à un taux de mutation p_m est définie pour chaque bit de C', b_i comme

$$b_i = \begin{cases} \overline{b_i}, & x \leq p_m \\ b_i, & \text{sinon} \end{cases} \tag{5.12}$$

où x est une variable aléatoire, $0 \leq x \leq 1$.

5.6 Convergence et terminaison

Afin de déterminer le critère d'arrêt d'un algorithme génétique, nous pouvons utiliser plusieurs conditions[29] :

- La solution trouvée est satisfaisante.
- Un nombre d'itérations est atteint.
- Dans le cadre où l'élitisme est utilisé, la constance du chromosome d'élite durant un certain nombre de générations peut constituer un critère d'arrêt.
- Divers tests statistiques peuvent être utilisés, comme par exemple la moyenne et l'écart-type des taux d'adaptation de la population sur un ensemble de générations.

Notons finalement qu'aucune certitude n'existe quant à une solution finale optimale, même locale. Selon l'hypothèse des blocs de constructions et en utilisant l'élitisme, nous sommes seulement sûr de nous rapprocher au fur et à mesure des générations d'un des optima.

Troisième partie

Développements pratiques

Chapitre 6

Exploitation des données d'entrée et module de visualisation

Les données d'entrée sont issues d'un logiciel de simulation créé par l'Agence Spatiale Européenne. Ce dernier permet de générer un ensemble d'observations de courbes de vitesses radiales sur base de différents paramètres : N le nombre d'observations, P la période, V_0 la vitesse du centre de masse (km/s), K la semi-amplitude (km/s), e l'excentricité, ϖ la longitude du périastre et T_p le temps du périastre (jour julien). Le logiciel ajoute également un bruit gaussien aux observations générées, mais ce dernier n'étant pas exactement quantifiable dans la pratique, nous en ferons ici abstraction.

Les données fournies sont composées de 96 000 courbes, chacune étant une combinaison des différents paramètres suivants : $N = \{20, 40\}$, $P = \{1\}$, $V_0 = \{50\}$, $K = \{40, 100\}$, $e = \{0, 0.05, 0.1, \ldots, 0.9, 0.95\}$, $\varpi = \{-150, -120, -90, \ldots, 150, 180\}$ et où T_p prend des valeurs continues dans l'intervalle $[0, 1]$. Nous possédons donc 100 courbes pour chaque combinaison des paramètres (T_p non compris). Notons que 96 000 courbes non bruitées ont également été fournies dans un but didactique. Ces dernières se révélant trop éloignées de leurs homonymes bruités, elles ont fourni des résultats médiocres dans le cadre des algorithmes d'apprentissage. Elles ne seront dès lors pas retenues dans la suite de ce document.

Afin de permettre une exploitation intuitive des données d'entrée, un module de visualisation et de génération des courbes d'ajustement a été réalisé. Ce module implémente l'ensemble des méthodes présentées au chapitre 3 ainsi que les algorithmes génétiques vus au chapitre 5. Il est également responsable de l'échantillonnage et de l'exportation des courbes générées. Précisons que les tests de Fisher-Snedecor utilisés pour l'ajustement par un polynôme et par une série de Fourier ont été utilisés avec un seuil de signification statistique classique de 95%. L'annexe A de ce document présente l'entièreté des courbes pouvant être générées par le module de visualisation sur base de diverses courbes d'entrée.

6.1 Taille de l'échantillon d'exportation

Comme nous l'avons vu section 3.3, les différentes fonctions d'ajustement nécessitent une normalisation afin d'être exploitées au mieux par les algorithmes d'apprentissage. Nous avions alors supposé la taille de l'échantillon à exporter comme connue. Ayant maintenant des bases théoriques suffisantes et un cadre bien défini,

nous pouvons dès lors déterminer la taille de cet échantillon. Pour cela, nous analyserons le comportement de l'algorithme des plus proches voisins pour des échantillons de tailles variées. Notre étude s'intéressera plus particulièrement aux fonctions d'interpolations continues et à l'erreur quadratique de l'excentricité trouvée par l'algorithme.

Le choix de l'algorithme des plus proches voisins peut s'expliquer par le fait qu'un seul paramètre est nécessaire à son utilisation et qu'il n'est pas un processus stochastique. Si plusieurs paramètres étaient nécessaires à l'utilisation de cette technique, ceux-ci auraient dû faire l'objet d'une détermination individuelle. Tandis que l'utilisation d'une technique basée sur des processus stochastiques aurait pu nécessiter plusieurs exécutions afin d'obtenir une solution qui soit acceptable. Ces deux constatations impliquent un coût au niveau du temps d'exécution qui n'est pas envisageable dans le cadre de ce mémoire.

Par ailleurs, le choix d'une méthode d'interpolation continue est basée sur l'intuition qu'une erreur quadratique minimale sur une taille d'échantillon donné traduira la capture des caractéristiques importantes des données d'entrée. La taille de l'échantillon trouvée pouvant alors être généralisée aux autres méthodes d'ajustement de courbe.

Le test a donc été réalisé en comparant les résultats de l'algorithme des plus proches voisins sur base d'échantillons d'interpolations linéaires de différentes tailles. L'ensemble de test est constitué de 3200 courbes et l'ensemble d'apprentissage de 6400 courbes. Précisons également que le nombre optimal de voisins, k, a été déterminé de manière exhaustive pour $k = 1, \ldots, 32$. La figure 6.1 illustre les résultats obtenus pour des données d'entrée où $N = 20$, où $N = 40$ et où les deux sont confondus.

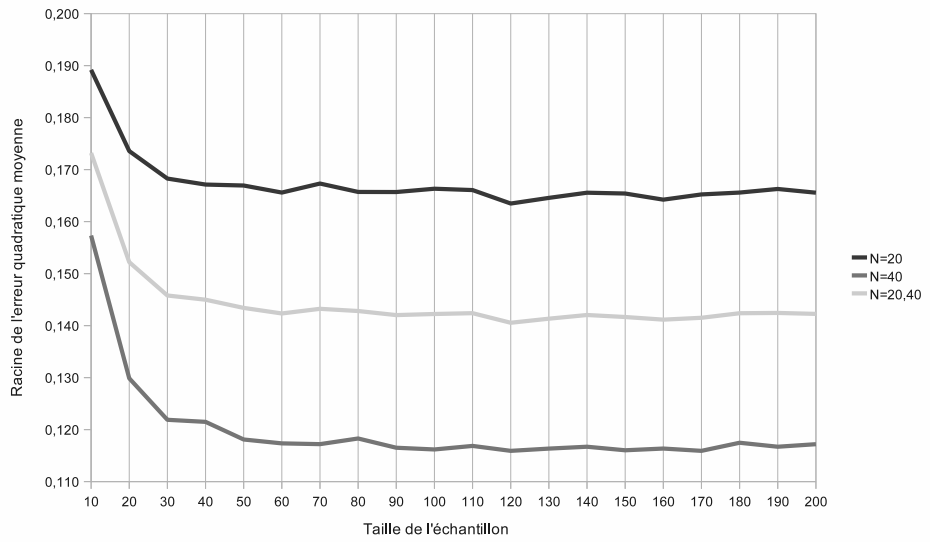

FIG. 6.1 – Résultat de l'exécution de l'algorithme des plus proches voisins sur différentes tailles d'échantillons de l'interpolation linéaire

Nous pouvons constater que l'erreur quadratique se stabilise aux alentours d'une taille d'échantillon de 120 points quel que soit le nombre d'observations intiales. Nous retiendrons dès lors cette valeur afin d'exporter et de normaliser (cf. section 3.3) les courbes générées. Notons finalement que les courbes paramétriques étant de la forme $f : \mathbb{R} \to \mathbb{R}^2$, nous exporterons donc 240 points pour ces dernières.

Chapitre 7

Détermination des paramètres des algorithmes génétiques

Soit la formule 1.5 de vitesse radiale d'une étoile binaire spectroscopique, en supposant que la période soit fixée à 1, nous pouvons décrire cette courbe au moyen de 5 paramètres : e l'excentricité, V_0 la vitesse du centre de masse, K la semi-amplitude, ϖ la longitude du périastre et T_p le temps de passage au périastre.

Désirant tester les algorithmes génétiques au summum de leur précision, les différents gènes associés à chaque paramètre ont été encodés aux limites permises par l'implémentation choisie. Plus précisément chaque gène a été encodé sur 64 bits. Ce choix s'inscrit dans une optique de développement à long terme où la précision requise se verrait bien supérieure à celle exigée dans le cadre de ce mémoire. Notons néanmoins que la précision utilisée est bien supérieure à celle qui ne sera jamais requise pour la détermination des paramètres orbitaux des étoiles spectroscopiques. Le but est de tester les possibilités des algorithmes génétiques à leurs limites.

Afin de borner les différents paramètres selon la formule 5.8, les valeurs minimales et maximales des différents paramètres ont été fixées de la manière suivante :

- $e \in [0, 1[$
- $V_0 \in [0, 100]$
- $K \in [0, 500]$
- $\varpi \in [-180, 180[$
- $T_p \in [0, 1[$

Les valeurs maximales de V_0 et K ayant été fournies par M. Gosset sur base des observations d'objets célestes extrêmes.

Sur base du chapitre 5, nous pouvons isoler six paramètres régissant le comportement des algorithmes génétiques :

- Le type de sélection T_s : sélection par roulette, sélection par rang, sélection par échantillonnage déterministe, sélection par échantillonnage des résidus avec remplacement et sans remplacement
- Le type de croisement T_c : croisement par points ou croisement uniforme
- p_m : la probabilité qu'ont les bits du chromosome d'être mutés
- p_c : la probabilité de croisement
- N : la taille de la population
- k : le nombre d'itération à effectuer

Ces différents paramètres possèdent de surcroît une dépendance non linéaire l'un envers l'autre[28] et sont fortement influencés par la nature du problème à résoudre. Remarquons que seuls les paramètres des algorithmes génétiques seront abordés dans ce chapitre. Les différents paramètres régissant le comportement des algorithmes d'apprentissage sont abordés dans le chapitre 8 et sont traités au cas

par cas.

Ne possédant pas la structure matérielle nécessaire à une exploration exhaustive des paramètres, ces derniers ont été déterminés de manière itérative. Sur base d'un ensemble de paramètres initiaux issus de la littérature[28][29][33] et admis comme fréquemment adaptés à de nombreux problèmes, chacun des paramètres s'est vu attribué une valeur minimisant l'erreur quadratique.

Soit l'ensemble des paramètres initiaux : $T_s =$ sélection par roulette, $T_c =$ croisement en un point unique, $p_m = 0.01, p_c = 0.9, N = 150, k = 75$. Nous cherchons donc à optimiser individuellement chaque paramètre selon la racine de l'erreur quadratique moyenne. Précisons que chaque test a été effectué sur un ensemble de 960 courbes avec utilisation de l'élitisme. Le premier paramètre à avoir été optimisé est le type de sélection. Ce dernier étant supposé avoir l'effet le plus important sur l'ensemble des autres paramètres.

