Titre: Title:	Système de synthèse de l'écriture manuscrite par l'utilisation du modèle Sigma-lognormal : bilan de la conception et documentation de l'application SIMSCRIPT
Auteurs: Authors:	Christian O'Reilly, Moussa Djioua, & Réjean Plamondon
Date:	2006
Type:	Rapport / Report
Référence: Citation:	O'Reilly, C., Djioua, M., & Plamondon, R. (2006). Système de synthèse de l'écriture manuscrite par l'utilisation du modèle Sigma-lognormal : bilan de la conception et documentation de l'application SIMSCRIPT. (Technical Report n° EPM-RT-2005-09). https://publications.polymtl.ca/3148/

Institution:	École Polytechnique de Montréal
Numéro de rapport: Report number:	EPM-RT-2005-09
URL officiel: Official URL:	
Mention légale: Legal notice:	

EPM–RT–2005-09

SYSTÈME DE L'ÉCRITURE MANUSCRITE PAR L'UTILISATION DU MODÈLE SIGMA-LOGNORMAL BILAN DE LA CONCEPTION ET DOCUMENTATION DE L'APPLICATION SIMSCRIPT

Christian O'Reilly, Moussa Djioua, Réjean Plamondon
Département de Génie électrique
École Polytechnique de Montréal

Janvier 2006

EPM-RT-2005-9

SYSTÈME DE SYNTHÈSE DE L'ÉCRITURE MANUSCRITE PAR L'UTILISATION DU MODÈLE SIGMA-LOGNORMAL

BILAN DE LA CONCEPTION ET DOCUMENTATION DE L'APPLICATION SIMSCRIPT

Christian O'Reilly
Moussa Djioua
Réjean Plamondon

Département de Génie Électrique
École Polytechnique de Montréal

janvier 2006

Dépôt légal :
Bibliothèque nationale du Québec, 2006
Bibliothèque nationale du Canada, 2006

EPM-RT-2005-09
Système de synthèse de l'écriture manuscrite par l'utilisation du modèle sigma-lognormal
Bilan de la conception et documentation de l'application SimScript
par : **Christian O'Reilly, Moussa Djioua, Réjean Plamondon**
Département de génies électrique
École Polytechnique de Montréal

Abstract

An application was built to simulate complex movements with the use of the sigma-lognormal model. The theory used to conduct simulation is shortly described, the built application interface and the applications functionalities are shown and discussed. Finally some simulations results are compared with typical handwritten data acquired from a subject with the use of a digitizer.

1. Introduction

En 1995, Plamondon a développé la version scalaire loi delta-lognormale (Plamondon, 1995a,b) servant à modéliser le profil de vitesse curviligne d'un mouvement humain. En 1998, pour pouvoir rendre compte des caractéristiques des trajectoires obtenues lors de mouvements, une version vectorielle de la loi delta-lognormale a été développée (Plamondon et al. 1998b.). Elle a démontré son efficacité tant pour la reproduction des profils cinématiques obtenus expérimentalement que pour rendre compte des diverses observations empiriques relevées dans l'étude des mouvements humains. Considérons à titre d'exemple la loi de la puissance 2/3 (Plamondon et al. 1998a.) et le compromis vitesse-précision (Plamondon, R. et al. 1997). Dernièrement, Varga et al. (2005) ont d'ailleurs utilisé ce modèle pour réaliser de la généralisation de caractères.

En 2005, Plamondon et Djioua ont proposé une nouvelle forme vectorielle, désignée par modèle sigma-lognormal, pour généraliser le modèle delta-lognormal permettant ainsi de modéliser des mouvements complexes avec plus de variabilités. Certains résultats ont été obtenus grâce à une application, nommée Simdlogn 2D, pour la simulation de mouvements élémentaires. L'application SimScript, décrite dans le présent rapport, a été développée dans l'optique d'effectuer des simulations de mouvements complexes à l'aide du modèle sigma-lognormal.

Dans la suite du document, les bases théoriques du modèle sigma-lognormal sont décrites dans un premier temps. L'application Simdlogn 2D est ensuite succinctement présentée suivie d'une description plus complète de la conception et de la réalisation de SimScript. Finalement les résultats de simulation à l'aide de SimScript sont présentés et comparés avec des données expérimentales acquises à l'aide d'une tablette à numériser.

2. Présentation du modèle sigma-lognormal

2.1 Mise en contexte

Récemment, Plamondon et Djioua (2005) ont proposé une généralisation du modèle delta-lognormal en utilisant le modèle sigma-lognormal. C'est un modèle vectoriel qui tient compte de l'information spatiale du mouvement. Le mouvement humain est considéré comme une superposition vectorielle de mouvements élémentaires ayant une direction et un sens, et dont le module de sa vitesse curviligne est une fonction lognormale. Dans le cas d'un mouvement rapide, la forme générale du modèle, donnée à la Figure.1, illustre la modélisation du système neuromusculaire. Dans ce cas, le mouvement rapide est considéré comme la superposition de deux mouvements élémentaires produits respectivement par les systèmes neuromusculaires agoniste et antagoniste. La généralisation de cette théorie permet de représenter n'importe quel mouvement complexe. Dans un premier temps, on s'intéresse à la mise en évidence de la capacité du modèle sigma-lognormal à synthétiser des mouvements artificiels similaires à ceux produits pas des sujets humains. Cette similarité 'morphologique' doit être assurée tant au niveau du tracé complexe qu'au niveau du profil de vitesse. Ce rapport résume les aspects théoriques du modèle ainsi que son implantation sous la forme d'une application de synthèse des trajectoires dans un espace à deux dimensions, et plus particulièrement la synthèse de l'écriture manuscrite.

