RAIRO-Oper. Res. 48 (2014) 509–520
DOI: 10.1051/ro/2014020

RAIRO Operations Research
www.rairo-ro.org

ON FINDING OPTIMAL PARAMETERS OF AN OSCILLATORY MODEL OF HANDWRITING

Gaëtan André[1] and Frédéric Messine[1]

Abstract. In this paper, we show how optimization methods can be used efficiently to determine the parameters of an oscillatory model of handwriting. Because these methods have to be used in real-time applications, this involves that the optimization problems must be rapidely solved. Hence, we developed an original heuristic algorithm, named FHA. This code was validated by comparing it (accuracy/CPU-times) with a multistart method based on Trust Region Reflective algorithm.

Keywords. Handwriting model, nonlinear programming, heuritic method, multistart method.

Mathematics Subject Classification. 90C90, 90C30, 90-08.

1. Introduction

The studies concerning handwriting interest many different fields of science such as for example computer science, psychology, neurology or psychomotricity. Thus, different models of handwriting are associated with those fields of science. For example, in computer science, one of the main issue is recognition which is often based on Markov or Neural Networks, see [8,9] to have some examples. In mind sciences, the understanding of the generation of handwriting as a human movement is a key question, and several models try to take it into account [7,10]. These so-called "generative models" make it possible to draw the handwriting trace of complete words and sentences by using a small number of parameters. Although it is now known that writing and reading are tightly linked in the brain: motor areas are involved in reading handwritten scripts [6]; "generative models" are

Received March 11, 2013. Accepted February 24, 2014.

[1] University of Toulouse, ENSEEIHT-IRIT, 2 rue Camichel, B.P. 7122, 31072 Toulouse Cedex 7, France. gaetan.andre@irit.fr; frederic.messine@n7.fr

Article published by EDP Sciences

barely used for recognition, [8]. However, some works have been done in that direction [4], so we would like in this paper to go further by using an oscillatory model of handwriting. The generative model we use is an improvement of the classical Hollerbach oscillatory model, [5] and our purpose is to extract its parameters as fast as possible (for application purposes). This oriented us to the development of a dedicated heuristic algorithm.

In Section 2, we present the model which will be used in this work. The formulation of this model into an optimization problem is presented in Section 3. In Section 4, we discuss how to apply standard optimization code (`lsqnonlin` of `MatLab`) to solve this nonlinear and non-convex optimization problem. Some instances are solved and the results are discussed depending on the starting points. This shows the difficulty to solve such a problem which owns many local minima. In Section 5, an original heuristic based algorithm, named FHA for Fast Harmonic Approximation, is provided. Comparisons between FHA and a multistart based local code are done and discussed in Section 6. Finally, in Section 7, we conclude.

2. An oscillatory model of handwriting

One of the first oscillatory model of handwriting was proposed by Hollerbach [5]. Hollerbach's work aimed at describing how a mass-spring model of the arm apparatus could lead to an oscillatory model of handwriting. In this work, we only focus on a model derived from a Hollerbach's handwriting one.

In this model, handwriting is viewed as the result of two superimposed oscillators. Each evolves on a distinct direction of the plane. Although any oscillator could work as well, it is more convenient to use harmonic oscillators. Moreover, that choice is more compliant with the spring muscle model. In this model, a pen position evolution is defined as follows:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = a_x \sin(\omega_x t + \phi_x), \tag{2.1}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = a_y \sin(\omega_y t + \phi_y), \tag{2.2}$$

where a_x and a_y are the horizontal and vertical velocity amplitudes; ω_x, ω_y, ϕ_x and ϕ_y are respectively the frequencies and the phases associated to these directions.

The parameters of the model (a_x, a_y, ω_x, ω_y, ϕ_x and ϕ_y) are supposed to be piecewise constant. The times where these values change, occur at horizontal (resp. vertical) zero-velocity for parameters related to the horizontal (resp. vertical) component. This is the most important change between our model and the Hollerbach's model (in the original Hollebach model, all parameters change at vertical zero-velocity points).