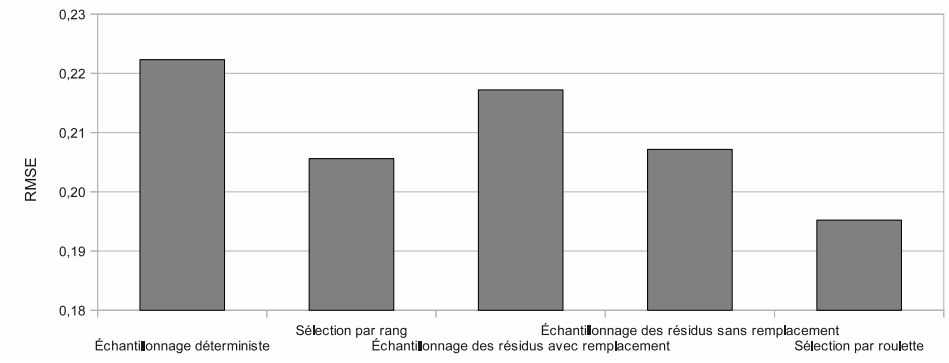

FIG. 7.1 – Détermination du type de sélection optimal des algorithmes génétiques pour un croisement en un point unique, $p_m = 0.01$, $p_c = 0.9$, $N = 150$ et $k = 75$

Nous pouvons voir que le choix de la sélection par roulette était judicieux. Le second paramètre ayant une forte influence est le type de croisement.

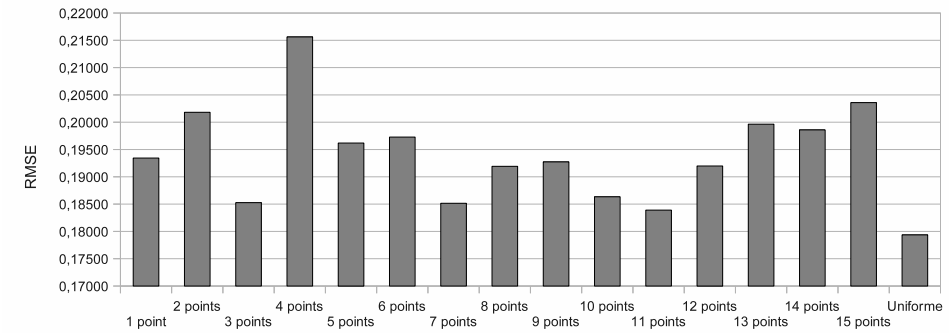

FIG. 7.2 – Détermination du type de croisement optimal des algorithmes génétiques pour une sélection par roulette, $p_m = 0.01$, $p_c = 0.9$, $N = 150$ et $k = 75$

Le croisement uniforme se révélant plus approprié, ce dernier sera dès lors retenu pour la suite des tests. Déterminons maintenant l'ensemble des autres paramètres de manière similaire.

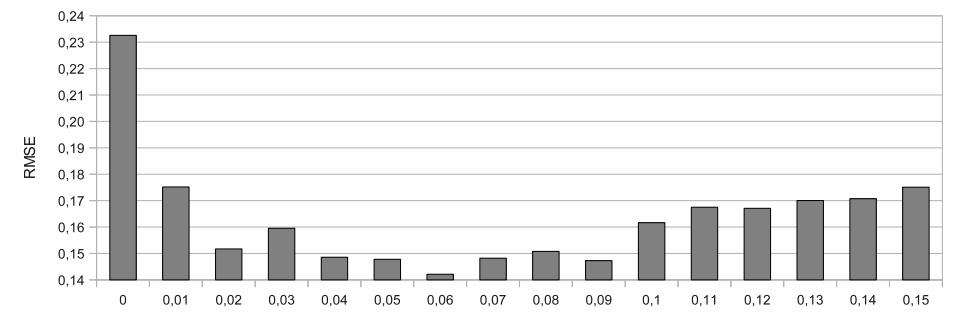

FIG. 7.3 – Détermination du taux de mutation optimal des algorithmes génétiques pour une sélection par roulette, croisement uniforme, $p_c = 0.9$, $N = 150$ et $k = 75$

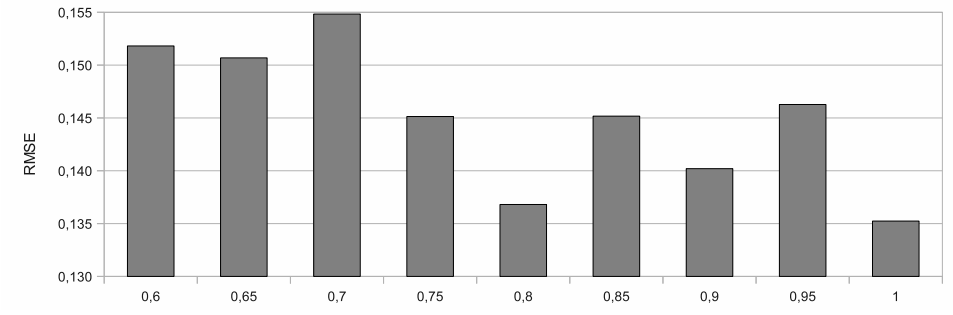

FIG. 7.4 – Détermination du taux de croisement optimal des algorithmes génétiques pour une sélection par roulette, croisement uniforme, $p_m = 0.06$, $N = 150$ et $k = 75$

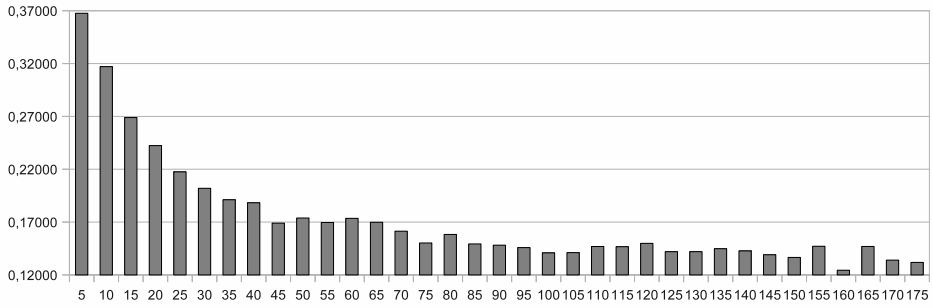

FIG. 7.5 – Détermination de la taille de la population optimale des algorithmes génétiques pour une sélection par roulette, croisement uniforme, $p_m = 0.06$, $p_c = 1$ et $k = 75$

L'erreur étant une fonction décroissante du nombre d'itérations, ce dernier a été déterminé sur base d'une limite de temps d'exécution considérée comme raisonnable (2 secondes sur un processeur cadencé à 2,4Ghz). L'ensemble des paramètres étant alors $T_s =$ sélection par roulette, $T_c =$ croisement uniforme, $p_m = 0.06, p_c = 1, N = 150, k = 150$. Rappelons enfin que rien n'assure que ces paramètres soient optimaux, nous sommes seulement sûrs de nous être rapprochés d'un optima.

Chapitre 8

Résultats

Nous présenterons dans ce chapitre l'ensemble des résultats obtenus au moyen des différentes techniques abordées dans ce document. Il va sans dire que ces résultats sont le fruit de nombreux essais visant à déterminer la complexité optimale des différents algorithmes. Cette notion de complexité idéale est brièvement abordée dans la section 8.2, traitant des algorithmes d'apprentissage. Les résultats des algorithmes génétiques sont quant à eux issus de la détermination des paramètres vue au chapitre 7. Afin de mener à bien ces tests, les algorithmes d'apprentissage ont été utilisés avec des ensembles de test de 24 000 courbes et des ensembles d'apprentissage de 48 000 courbes. Ces résultats sont issus de logiciels originaux, exception faite des réseaux de neurones et des arbres extrêmement randomisés implémentés sur la plate-forme PEPITo®.

L'ensemble des résultats est analysé en fonction de la corrélation de Pearson entre l'excentricité estimée et l'excentricité réelle. Dans le cadre de notre étude, la corrélation de Pearson pour le test de N courbes est donc définie comme

$$r_p = \frac{\sum_{i=1}^{N}(e_i - \overline{e})(\hat{e}_i - \overline{\hat{e}})}{\sqrt{\sum_{i=1}^{N}(e_i - \overline{e})^2}\sqrt{\sum_{i=1}^{N}(\hat{e}_i - \overline{\hat{e}})^2}} \tag{8.1}$$

où e_i est l'excentricité réelle de la courbe i et où $\hat{e}_i$ est l'excentricité estimée de cette courbe. Le choix de la corrélation de Pearson au détriment d'autres techniques d'évaluation s'explique par sa facilité d'interprétation.

Au terme de la présentation des résultats, nous analyserons plus en profondeur le comportement de l'excentricité évaluée par les techniques suscitant un intérêt tout particulier. Cette analyse mettra en évidence l'influence des différents paramètres des courbes de vitesses radiales sur l'évaluation de l'excentricité.

8.1 Algorithmes génétiques

Comme nous l'avons vu au chapitre 5, les algorithmes génétiques déterminent l'entièreté des paramètres des courbes des vitesses radiales afin d'en effectuer les ajustements. Rappelons que différentes combinaisons de paramètres peuvent mener à des courbes identiques. Un des exemples les plus simples se produit lorsque $e = 0$, alors la valeur de ϖ a pour effet de décaler la courbe le long de l'axe des abscisses, tout comme T_p. Une analyse des différents résultats se révèle donc être un processus complexe et ne sera pas entrepris dans cette section. Le test a été mené sur un ensemble de 24 000 courbes, donnant une corrélation de Pearson de 0.902895.

8.2 Algorithmes d'apprentissage

8.2.1 Algorithme des plus proches voisins pour la régression

Pour l'algorithme des plus proches voisins, la détermination du nombre de voisins a été effectuée de manière exhaustive pour $k = 1, \ldots, 32$. Précisons également que le logiciel implémentant cet algorithme a été réécrit en langage C afin d'optimiser son temps d'exécution. Chacune des courbes d'ajustement s'est vue attribuer un nombre de voisins oscillant entre 10 et 18, exception faite des splines cubiques périodiques où le nombre idéal de voisins a été fixé à 27.