2.2 Représentation d'un mouvement élémentaire

Un mouvement élémentaire correspond au mouvement effectué par l'effecteur terminal du membre supérieur suite à l'activation isolée du système neuromusculaire agoniste sans l'activation de son antagoniste ou vice versa. Selon le modèle sigma-lognormal, sa vitesse curviligne est un vecteur dont le module possède un profil lognormal.

En considérant le système neuromusculaire agoniste, le profil de vitesse, qui correspond au module du vecteur $\vec{v}_1(t)$, est donné par la relation suivante :

$$v_1(t) = \frac{\left\| \vec{D}_1 \right\|}{\sigma_1 \sqrt{2\pi}(t - t_{01})} e^{-\frac{1}{2\sigma_1^2}\left[\ln(t-t_{01})-\mu_1\right]^2} \quad (1)$$

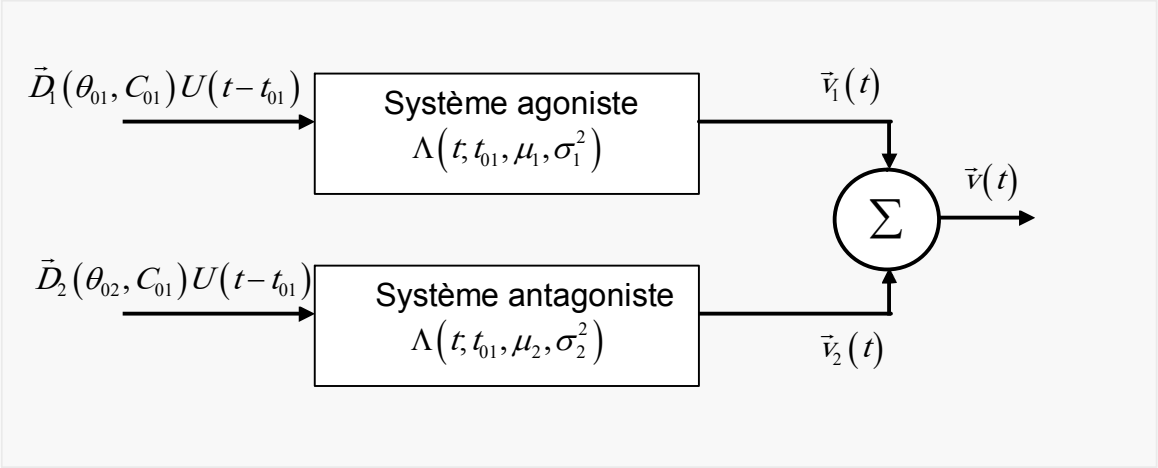


Figure.1. Représentation selon le modèle sigma-lognormal du système neuromusculaire générateur d'un mouvement simple et rapide.

La direction du vecteur $\vec{v}_1(t)$ dépend de la direction globale du mouvement, donnée par l'angle de la direction initiale θ_{01} et par la courbure constante C_{01}, induite par la posture adoptée pour effectuer le mouvement. La Figure.2 illustre une situation recréant un mouvement élémentaire généré par le modèle sigma-lognormal de la composante agoniste du système neuromusculaire.

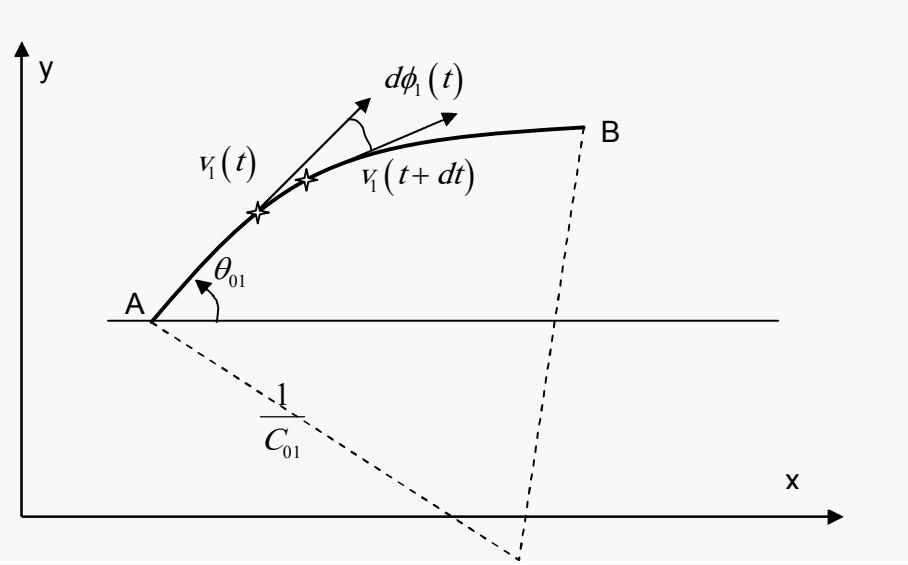


Figure.2. Illustration de la trajectoire décrite par un mouvement élémentaire et générée par la composante agoniste du système neuromusculaire.

D'après la définition de la notion de courbure[1], l'angle $\varphi_1(t)$ que fait le vecteur $\vec{v}_1(t)$ avec l'horizontal est donné par la relation suivante :

$$\varphi_1(t) = \theta_{01} + C_{01}\int_{t_0}^{t} v_1(\tau)d\tau \tag{2}$$

Et par suite, les coordonnées cartésiennes des points de la trajectoire sont données par les relations suivantes :

$$v_{1x}(t) = v_1(t)\cos(\varphi_1(t)) \tag{3}$$

$$v_{1y}(t) = v_1(t)\sin(\varphi_1(t)) \tag{4}$$

Et

$$x(t) = \int_{t_0}^{t} v_{1x}(\tau)\, d\tau \tag{5}$$

$$y(t) = \int_{t_0}^{t} v_{1y}(\tau)\, d\tau \tag{6}$$

[1] Notons que dans ce cas, la dimension de la courbure est le radian par centimètre [rd/cm]. Elle représente la variation par centimètre de l'angle au centre qui intercepte le trait courbe