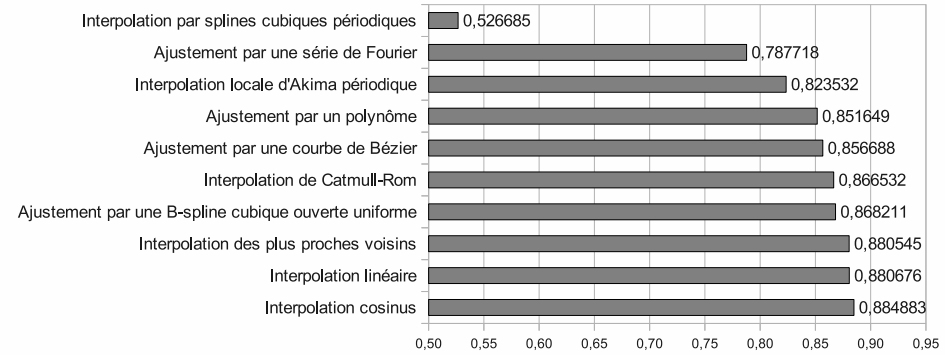

FIG. 8.1 – Résultats de l'algorithme des plus proches voisins pour l'ensemble des techniques d'ajustement de courbes

Nous pouvons constater à la vue de ces résultats que l'algorithme des plus proches voisins malgré sa simplicité se révèle être un bon approximateur pour la plupart des courbes d'ajustement. Notons également que les splines cubiques périodiques semblent être un très mauvais choix de fonction d'ajustement de courbes. Nous nous attendons donc à des résultats similaires pour l'ensemble des algorithmes d'apprentissage les utilisant. À l'opposé, les méthodes d'interpolation basiques semblent fournir les meilleurs résultats dans le cadre de cet algorithme.

8.2.2 Arbres extrêmement randomisés

Nous avons déjà entrevu la régularisation des arbres extrêmement randomisés lors de la section 4.2.1, pour rappel ces derniers sont bâtis sur base de 3 paramètres, M le nombre d'arbres à générer, K le nombre d'attributs sélectionnés aléatoirement et N_{min} la limite de taille du sous-ensemble en dessous duquel stopper l'algorithme. L'erreur étant ici une fonction décroissante du nombre d'arbres, ce dernier doit donc être suffisament élevé. Après test, une valeur de $M = 70$ s'est révélée suffisante pour l'ensemble des courbes, tandis que $N_{min} \approx 5$ et $K = 10$.

Un K égal pour l'ensemble des courbes générées peut s'expliquer par la pertinence de l'entièreté des attributs issus de l'échantillonnage. La qualité du modèle généré dépend ici principalement de M et de N_{min} pour le contrôle de la complexité. Nous pouvons constater que les arbres extrêmement randomisés donnent globalement une meilleure approximation que l'algorithme des plus proches voisins pour une phase d'apprentissage se révélant la plus courte pour l'ensemble des techniques testées. Notons également que les différentes techniques d'ajustement de courbes se voient réparties, selon leurs résultats, de manière quasi similaire en comparaison avec l'algorithme des plus proches voisins. Cela nous permet de nous faire une idée

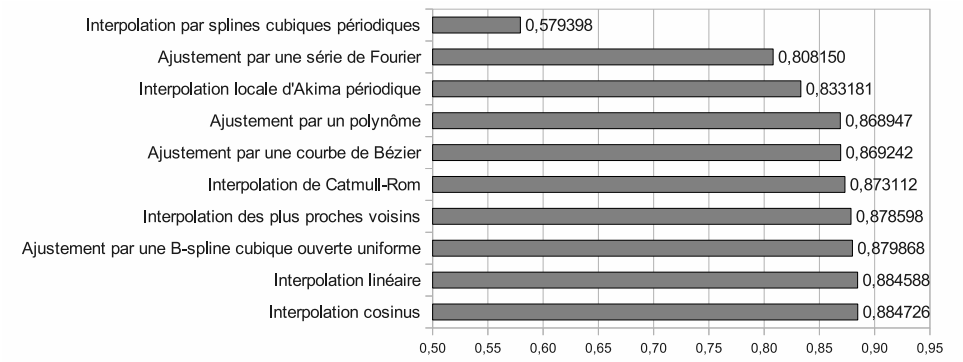

FIG. 8.2 – Résultats des arbres extrêmement randomisés sur l'ensemble des techniques d'ajustement de courbes

quant à la qualité de l'ajustement des différentes techniques utilisées dans le cadre des algorithmes d'apprentissage.

8.2.3 Algorithme de réseaux de neurones pour la régression

Afin d'obtenir une structure de réseaux de neurones suffisamment complexe pour capturer les propriétés importantes des données d'entrée, le nombre de neurones dans les couches cachées a été fixé, après tests, à 25. De même, le nombre de couches cachées a été fixé à 2, permettant une phase d'apprentissage plus aisée. Nous obtenons donc au final un réseau de 2 couches cachées, dont les couches extérieures ont pour fonction d'activation, l'identité, et les couches cachées, la sigmoïde. La régularisation étant dès lors éffectuée au cas par cas selon le nombre de cycles de la descente de gradient à effectuer.

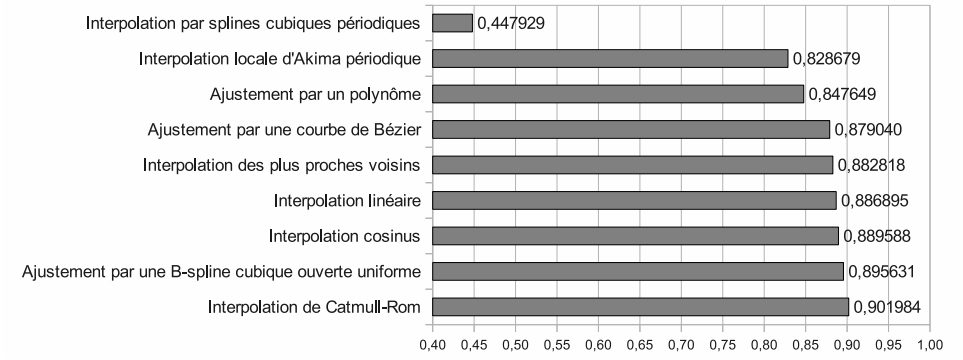

FIG. 8.3 – Résultats de l'algorithme des réseaux de neurones sur l'ensemble des techniques d'ajustement

Nous pouvons nous étonner de l'absence de l'ajustement de courbes par une série de Fourier. Les résultats de ces dernières étant en effet inaccessibles à cause d'une erreur de programmation dans le logiciel utilisé (PEPITo®). La cause de cette erreur n'a pu être déterminée, et ce malgré le nombre d'essais effectués sous différents environnements, avec différents ensembles d'apprentissage, différents ensembles de test et des valeurs de paramètres variés.

Nonobstant cet accroc, nous pouvons constater que les résultats fournis par

cette technique se révèlent être les plus précis parmi l'ensemble des algorithmes d'apprentissage. Nous retiendrons plus particulièrement l'interpolation de Catmull-Rom afin de mener une analyse plus poussée vis-à-vis des algorithmes génétiques.

8.3 Influence des paramètres des données d'entrée sur l'évaluation de l'excentricité

Nous nous attarderons dans cette section à explorer l'influence des paramètres d'entrée sur l'évaluation de l'excentricité par les méthodes ayant donné les meilleurs résultats. Nous analyserons plus particulièrement le comportement de l'algorithme des réseaux de neurones basé sur une interpolation de Catmull-Rom et les algorithmes génétiques. Nous pouvons d'ores et déjà préciser que certains paramètres tels V_0, T_p et ϖ ne seront pas couverts car influençant très peu l'évaluation de l'excentricité.

Notre première analyse portera sur l'influence de la semi-amplitude, K.

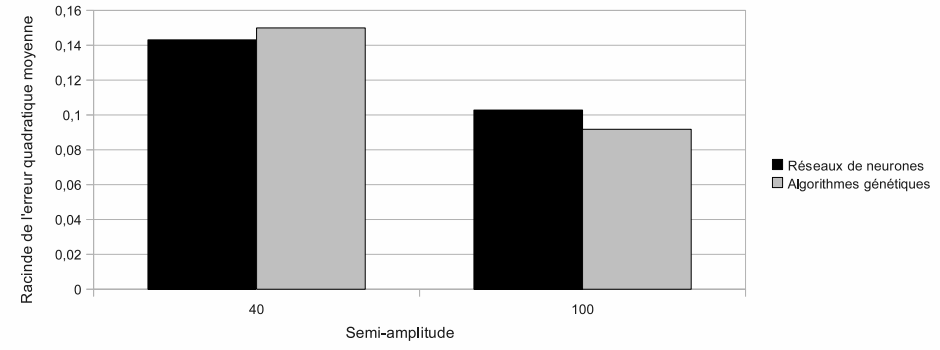

FIG. 8.4 – Influence de la semi-aplitude sur l'évaluation de l'excentricité par les algorithmes génétiques et par les réseaux de neurones basés sur l'interpolation de Catmull-Rom

Nous pouvons constater que la semi-amplitude a une forte influence sur l'approximation de l'excentricité. Ceci peut s'expliquer par la prédominance du bruit des observations lorsque la semi-amplitude est faible. Analysons maintenant le comportement de ces deux méthodes par rapport à l'excentricité réelle (figure 8.5). Nous pouvons voir que les algorithmes génétiques sont très performants pour des excentricités inférieures à 0.8. Au délà de cette limite, les réseaux de neurones basés sur l'interpolation de Catmull-Rom se révélent plus stables et plus performants. Ceci peut intuitivement s'expliquer par le fait que l'algorithme génétique n'a pas connaissance de l'excentricité réelle de la courbe. L'absence d'un ensemble de points importants dans la description de la courbe entraînant dès lors une mauvaise évaluation de l'excentricité. Les réseaux de neurones quant à eux, ont connaissance de cette information et vont tenter de mettre à profit cette connaissance au moyen de leur phase d'apprentissage.

Un dernier paramètre influançant l'évaluation de l'excentricité est le nombre d'observations effectuées. Comme nous pouvons nous en douter, l'erreur sur l'évaluation de l'excentricité est une fonction décroissante de la taille des observations. Si nous reprenons la figure 8.5 et que nous la scindons selon N, le nombre d'observations effectuées, nous pouvons plus particulièrement constater (cf. figure 8.6) que les algorithmes génétiques tirent fortement profit de l'augmentation du nombre

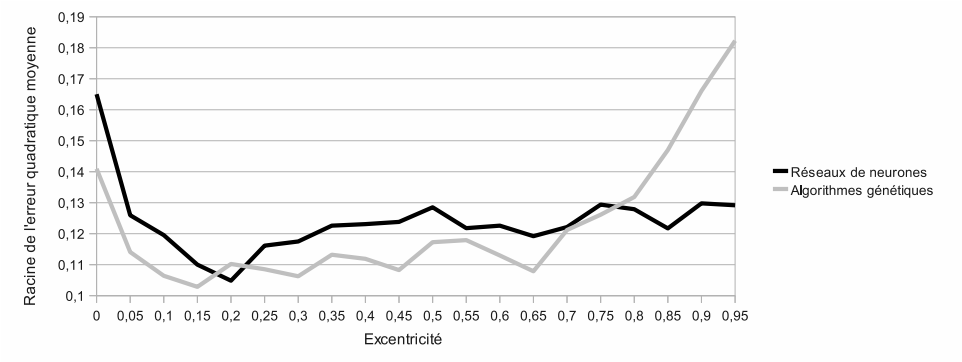

FIG. 8.5 – Comparaison de l'excentricité réelle sur son approximation par les algorithmes génétiques et par les réseaux de neurones basés sur l'interpolation de Catmull-Rom

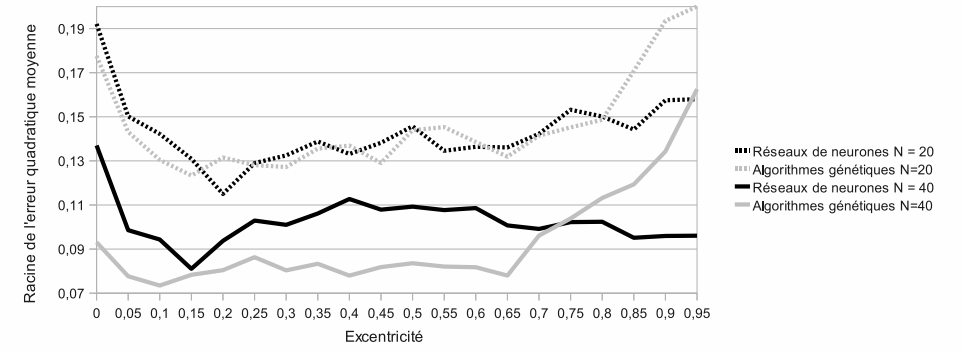

FIG. 8.6 – Influence du nombre d'observations sur l'évaluation de l'excentricité par les algorithmes génétiques et par les réseaux de neurones basés sur l'interpolation de Catmull-Rom

d'observations. Cependant, cette augmentation des observations ne diminue en rien leur difficulté à approximer des courbes de haute excentricité.

Nous pouvons également constater que les deux méthodes peines à obtenir une évaluation correcte pour des excentricités trop faibles. L'évaluation de l'excentricité se faisant alors légèrement au-dessus de l'excentricité réelle.

Nous pouvons finalement constater que chacune des deux méthodes possède ses avantages et ses inconvénients. Les algorithmes génétiques étant plus adaptés à des orbites de relativement faible excentricité ($e < 0.75$) tandis que les réseaux de neurones basés sur l'interpolation de Catmull-Rom se révèlent plus stables lors des hautes excentricités.

Conclusion

L'utilisation des algorithmes génétiques et des algorithmes d'apprentissage dans le cadre de l'évaluation de l'excentricité des étoiles binaires spectroscopiques s'est révélée fructueuse. Sur base des algorithmes génétiques, nous sommes parvenus à ajuster les courbes de vitesses radiales avec une erreur quadratique minime tandis que sur base des réseaux de neurones utilisés conjointement avec une interpolation de Catmull-Rom, nous sommes parvenus à obtenir une erreur quadratique stable pour des orbites de hautes excentricités.

Dans le cadre de développements futurs, ces techniques pourraient faire l'objet de diverses améliorations. Nous pouvons citer entre autres les réseaux de neurones cycliques permettant d'ajuster des modèles de plus grandes complexités, les algorithmes génétiques parallèles diminuant le temps d'éxécution et/ou diminuant la variance des courbes générées, les algorithmes génétiques continus ou encore la détermination des poids des réseaux de neurones sur base d'un algorithme génétique. Ces différentes techniques peuvent également être adaptées afin de déterminer les paramètres orbitaux des étoiles spectroscopiques multiples dont deux ou trois composantes sont visibles. Notons finalement que les algorithmes génétiques devront dans un premier temps faire l'objet d'une adaptation de la taille de chacun des gènes et de la détermination des paramètres idéaux correspondants à cette nouvelle taille. Cela permettra de diminuer le temps d'exécution et d'améliorer potentiellement les résultats.

Nous conclurons ce document par une brève analyse quantitative du travail réalisé. En effet, le module de visualisation et d'exportation des données d'entrée représente à lui seul une grande majorité du travail. Il est constitué de 71 classes Java représentées à l'aide de 500 méthodes accumulant 5500 lignes de code. De manière plus modeste, l'implémentation de l'algorithme des plus proches voisins réprésente quant à lui 750 lignes de code en langage C.

Quatrième partie

Annexes

Annexe A

Courbes d'ajustement générées par le module de visualisation

Nous présenterons dans cette annexe un ensemble de courbes d'ajustement générées à l'aide du module de visualisation et d'exportation des données. Le but étant de fournir au lecteur une vision intuitive des capacités d'approximation des différentes méthodes vues au chapitre 3 et au chapitre 5. Chacune des onze techniques présentées s'est vue confrontée à trois ensembles de données d'entrée de complexité grandissante, tous trois issus du module de simulation des données d'entrée de l'Agence Spatiale Européenne. Plus précisemment, les paramètres des courbes de vitesses radiales d'entrée (cf. chapitre 6) se sont vues affectés de la manière suivante :

1. $N = 40, \varpi = 0^o, e = 0, K = 100km/s$
2. $N = 40, \varpi = 90^o, e = 0.5, K = 40km/s$
3. $N = 20, \varpi = 90^o, e = 0.9, K = 40km/s$

Afin de permettre une comparaison objective, chacune des courbes présentées sera également accompagnée des différentes observations effectuées et de la courbe de vitesse radiale réelle associée (tous deux représentés en gris clair). Précisons finalement que le test de Fisher-Snedecor utilisé lors de l'ajustement par un polynôme et par une série de Fourier possède un seuil de signification statistique de 95% et que l'algorithme génétique utilisé possède les paramètres suivants :

- Taille de la population : 75
- Nombre d'itérations : 75
- Taux de croisement : 100%
- Taux de mutation : 6%
- Type de sélection : Roulette
- Fonction d'adaptation : Somme des erreurs au carré
- Type de croisement : Uniforme

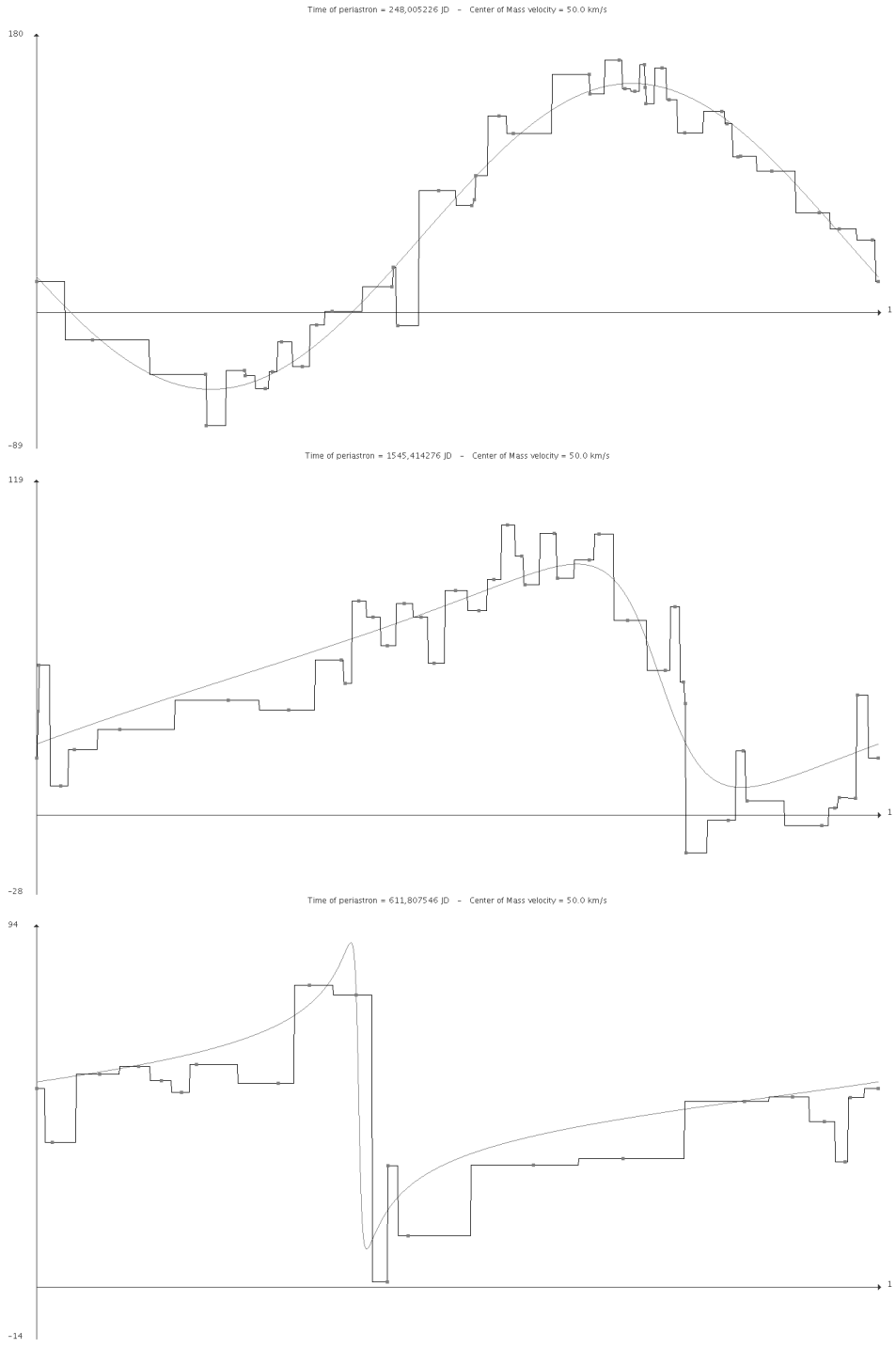

FIG. A.1 – Représentation des courbes de vitesses radiales, des observations liées et des interpolations des plus proches voisins de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