2.3 Représentation d'un mouvement complexe

Selon le modèle sigma-lognormal, la vitesse curviligne d'un mouvement complexe est la résultante vectorielle des vitesses curvilignes de ses mouvements élémentaires. Ainsi, si le mouvement est constitué de M mouvements élémentaires générés par les systèmes neuromusculaires agoniste et antagoniste, sa vitesse curviligne est donnée par :

$$\vec{V}(t) = \sum_{i=1}^{M} \vec{V}_i(t) \tag{7}$$

Et ses coordonnées cartésiennes sont données par

$$V_x(t) = \sum_{i=1}^{M} v_i(t) \cos\left(\theta_{0i} + C_{0i} \int_{t_{0i}}^{t} v_i(\tau) d\tau \right) \tag{8}$$

$$V_y(t) = \sum_{i=1}^{M} v_i(t) \sin\left(\theta_{0i} + C_{0i} \int_{t_{0i}}^{t} v_i(\tau) d\tau \right) \tag{9}$$

Avec

$$v_i(t) = \frac{D_i}{\sigma_i (t - t_{0i}) \sqrt{2\pi}} \exp\left\{ \frac{1}{2\sigma_i^2} \left[\ln(t - t_{0i}) - \mu_i \right]^2 \right\} \tag{10}$$

Chaque mouvement élémentaire d'indice i est complètement décrit dans un espace à deux dimensions par les six paramètres suivants : $(t_{0i}, D_i, \mu_i, \sigma_i, \theta_{0i}, C_{0i})$.

La synthèse de l'écriture manuscrite consiste à générer la trajectoire du mouvement correspondant à partir des paramètres sigma-lognormaux. Les coordonnées des points de cette trajectoire sont déduites en utilisant les relations suivantes :

$$x(t) = \int_0^{\infty} V_x(\tau)\, d\tau$$

$$y(t) = \int_0^{\infty} V_y(\tau)\, d\tau$$

Le schéma synoptique global du système de synthèse de l'écriture manuscrite est illustré à la Figure.3.

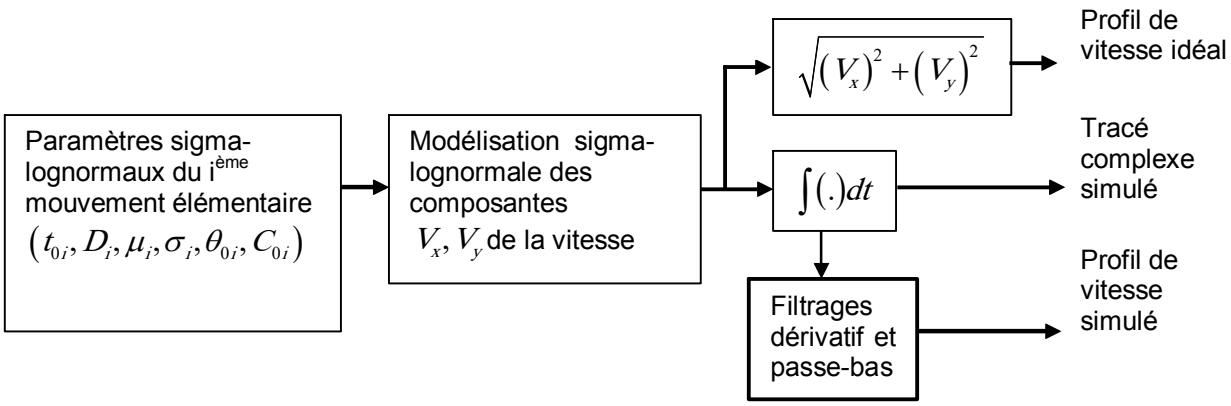


Figure.3. Illustration du schéma général du système de synthèse de l'écriture manuscrite en utilisant le modèle sigma-lognormal.

3. Présentation de Simdlogn 2D

La conception de SimScript a été réalisée en prenant comme point de départ le code et l'interface de l'application Simdlogn 2D. Ci-dessous, nous présenterons rapidement cette application.

3.1 Objectifs

L'objectif de Simdlogn 2D était de permettre la simulation de mouvements rapides à l'aide du modèle sigma-lognormal. Ceci a permis dans un premier temps de mieux comprendre par l'expérimentation les implications des différents éléments du modèle

sigma-lognormal et, dans un second temps, de valider les algorithmes d'extraction développés au laboratoire.

3.2 Interface

L'interface de Simdlogn 2D est principalement constituée de contrôles affichant deux profils de vitesse. Le premier illustre une delta-lognormale, c'est-à-dire la soustraction de deux lognormales. Le second affiche une sigma-lognormale, c'est-à-dire qu'il considère l'addition des deux lognormales ayant une orientation arbitraire spécifiée par les angles θ_1 et θ_2. L'interface comprend aussi la trace que génère un tel profil ainsi que tous les contrôles nécessaires pour modifier les paramètres en jeu et observer en direct les modifications résultantes sur le profil de vitesse et la trace. L'écran principal de l'interface est affiché à la figure 4.

3.3 Problèmes et lacunes

Le besoin de mettre au point SimScript comme remplacement à Simdlogn 2D vient tout d'abord d'une différence dans les objectifs visés. La première version visait à simuler des mouvements rapides simples alors que la deuxième version vise la simulation de mouvements complexes.

Simdlogn 2D avait aussi un problème de fuites de mémoire et était insuffisamment documentée. La conception de SimScript cherchera donc à combler ces deux lacunes.

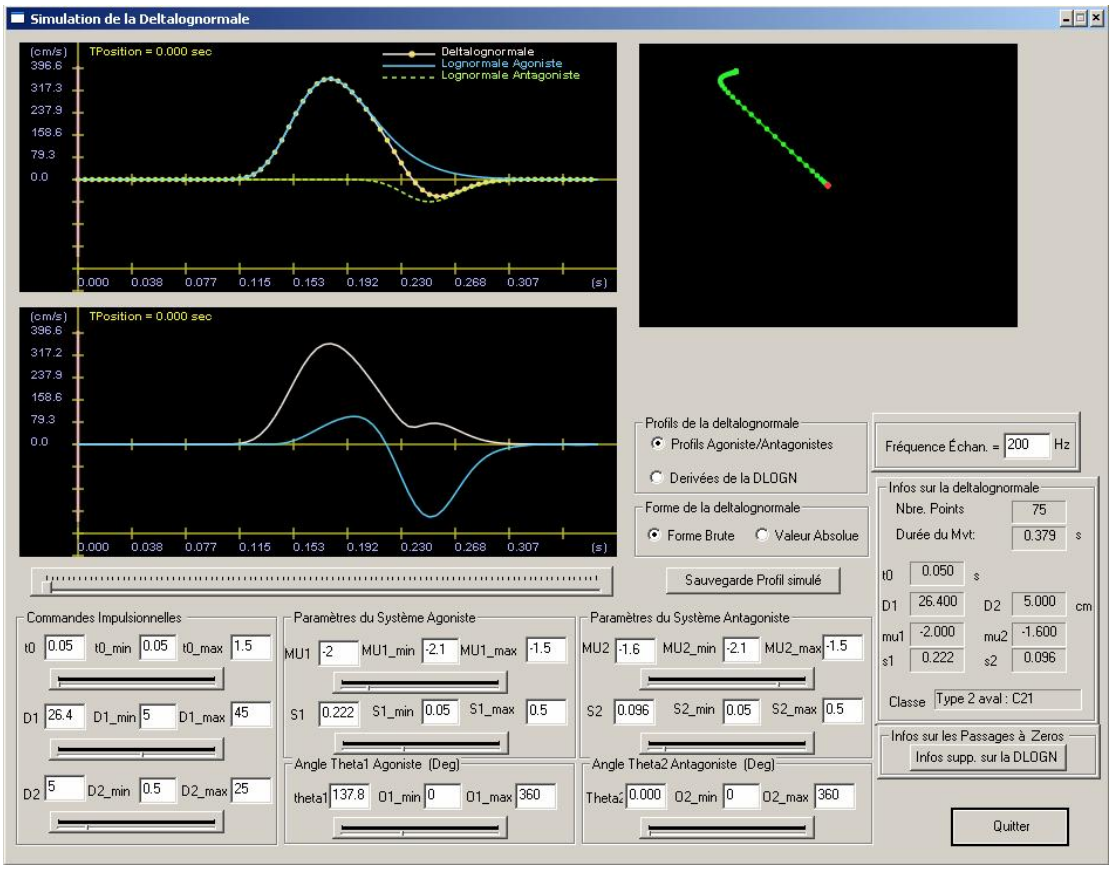


Figure.4. Interface de la première version de l'application.

4. Design de SimScript

4.1 Objectifs

L'application doit permettre la simulation des tracés complexes, à l'aide de la superposition de plusieurs vecteurs de vitesse, dont les modules ont des profils lognormaux. Plus spécifiquement, elle doit être conçue pour permettre de faire la synthèse de l'écriture manuscrite. On doit aussi pouvoir générer un fichier numérique d'un tracé complexe dans le même format que ceux obtenus lors de l'acquisition de données réelles. Ces fichiers pourront ensuite servir à tester les algorithmes d'extractions développés au laboratoire Scribens.

4.2 Outils de conception et conventions de programmation

La conception de SimScript s'est voulue en continuité avec les choix de conception qui ont été pris lors de la réalisation de l'application Sign@médic. Nous reverrons succinctement ces choix. Pour une version plus complète de cette discussion, le lecteur est encouragé à se référer à la section 5 de la version du 30 juin 2004 du rapport technique interne « Bilan et documentation de l'application Sign@médic ».

Comme il a été souligné dans la section précédente, le code de base a été tiré de Simdlogn 2D. À ce code a été intégré autant que possible le code développé pour l'application Sign@médic. Le but de cette intégration est de permettre éventuellement la constitution d'une boîte à outil de programmation qui serait utile pour les divers projets de programmation du laboratoire. L'avantage d'une telle librairie de fonctions et d'objets C++ est de permettre le développement rapide de nouvelles applications, qui seront plus fiables et mieux documentées puisque réutilisant un code déjà éprouvé.

Le développement de SimScript a été effectué sur CVS. Nous disposons donc d'une base de données de toutes les modifications effectuées.

Le programme Doxygen a été utilisé pour générer la documentation du projet à l'aide de balises insérées à même les commentaires dans le code. Les commentaires Doxygen des divers groupes d'objets ont été groupés en modules de façon à disposer d'un classement méthodique de la documentation.

La programmation de l'application a été entièrement réalisée dans l'environnement de travail Microsoft Visual C++ .Net. L'interface a été construite à l'aide des MFC (Microsoft Foundation Classes). Ces choix sont en continuité avec les outils utilisés dans le laboratoire Scribens pour les projets de programmation précédents. On s'assure ainsi une compatibilité entre les divers projets.

Le nom des classes, des variables, des données, des fonctions membres, de l'identification des pointeurs et des paramètres d'une méthode suivent des règles standardisées communément acceptées et utilisées dans le domaine de la programmation. On trouvera dans le tableau 4.1 les principales nomenclatures qui ont été suivies lors du codage.

Les conventions de codage sont essentiellement issues des exemples présentés dans le site internet suivant :

http://www.possibility.com/Cpp/CppCodingStandard.html#important.