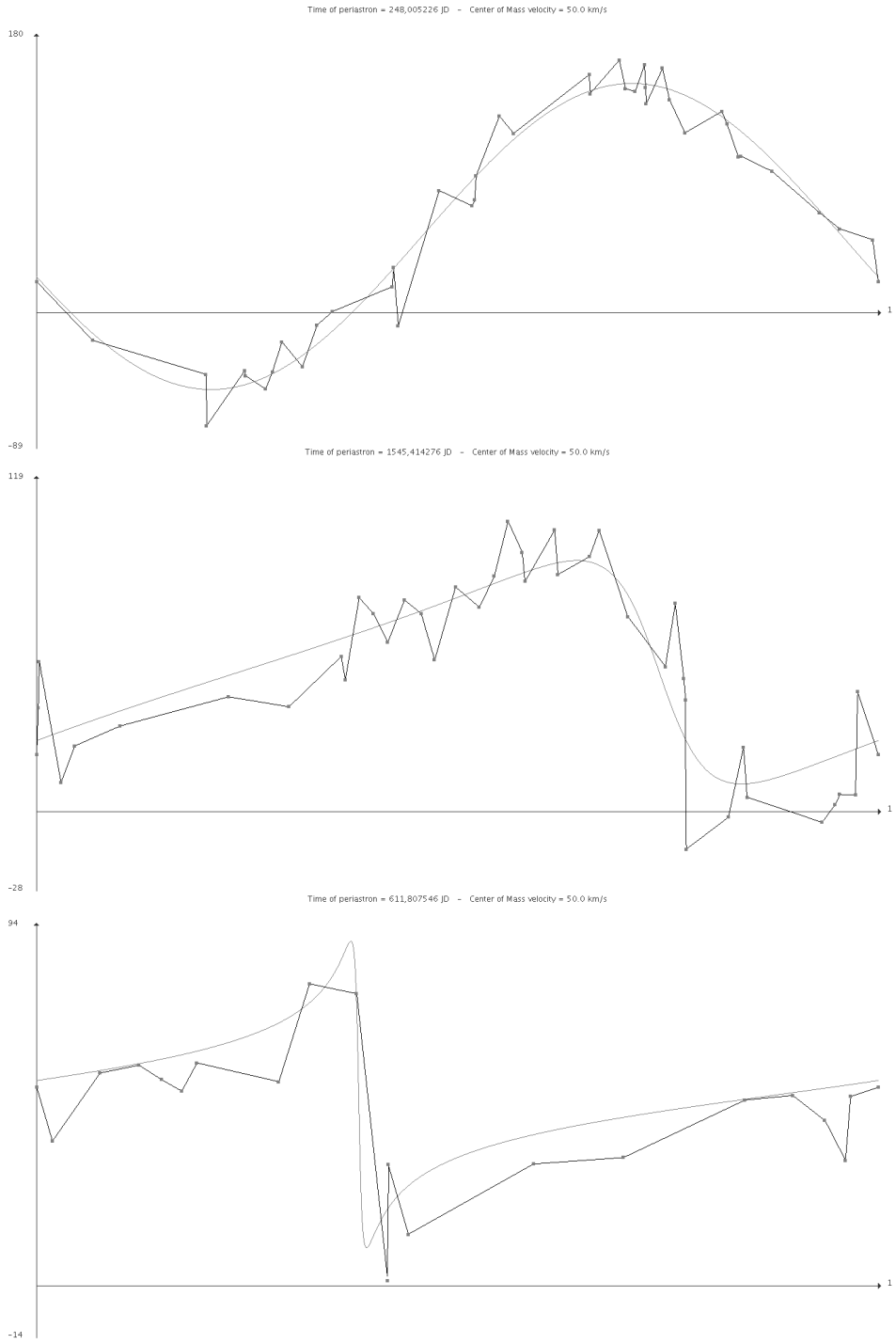

FIG. A.2 – Représentation des courbes de vitesses radiales, des observations liées et de l'interpolation linéaire de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

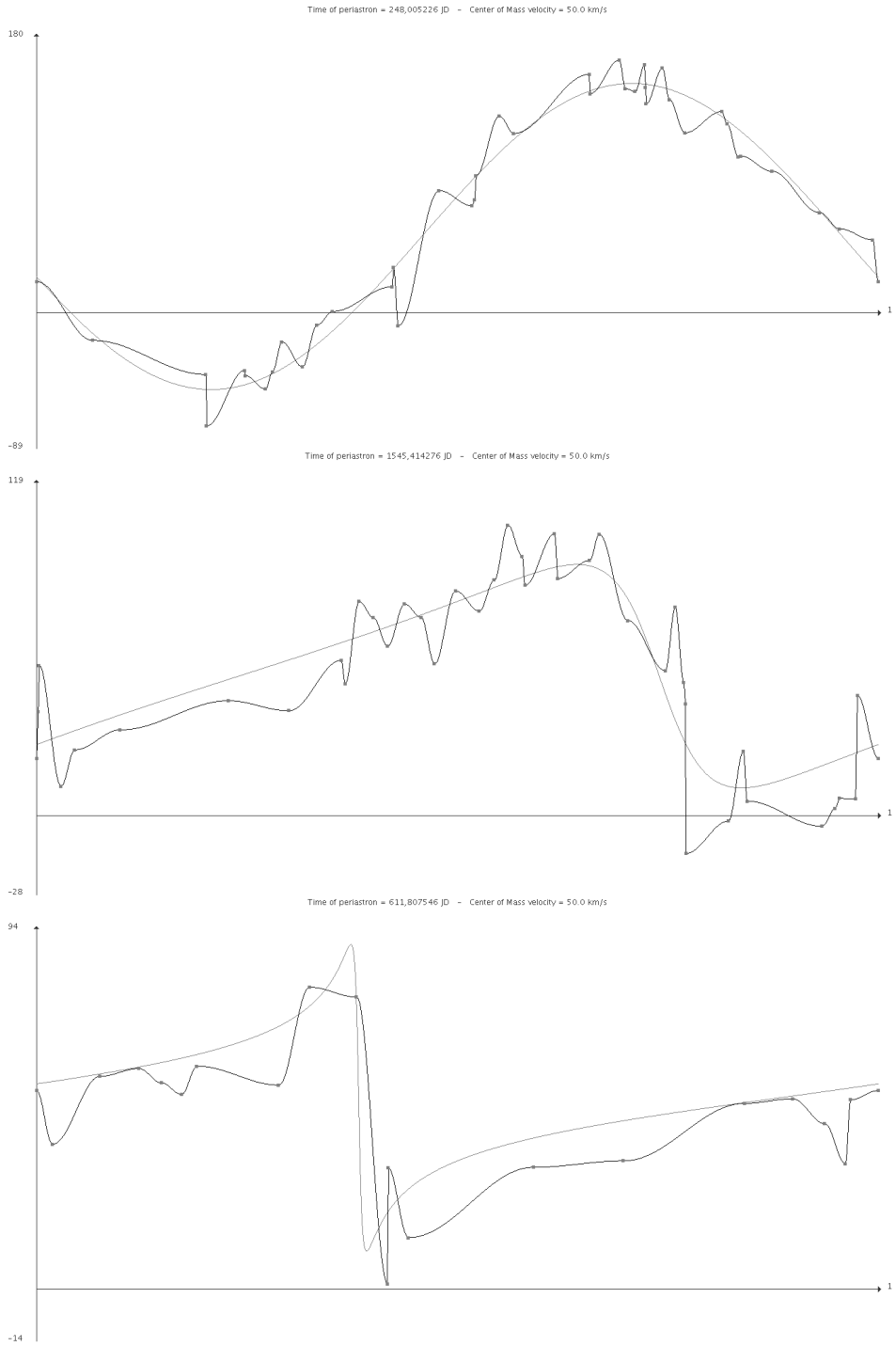

FIG. A.3 – Représentation des courbes de vitesses radiales, des observations liées et de l'interpolation cosinus de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

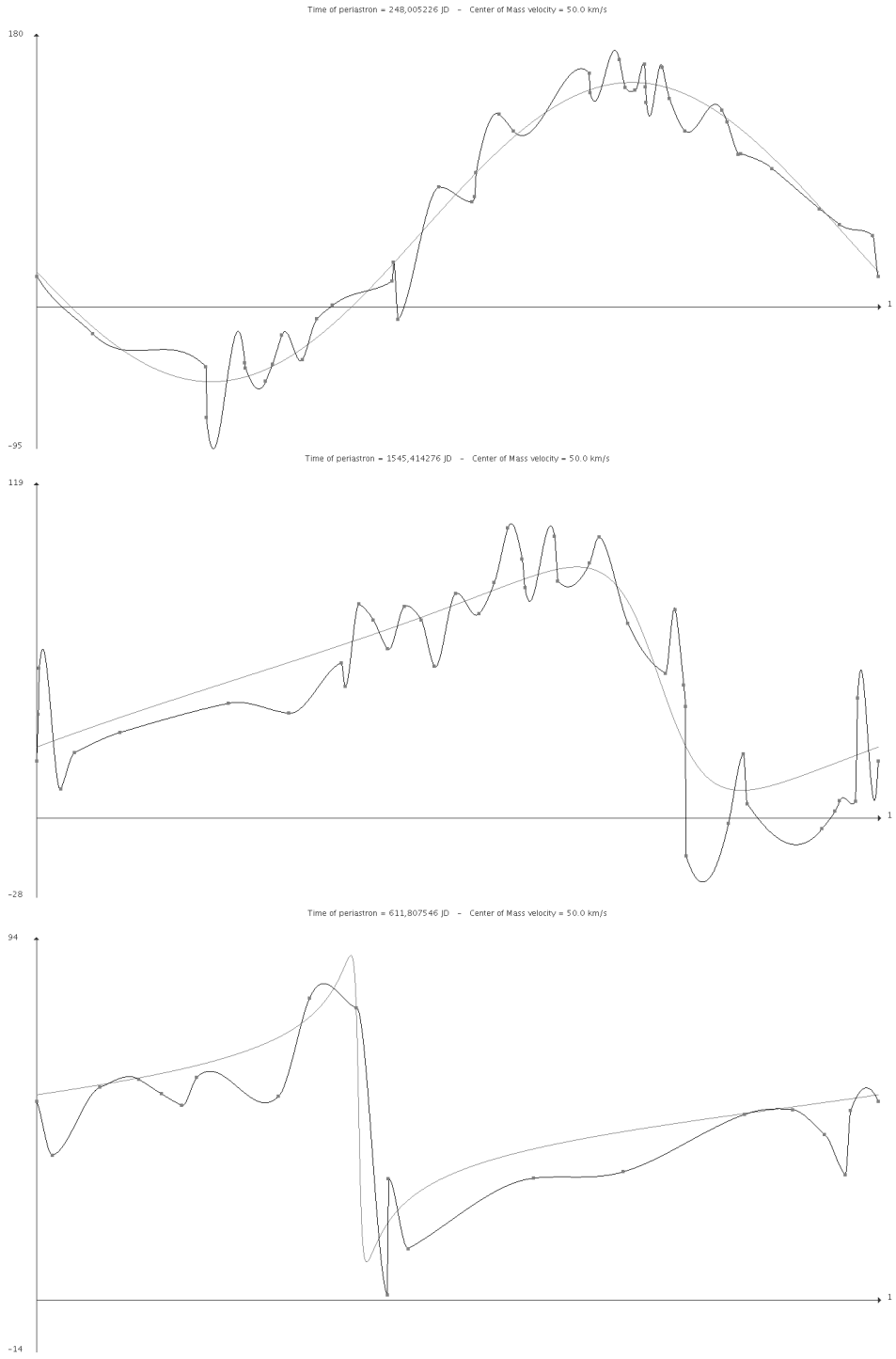

FIG. A.4 – Représentation des courbes de vitesses radiales, des observations liées et de l'interpolation locale d'Akima périodique de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