Concept	Exemple

Accesseur de classe	getTraceData
Attribut de classe	m_WacomTabletPressure
Classe	ExperimentProtocol
Constante	INTUOS2_FREQUENCY
Énumération	TEST_TYPE
Fonction	test_classe_acquisition
Instance de classe	experimentProtocol
Méthode de classe	StartAcquisition
Paramètre d'une méthode	r_CurrentSubject
Pointeur	p_TraceList
Variable globale	g_NumberOfTrials
Variable statique	s_WacomSensibilityLevel
Variables	data_point_number

Tableau 4.1 - Exemples de nos choix de standardisation des conventions de codage

Par contre, il est important de noter que ces standards de programmation sont inégalement respectés à travers tout le projet, le code de base venant de diverses sources qui n'appliquaient pas nécessairement ces standards. Dans l'optique de la conception d'une boîte à outil de programmation, il serait bon d'écrire un standard plus exhaustif et de corriger le code de façon à ce qu'il respecte complètement ces standards.

4.3 Traitement des données

De façon à faciliter l'organisation de la documentation et du code, les classes relatives au traitement des données ont été regroupées principalement en trois modules : View, Filter et Engin.

4.3.1 View

Le module View regroupe toutes les classes relatives à l'affichage ou à l'interface qui peuvent être aisément réutilisées dans diverses applications actuelles et à venir. Le tableau 4.2 présente la liste des principales classes de ce groupe et le tableau 4.3 pour sa part décrit les structures utilisées.

Classes	**Définition**
CColorButton	Cette classe permet d'utiliser des boutons dont on peut personnaliser la couleur. Cette fonctionnalité n'est pas disponible

	aisément avec les classes de base fournies par les MFC. CColorButton est dérivée de la classe CButton.
GLDrawing	Cette classe permet d'afficher un objet CSignal en OpenGL. Elle contient un objet de type CProfilView et permet d'utiliser celui-ci de façon plus simple.
CProfilView	Classe de base permettant d'afficher un objet de type CSignal.
CGlViewXY	Dérivation de la classe CProfilView utilisée pour afficher la trace générée par le mouvement représenté dans un objet de type CSignal.
ProfilView	Classe englobant l'utilisation d'un objet de type GLDrawing. Bien que cette classe ne soit pas abstraite, elle n'est pas conçue de façon à être utilisée directement. Au besoin, l'on peut utiliser les classes qui en ont été dérivée : CProfilViewSimul et CProfilViewNumeric.
CProfilViewSymbolic	Classe dérivée de ProfilView. Elle offre une façon simple d'afficher la trace contenue dans un objet de type CTraceSimul.
CSliderExt	Cette classe est dérivée de la classe MFC CSliderCtrl. Elle est utilisée pour gérer un contrôle constitué d'une glissière et de trois boîtes de texte, l'une pour la valeur courante de la glissière, l'autre pour sa valeur maximale et la dernière pour sa valeur minimale.
CTabCtrlObj	Cette classe est dérivée de la classe MFC CTabCtrlObj et est utilisée pour contrôler une série d'onglets auxquels sont associés des formulaires particuliers.
TraceView	Classe contenant un objet de type CGlViewXY permettant d'englober les manipulations nécessaires pour l'affichage d'une trace. Cette classe n'est pas utilisée directement. On utilise plutôt les classes dérivées CTraceViewAcq et CTraceViewSimul.
CTraceViewSimul	Gère l'affichage d'un objet de type CTraceSimul. Cette classe est dérivée de la classe TraceView.
SignalID	Classe servant à identifier uniquement une composante d'un signal. Par exemple, dans un vecteur de plusieurs objets CSignal, on peut identifier avec un objet SignalID la composante vitesse selon l'axe des X du troisième objet CSignal du vecteur.

Tableau 4.2 – Principales classes du module View

Structures	**Définition**

Color	Une structure adaptée à la manipulation d'une couleur RGB dans le cadre de l'utilisation de OpenGL.

Tableau 4.3 – Principales structures du module View

4.3.2 Filter

Plusieurs classes ont été conçues de façon à permettre de gérer facilement le filtrage des données numériques. Le design de ce module a été fait afin d'offrir un maximum de convivialité autant pour la création de nouveaux filtres que pour le filtrage des données à l'aide de filtres déjà conçus.

Les objectifs principaux dans la conception de l'architecture de ce module sont de minimiser les possibilités de fuites de mémoire et de permettre à l'utilisateur du module de manipuler une interface de filtrage indépendante du type de filtre utilisé.

Cette architecture a été grandement bâtit sur le principe du patron de conception « Abstract Factory ». Une description complète de ce patron peut être trouvée dans le livre *Design Patterns - Elements of reusable object-oriented* software écrit par Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides.

Selon ce patron, la classe principalement utilisée par l'utilisateur du module est FilterFactory. Par contre, pour offrir plus de convivialité et pour réduire les risques de fuite de mémoire, l'utilisation de cet objet a été englobée par la classe CFilter.

Classes	**Définition**
AbstractFilter	Classe virtuelle pure servant à implémenter des filtres.
DerivativeFilter	Implémentation d'un filtre dérivatif.
CFilter	Objet englobant l'utilisation de la classe AbstractFilter et FilterFactory dans le but d'en faciliter l'utilisation.
FilterFactory	Classe statique permettant de créer des filtres spécifiques à partir d'une instance générique de AbstractFilter.
LowPassMeteorFilter	Implémentation d'un filtre passe-bas.

Tableau 4.4 – Principales classes du module Filter

4.3.3 Engin

Le module Engin contient les classes principales nécessaires à la représentation et au traitement des données. Le tableau 4.5 contient les objets les plus importants de ce module pour la simulation.

Classes	Définition
CSignal	Contient un tableau de données représentant le signal numérique sous ses diverses formes (position, vitesse, accélération, etc.). Cette classe possède aussi plusieurs méthodes permettant d'effectuer le filtrage des données et le calcul de valeurs dérivées. À titre d'exemple, on peut obtenir la vitesse et l'accélération à partir de la position en fonction du temps.
CLognormal	Cette classe contient principalement les paramètres d'une équation lognormale. Elle contient aussi les méthodes nécessaires pour calculer le signal numérique généré par la lognormale analytique.
Trace	Classe générique à partir de laquelle on peut dériver des classes pour représenter une trace dont on a faite l'acquisition ou que l'on simule.
CTraceSimul	Classe dérivée de Trace. Cette implémentation est utilisée pour la représentation de traces simulées. Elle contient une liste de lognormales analytiques.

Tableau 4.5 – Principales classes du module Engin

4.4 Interface

Dans le cadre de ce projet, le principal élément à considérer dans la conception de l'interface usager de l'application est de permettre à l'utilisateur de modifier les paramètres de toutes les lognormales qui forment le signal simulé et d'en voir les résultats instantanément. Cette interaction directe permet de manipuler de façon plus aisée et plus rapide le profil de vitesse et la trace générée. Elle permet aussi de comprendre plus facilement les implications pratiques résultant de la théorie. C'est dans cette optique d'interaction directe que tous les paramètres de la lognormale active - celle pointée par le curseur triangulaire dans le cadre « Lognormales » - peuvent être modifiés dans l'onglet « Ajout Log » à l'aide de glissières.

L'interface est constituée principalement de quatre blocs. Le premier, le cadre « Lognormales », liste les lognormales formant le signal en construction. Pour chacune des lognormales, on peut y voir : les paramètres de la lognormale, une case à cocher

permettant de la soustraire au calcul sans pour autant l'effacer et un bouton permettant de l'effacer complètement. On peut gérer un nombre arbitraire de lognormales grâce à une barre de défilement qui apparaît lorsque nécessaire. Pour faciliter les manipulations, les lognormales sont constamment reclassées en ordre croissant du paramètre t_0. Un curseur triangulaire noir indique la lognormale en traitement.

Le deuxième élément important de l'interface est constitué du profil de vitesse du signal. Tel qu'illustré à la figure 3, on peut y afficher au choix le profil idéal et/ou simulé ainsi que le signal de pression. La couleur des différentes courbes affichées peut être changée au choix.

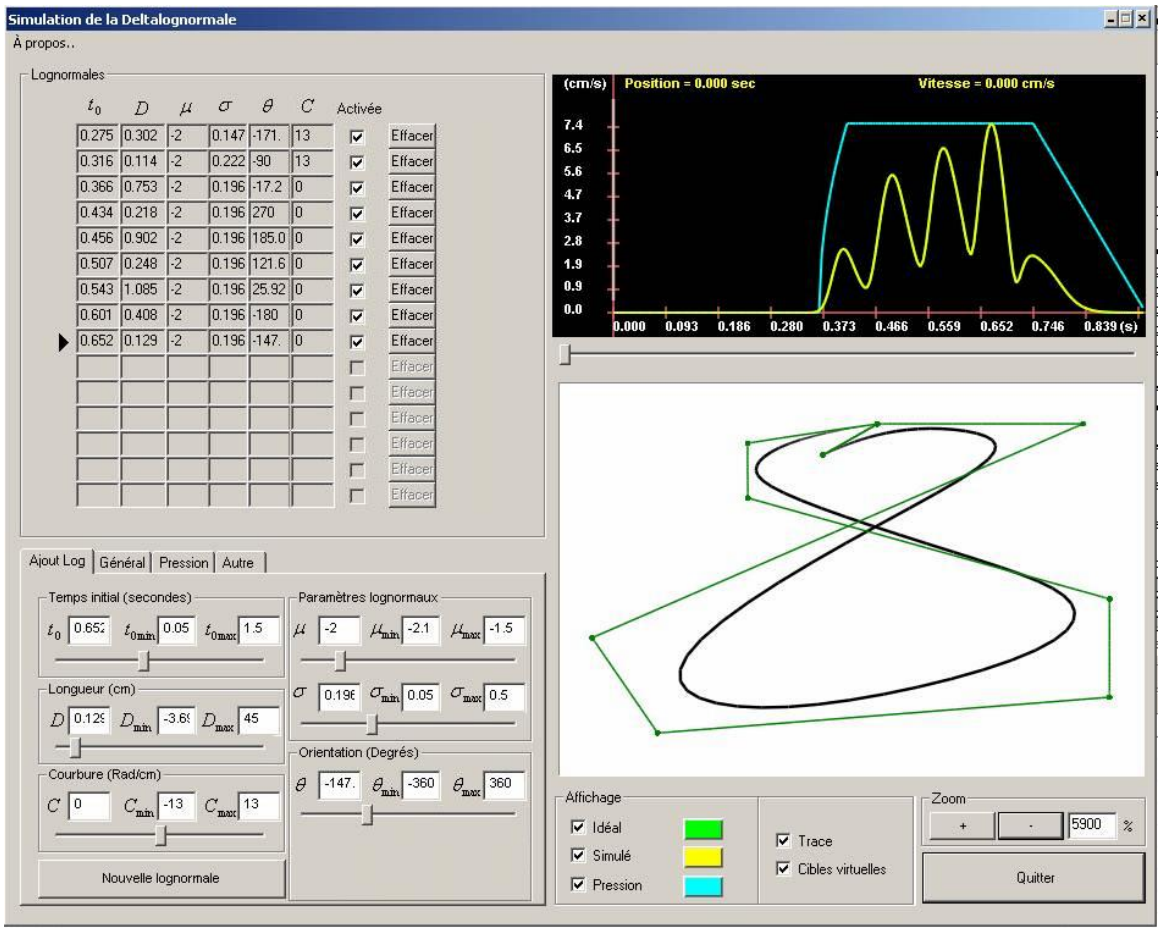


Figure.5. Vue d'ensemble de l'interface de SimScript.