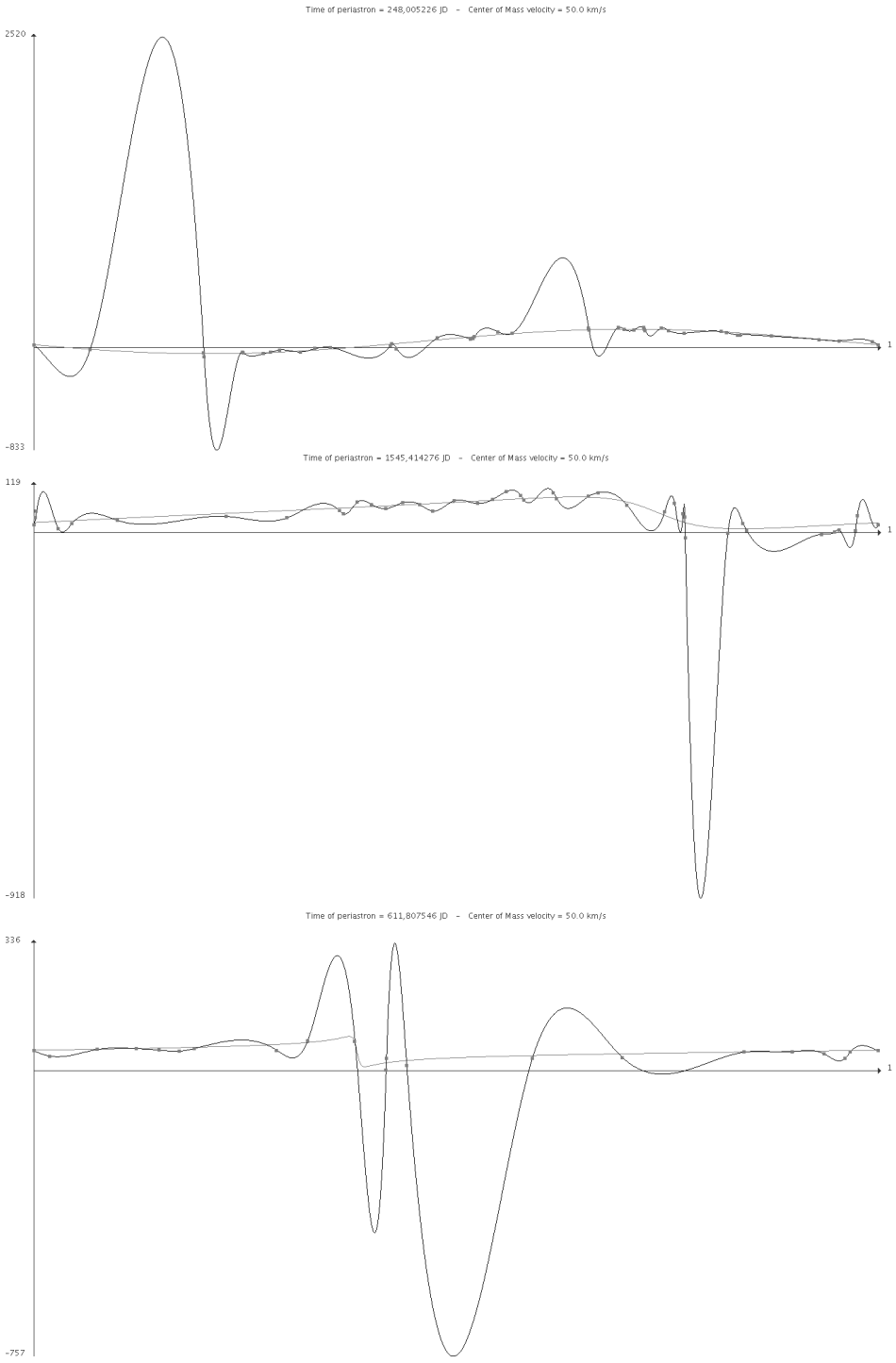

FIG. A.5 – Représentation des courbes de vitesses radiales, des observations liées et de l'interpolation par splines cubiques périodiques de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

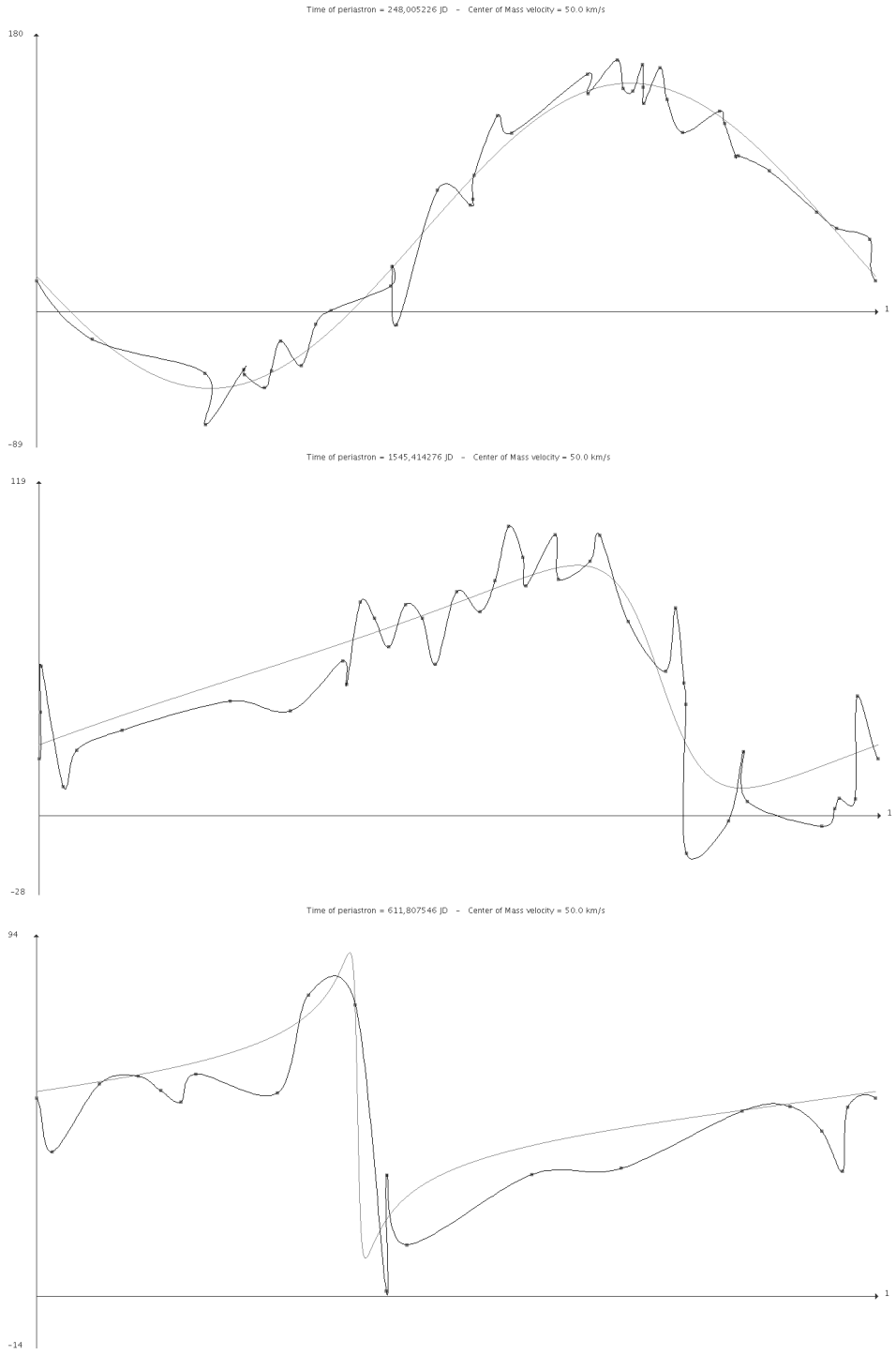

FIG. A.6 – Représentation des courbes de vitesses radiales, des observations liées et de l'interpolation par splines de Catmull-Rom périodiques de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

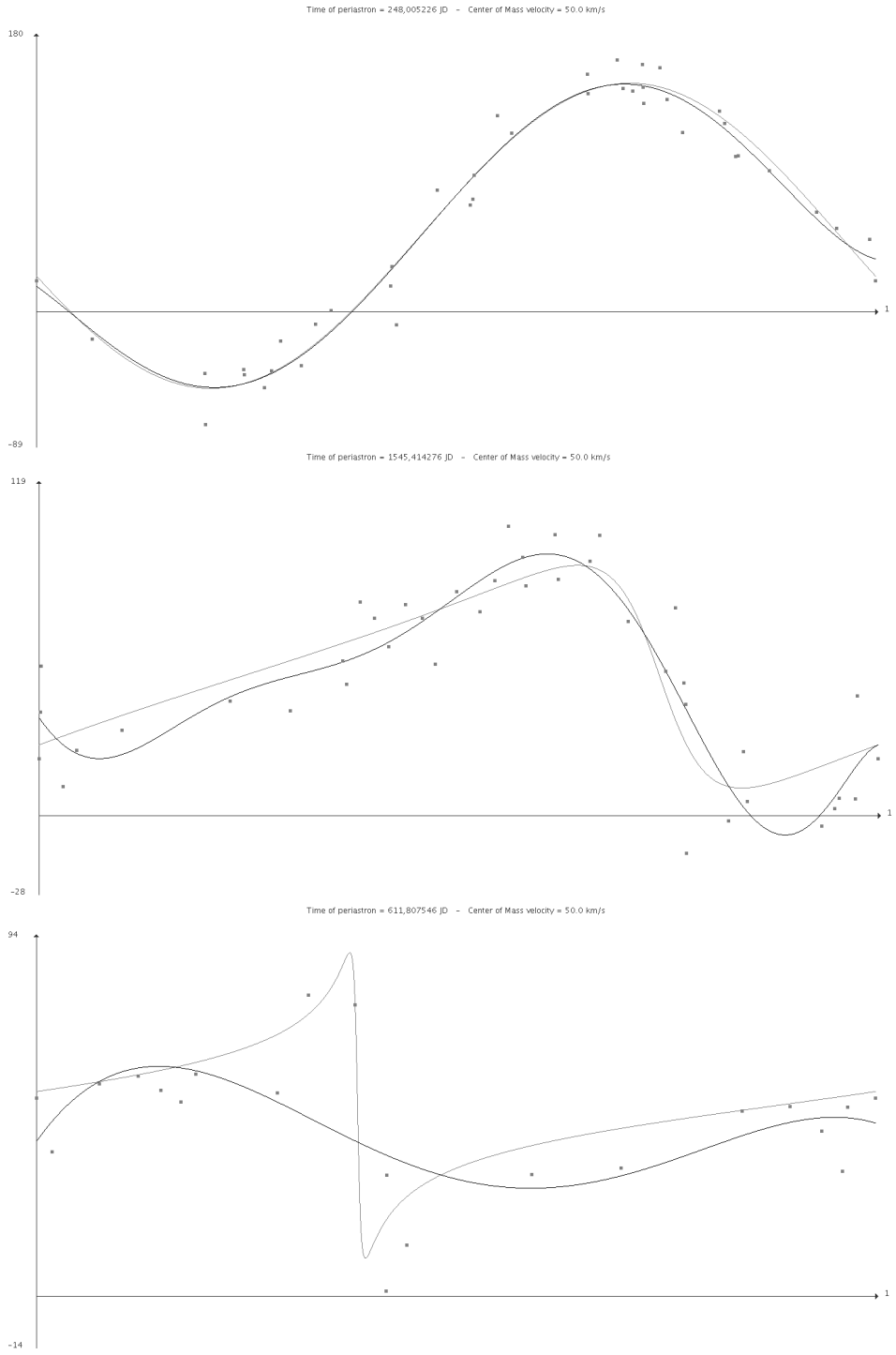

FIG. A.7 – Représentation des courbes de vitesses radiales, des observations liées et de l'ajustement par un polynôme de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

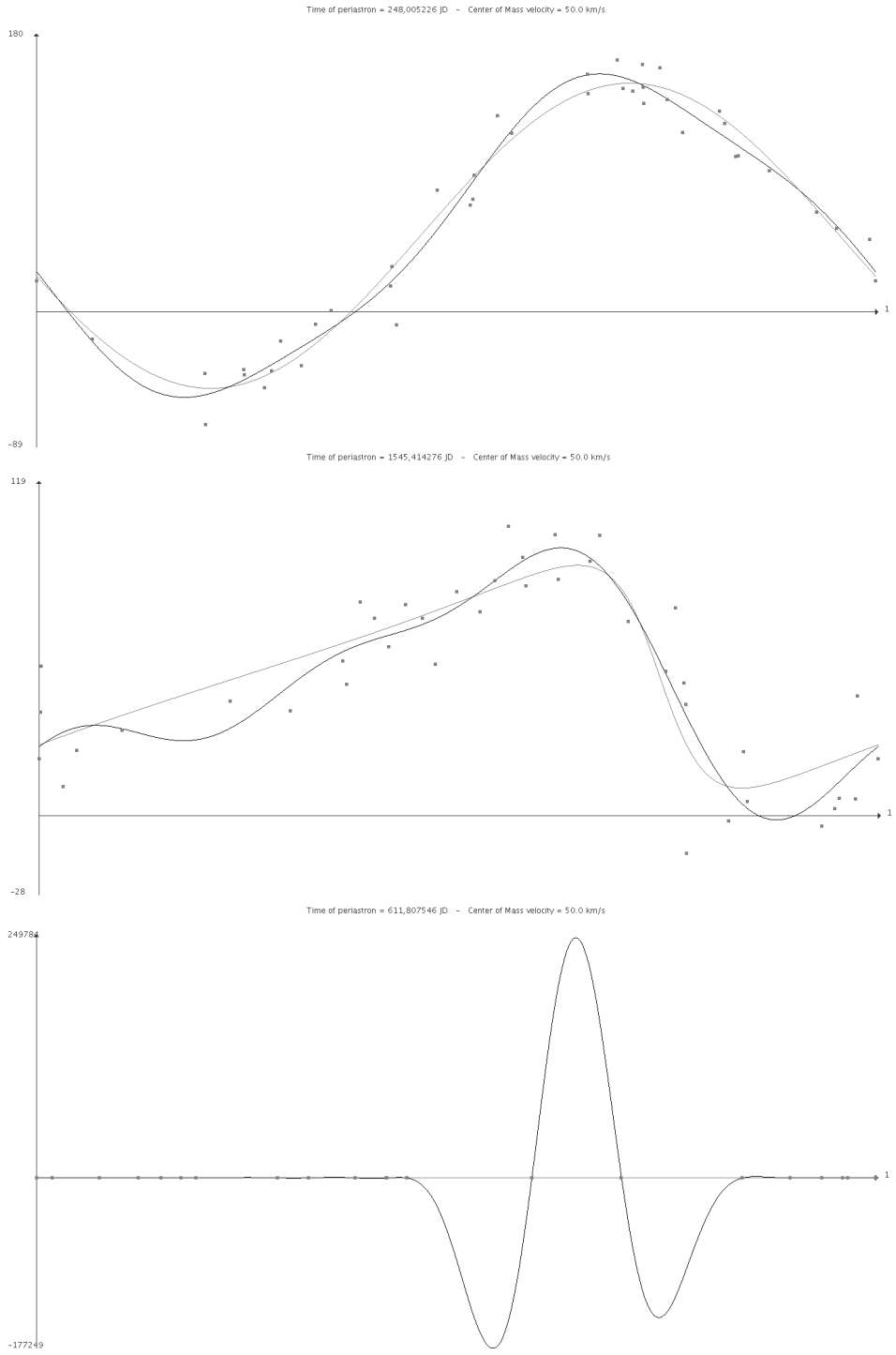

FIG. A.8 – Représentation des courbes de vitesses radiales, des observations liées et de l'ajustement par séries de Fourier de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

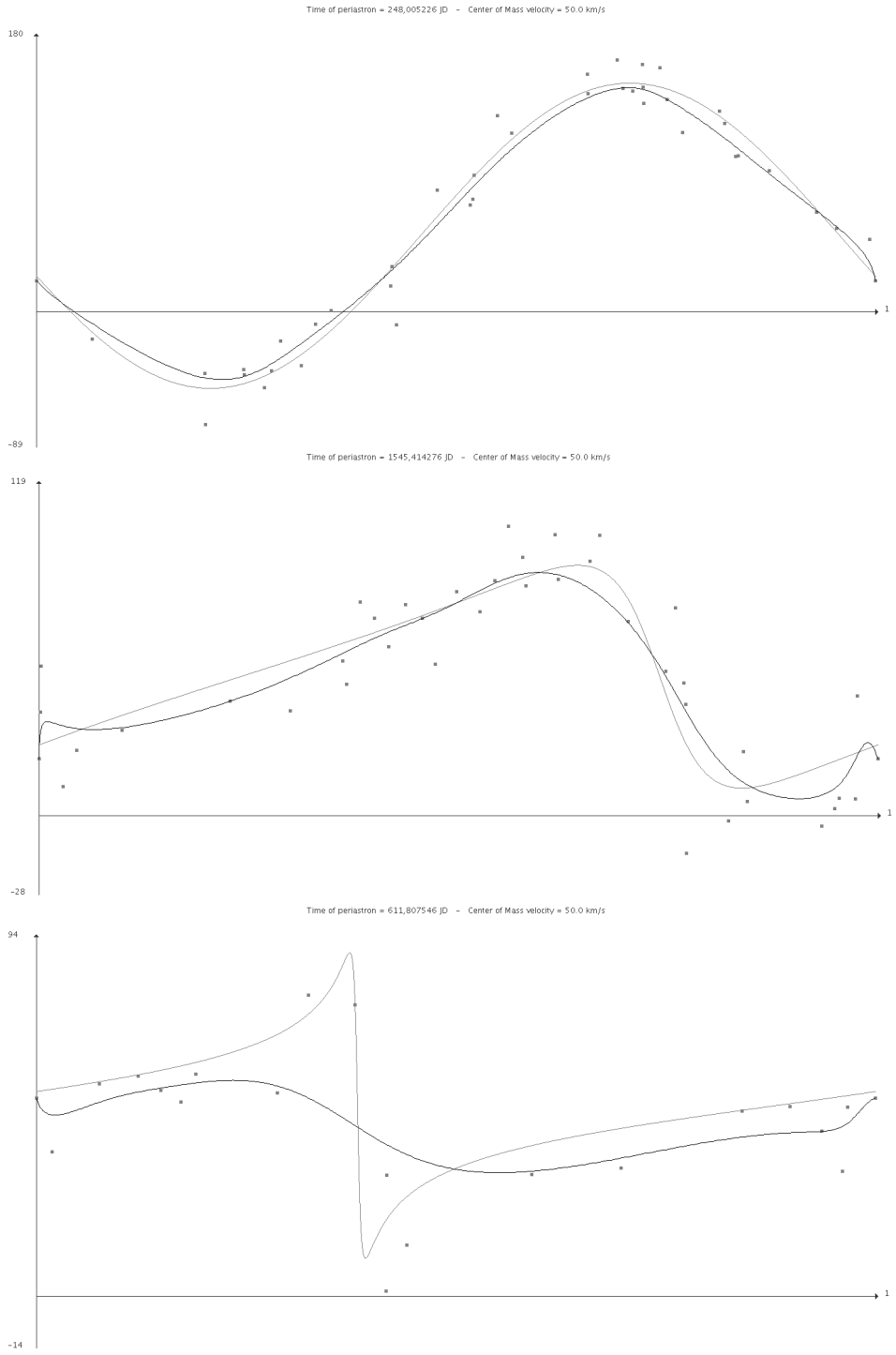

FIG. A.9 – Représentation des courbes de vitesses radiales, des observations liées et de l'ajustement par une courbe de Bézier de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