Le troisième bloc est constitué de la trace générée par le profil de vitesse. On peut aussi afficher sur le même contrôle les cibles virtuelles constituant le plan moteur spécifique à ce profil de vitesse. Un contrôle a été ajouté pour agrandir ou rapetisser la trace (effet de zoom). Une glissière permet de déplacer un curseur en simultané sur le profil de vitesse et sur la trace. On peut ainsi visualiser, grâce à la barre rouge verticale sur le profil de vitesse et au point rouge sur la trace, quelle partie de la trace et du profil de vitesse sont en correspondance.

Le dernier bloc constituant l'interface est un contrôle contenant quatre onglets. Le premier, « Ajout Log », permet de modifier la lognormale en traitement ou d'ajouter une nouvelle lognormale à la trace. Tous les éléments décrits plus haut sont illustrés à la figure 5.

Pour sa part, l'onglet « Général » permet de déplacer le point de départ de la trace grâce à des glissières. Il contient aussi les contrôles nécessaires pour enregistrer le mouvement simulé dans un format numérique ou analytique ainsi qu'un bouton chargeant un fichier précédemment sauvegardé en format analytique. Finalement, il fournit le nombre d'échantillons du signal numérique généré, sa durée ainsi que sa fréquence d'échantillonnage.

L'onglet « Pression » permet de gérer au besoin la forme du signal de pression. On peut ainsi générer des levés de crayons. Ceci est indispensable pour représenter la trace de certaines lettres, par exemple pour un « t ». L'interface de ces deux derniers est illustrée à la figure 6.

Finalement, l'onglet « Autre » a été ajouté. Il est présentement vide mais disponible pour des ajouts ultérieurs.

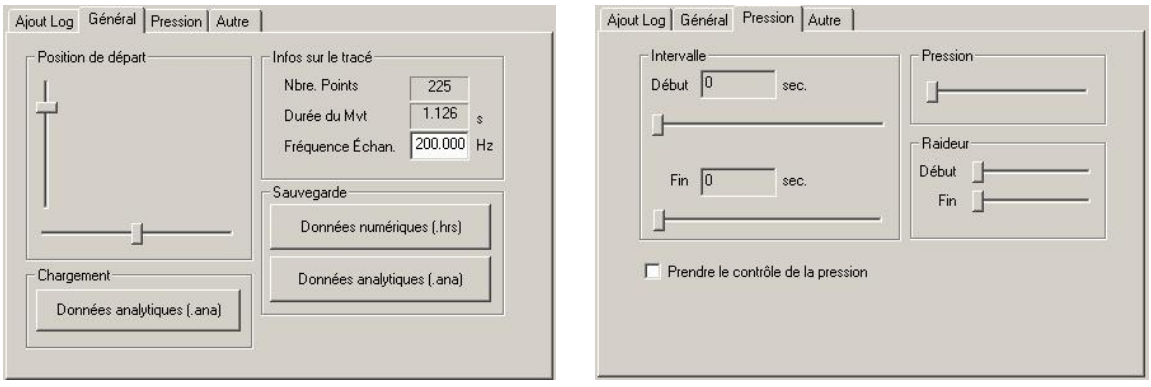


Figure.6. Vue du contenu des onglets « Général » et « Pression ».

5. Simulations et résultats

Pour tester les capacités de SimScript, nous avons comparé les caractères simulés avec des caractères manuscrits réels. Les lettres réelles ont été enregistrées à l'aide d'une tablette à numériser Wacom standard et le logiciel d'acquisition Sign@médic développé au laboratoire.

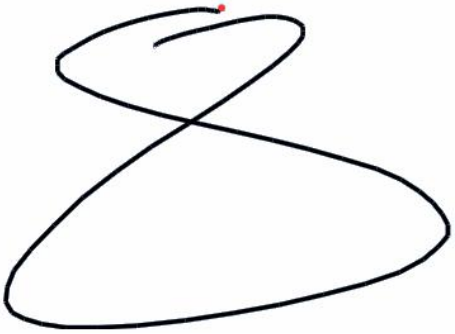
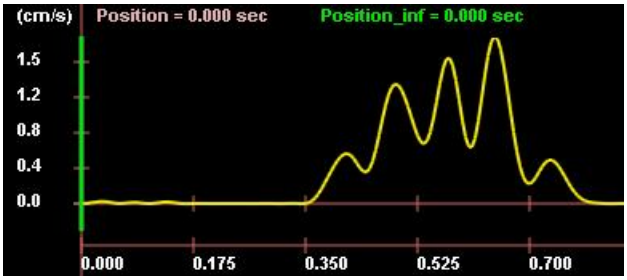
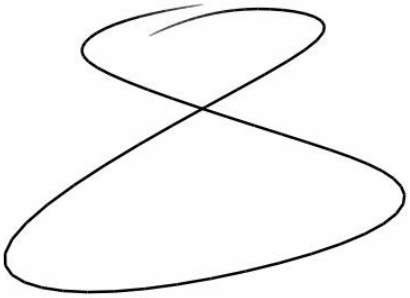
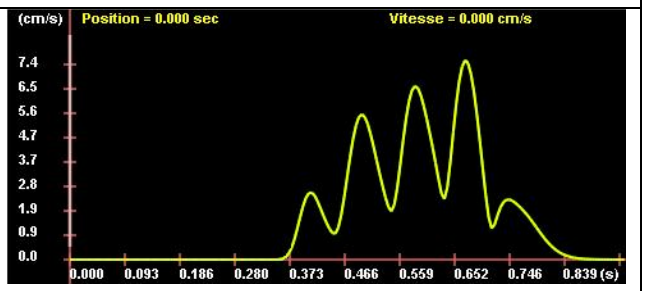
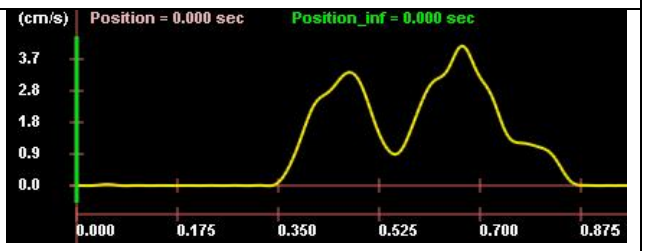
Les traces et les profils de vitesse acquis et simulés de deux caractères différents sont présentés à la figure 7.

On constate qu'il est possible d'obtenir par simulation de l'écriture manuscrite dont la trace et le profil de vitesses sont similaires à ceux de l'écriture manuscrite numérisée par le système sign@medic.

6. Conclusion

L'application développée et présentée dans ce document permet aisément de visualiser les implications de la généralisation du modèle delta-lognormal par le modèle sigma-lognormal. Si le modèle delta-lognormal a pu reproduire la plupart des variabilités observées au niveau du profil de vitesse, avec le modèle sigma-lognormal, on peut particulièrement y reproduire la plupart des variabilités observées au niveau du tracé d'un mouvement rapide. Généralement, l'application SimScript est particulièrement adaptée à la synthèse de l'écriture manuscrite et constitue un outil pratique et puissant pour la compréhension du modèle sigma-lognormal.

Il est proposé que dans un travail ultérieur, un document sur les standards de programmation à utiliser dans le développement d'applications en C++ au laboratoire Scribens soit rédigé. Il est aussi recommandé qu'une boîte à outil de programmation soit construite de façon à augmenter l'efficacité du développement d'applications et la qualité des programmes résultants.

	Traces	Profils
ACQUISITION		
SIMULATION		
ACQUISITION		
SIMULATION		

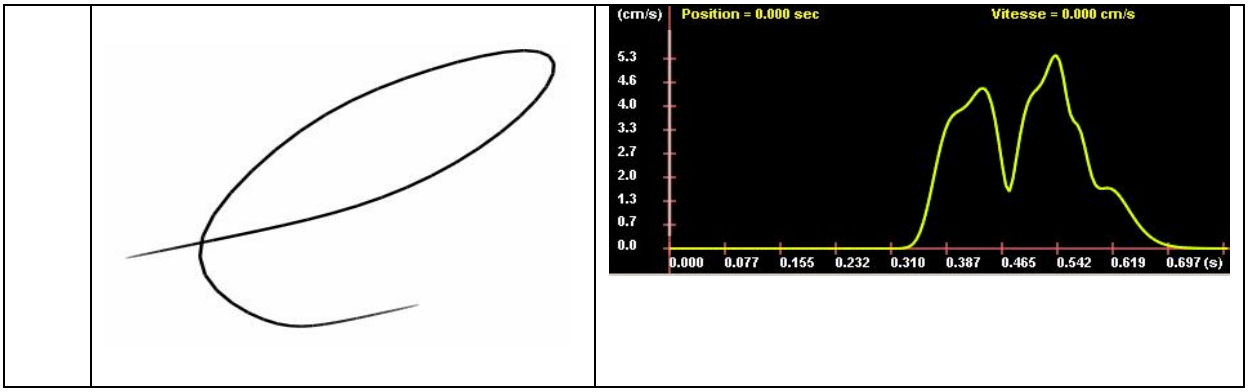


Figure.7. Traces et profils de vitesse simulés et réels d'un échantillon de l'écriture manuscrite

7. Références bibliographiques

GAMMA, E., 1995. *Design patterns : elements of reusable object-oriented software.* Reading, Massachusetts ; Don Mills, Ontario : Addison-Wesley, 1995. 395 p.

PLAMONDON, R., 1995a. «A Kinematic Theory of Rapid Human Movements: Part I: Movement Representation and Generation», **Biological Cybernetics**, vol. 72, no 4, p. 295-307.

PLAMONDON, R., 1995b. «A Kinematic Theory of Rapid Human Movements: Part II: Movement Time and Control», **Biological Cybernetics**, vol. 72, no 4, p. 309-320.

PLAMONDON, R., ALIMI, A., 1997. «Speed/Accuracy Tradeoffs in Target Directed Movements», **Behavioral and Brain Sciences**, vol. 20, no 2, p. 279-349.

PLAMONDON, R., GUERFALI, W., 1998a. « The 2/3 Power Law : When and Why? », **Acta Psychologica**, vol. 100, p. 85-96.

PLAMONDON, R., GUERFALI, W., 1998b. « The Generation of Handwriting with Delta-Lognormal Synergies », **Biological Cybernetics**, vol. 78, p. 119-132

PLAMONDON, R., DJIOUA, M., 2005. «Handwriting Stroke Trajectory Variability in the Context of the Kinematic Theory», **Proceeding of the 12th Biennal Conference of the International Graphonomics Society,** 26-29 June, 2005 , Italy , pp:250-254 .

VARGA, T., KILCHHOFER, D., BUNKE, H., 2005. «Template-based Handwriting Generation for the Training of Recognition Systems», **Proceeding of the 12th Biennal Conference of the International Graphonomics Society,** 26-29 June, 2005 , Italy , pp:206-211.

L'École Polytechnique se spécialise dans la formation d'ingénieurs et la recherche en ingénierie depuis 1873

École Polytechnique de Montréal

École affiliée à l'Université de Montréal

Campus de l'Université de Montréal
C.P. 6079, succ. Centre-ville
Montréal (Québec)
Canada H3C 3A7

www.polymtl.ca

ÉCOLE
POLYTECHNIQUE
MONTRÉAL