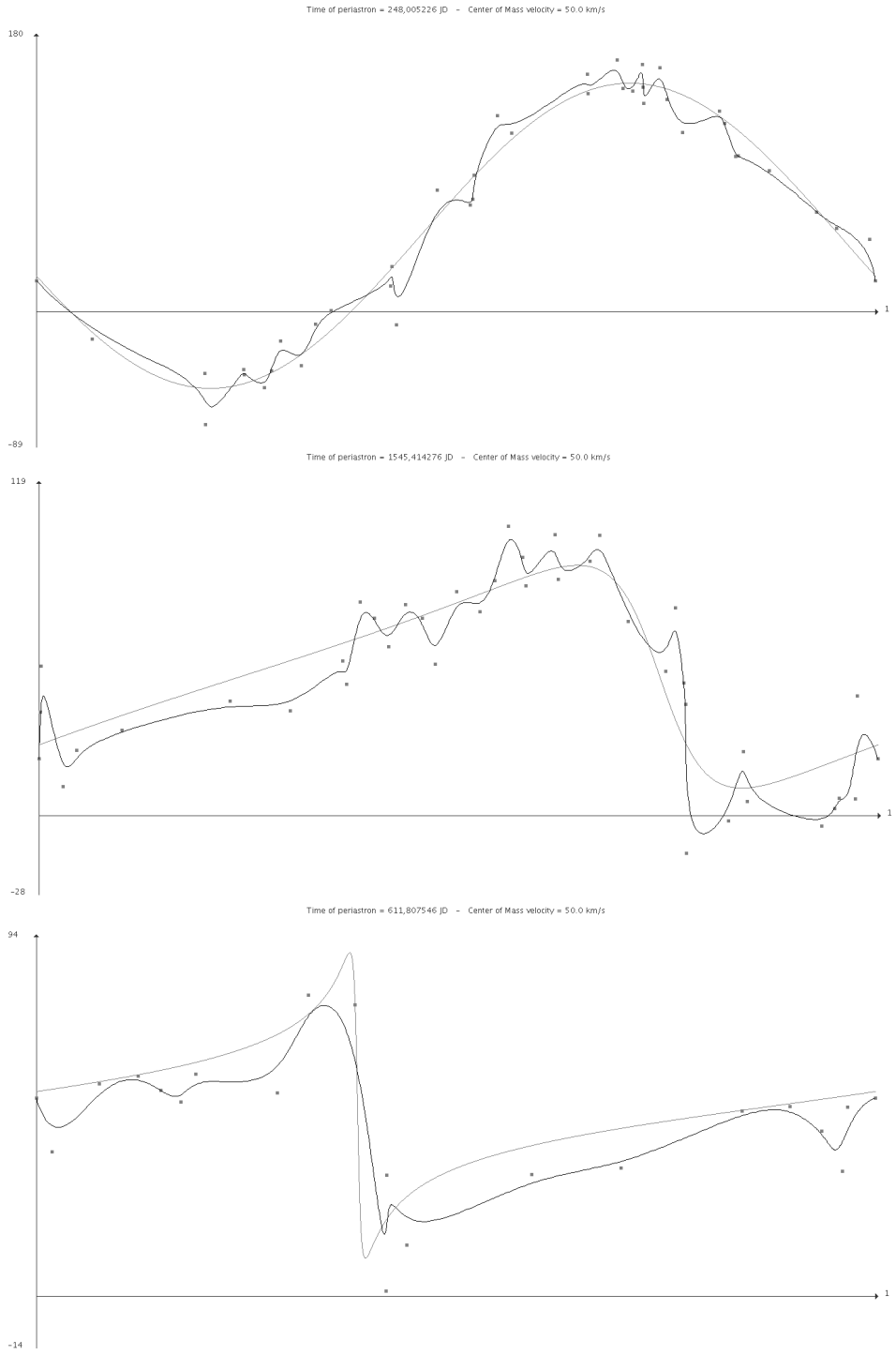

FIG. A.10 – Représentation des courbes de vitesses radiales, des observations liées et de l'ajustement par une B-spline cubique ouverte uniforme de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

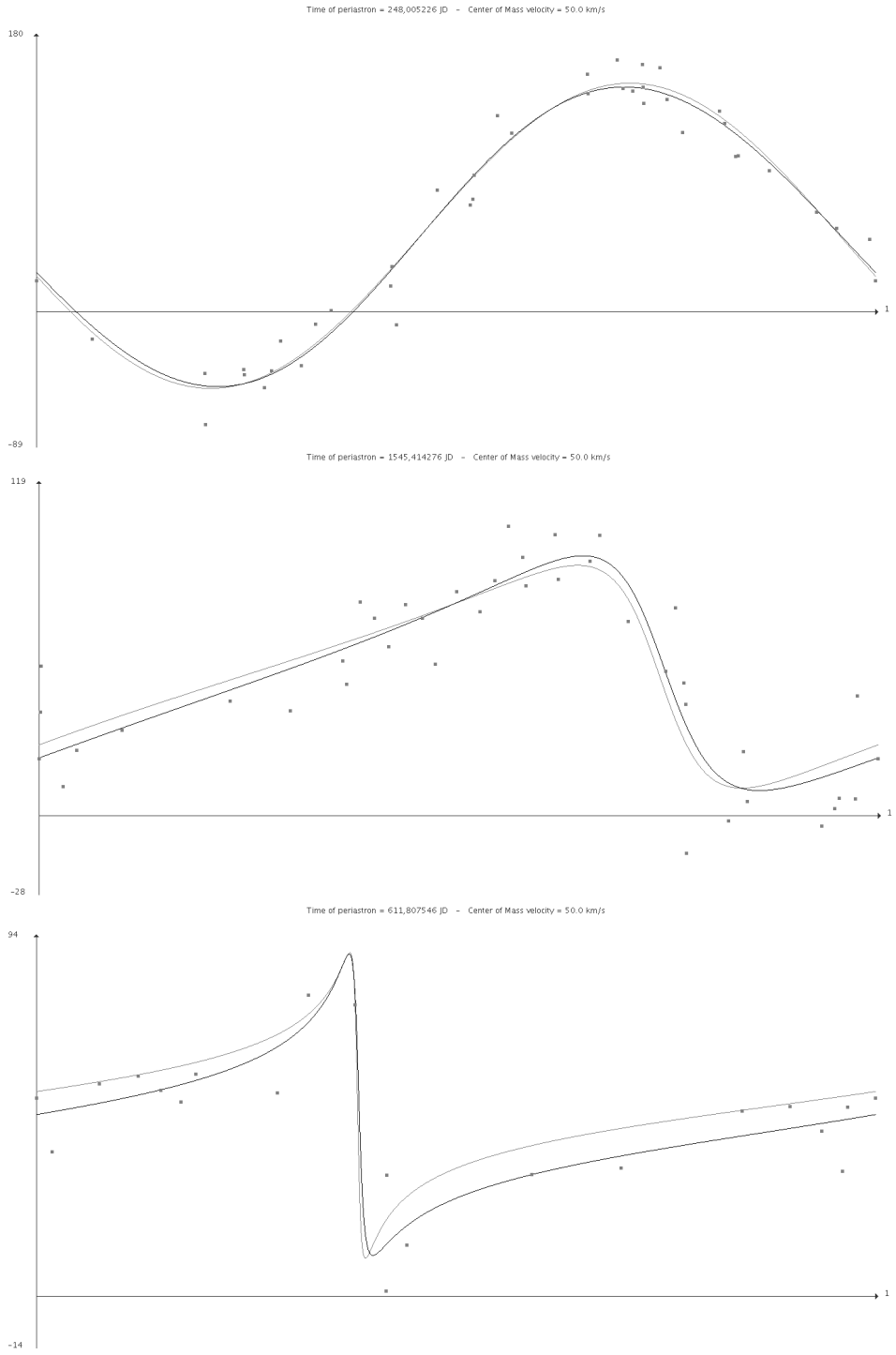

FIG. A.11 – Représentation des courbes de vitesses radiales, des observations liées et de l'ajustement par un algorithme génétique de ces observations pour les paramètres suivants (de haut en bas) : $V_{r0} = \{N = 40, \varpi = 0^o, e = 0, K = 100km/s\}$, $V_{r1} = \{N = 40, \varpi = 90^o, e = 0.5, K = 40km/s\}$, $V_{r2} = \{N = 20, \varpi = 90^o, e = 0.9, K = 40km/s\}$.

Bibliographie

[1] M. PERRYMAN. Gaia - taking the galactic census. *European Space Agency*, février 2006.

[2] Nature Publishing Group. *Encyclopedia of astronomy and astrophysic*, 2001.

[3] N. VANDEWALLE. La gravitation universelle, cours de physique générale. Université de Liège.

[4] J.M. HAMEURY D. EGRET, J.-L. HALBWACHS. Etoiles doubles. *Ecole CNRS de Goutelas XXIII*, 2000.

[5] S.G. BRUSH G.J. HOLTON. *Physics, the human adventure : from Copernicus to Einstein and beyond*. Rutgers University Press, 2001.

[6] C. FLAMMARION. *Les étoiles et les curiosités du ciel supplément de "l'Astronomie populaire"*. Marpon et Flammarion, 1882.

[7] Y. DAMERDJI. Resolving sb1 orbits for the gaia project. *GAIA DU434-WP00200*, mai 2008.

[8] J. GALBRAITH. The optics of planet hunting, avril 2010.

[9] W.D. HEINTZ. "Double stars". Reidel, 1978.

[10] C.M. BISHOP. *Pattern Recognition and Machine Learning*. Springer Science, 2006.

[11] E. HAIRER. Analyse numérique. *Université de Genève*, octobre 2001.

[12] A. ANDREEV B. SENDOV. *Approximation and Interpolation Theory VOL. III*. Elsevier Science, 1994.

[13] L.R. RABINER R.E. CROCHIERE. *Multirate Digital Signal Processing*. Prentice-Hall, 1983.

[14] C. MOLER. *Numerical Computing with MATLAB*. Society for Industrial and Applied Mathematics, 2004.

[15] P. BOURKE. Interpolation methods. *University of Western Australia*, Décembre 2009.

[16] H. AKIMA. A new method of interpolation ans smooth curve fitting based on local procedures. Technical report, ESSA Research Laboratories, 1970.

[17] D.S.G. POLLOCK. Smoothing with cubic splines. Queen Mary and Westfield College, The University of London.

[18] K.I. JOY. Catmull-rom splines, 2002.

[19] E. WEISSTEIN. Least squares fitting–polynomial.from mathworld–a wolfram web resource.

[20] W.H. PRESS S.A. TEUKOLSKY W.T. VETTERLING B.P. FLANNERY. *NUMERICAL RECIPES The Art of Scientific Computing Third Edition*. Cambridge university press, 2007.

[21] G.W. COLLINS. *Fundamental Numerical Methods and Data Analysis*. NASA Astrophysics Data System (ADS), 2003.

[22] D.G. MONET. A method for solving binary star orbits using the fourier transform. *Yerkes Observatory*, Mai 1979.

[23] D. SALOMON. *Curves and Surfaces for Computer Graphics*. Springer Science, 2006.

[24] P. GEURTS L. WEHENKEL. Cours d'apprentissage inductif appliqués, 2009.

[25] J. CHODOROWSKI L. MICLET. Apprentissage et evaluation de modèles de langage par des techniques de correction d'erreurs.

[26] J. FRIEDMAN T. HASTIE, R. TIBSHIRANI. *The Elements of Statistical Learning*. Springer Science, 2009.

[27] L. WEHENKEL P. GEURTS, D. ERNST. Extremely randomized trees. *Machine Learning*, 3(1), 2006.

[28] M. MITCHELL. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[29] S.E. HAUPT R.L. HAUPT. *PRACTICAL GENETIC ALGORITHMS*. Wiley-interscience, 2004.

[30] E.K. PREBYS. The genetic algorithm in computer science. *MIT Undergraduate Journal of Mathematics, MIT Press*, juin 1999.

[31] K.P. IRVINE. *Assembleur x86*. CampusPress, 2003.

[32] B.L. Miller. Genetic algorithms, selection schemes, and the varying effects of noise. *Dept. of computer science, university of Illinois*, août 1996.

[33] A.A.R. TOWNSEND. Genetic algorithms – a tutorial, 2003.