Our goal is to extract the parameters described in the model from real recorded strokes. In this work, a stroke is the pen trajectory leading to the formation of a

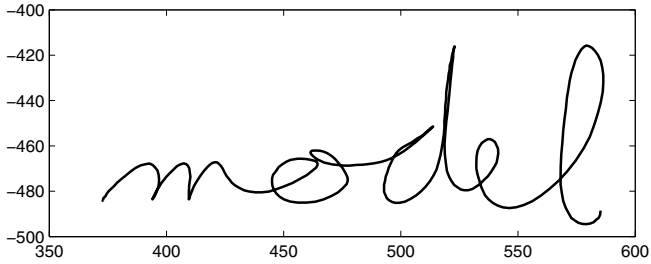


FIGURE 1. Visual example of a trace.

word. This means that for each recorded stroke, we want to be able to compute two series: $(t_{0xi}, a_{xi}, \omega_{xi}, \phi_{xi})_{1\leq i\leq N_x}$ and $(t_{0yi}, a_{y_i}, \omega_{y_i}, \phi_{y_i})_{1\leq i\leq N_y}$, where the series (t_{0x}) and respectively (t_{0y}) are the times when the horizontal (resp. vertical) velocity is null. Note that N_x and N_y are not necessarily equal.

The strokes were previously recorded using a digitizing device which is supposed to provide the pen positions at a fixed rate of 100Hz. For each stroke the following series is provided:

$$S = (t_i, x_i, y_i)_{0\leq i\leq N, N\in\mathbb{N}^*} \text{ where } \forall i > 0, t_i > t_{i-1}. \tag{2.3}$$

Because, we work on velocities, we have first to compute the derivatives of x and y by t (x and y are signals). Then, we apply a zero-crossing search algorithm on the low pass filtered derivatives in order to obtain (t_{0x}) and (t_{0y}).

All numerical examples given in the following sections are based on a set of 15 strokes available here [1]. For these examples, N is between 232 and 731, N_x is ranging from 15 to 45 and Ny from 14 to 44. In the next section, we present how to extract from the S series the piecewise constant parameters of equations (2.1) and (2.2). On Figure 1, we provide an example of a typical trace.

3. Problem formulation

From real recorder strokes, the problem is to compute $(a_{xi}, \omega_{xi}, \phi_{xi})_{1\leq i\leq N_x}$ and $(a_{y_i}, \omega_{y_i}, \phi_{y_i})_{1\leq i\leq N_y}$ by using $(t_{0xi})_{1\leq i\leq N_x}$, $(t_{0yi})_{1\leq i\leq N_y}$ and the two temporal derivatives $\frac{\mathrm{d}x}{\mathrm{d}t}$ and $\frac{\mathrm{d}y}{\mathrm{d}t}$ (calculated by using a right-sided finite difference scheme from S see Eq. (2.3)). In order to simplify the notations, we call θ the variables of our optimization problem, with

$$\theta = \left((a_{xi}, \omega_{xi}, \phi_{xi})_{1\leq i\leq N_x}, (a_{y_i}, \omega_{y_i}, \phi_{y_i})_{1\leq i\leq N_y}\right).$$

This problem can be solved by using the maximum-likelihood estimation method (using normal distribution) and it can also be equivalently formulated as a regular

curve-fitting problem. Hence, we obtain the the following formulation:

$$\underset{\theta}{\operatorname{argmin}} f(\theta) = \sum_{j=0}^{N_x-1} \sum_{i=\iota_x(j)}^{\iota_x(j+1)-1} \left(\frac{\mathrm{d}x_i}{\mathrm{d}t} - a_{xj} \sin\left(\omega_{xj} t_i + \phi_{xj}\right) \right)^2 + \sum_{j=0}^{N_y-1} \sum_{i=\iota_y(j)}^{\iota_y(j+1)-1} \left(\frac{\mathrm{d}y_i}{\mathrm{d}t} - a_{yj} \sin\left(\omega_{yj} t_i + \phi_{yj}\right) \right)^2, \tag{3.1}$$

where ι_x and ι_y are functions which return the indices of the zero-velocity points in $\frac{\mathrm{d}x}{\mathrm{d}t}$ and $\frac{\mathrm{d}y}{\mathrm{d}t}$ series.

From parameters θ, we reconstruct the derivatives $\frac{\mathrm{d}x}{\mathrm{d}t}$ and $\frac{\mathrm{d}y}{\mathrm{d}t}$ ($\frac{\mathrm{d}x}{\mathrm{d}t}$ and $\frac{\mathrm{d}y}{\mathrm{d}t}$ are column vectors of size N). This *synthesized* solution, that we call $\mathcal{C}_s = \left(\frac{\mathrm{d}x}{\mathrm{d}t} \ \frac{\mathrm{d}y}{\mathrm{d}t}\right)$, is based on the same sample of time as the derivatives calculated from the recorded stroke $\mathcal{C}_o$. An error measurement between the original stroke and the synthesized solution is given by:

$$\operatorname{err}(\mathcal{C}_s) = \frac{\|\mathcal{C}_s - \mathcal{C}_o\|_2}{\|\mathcal{C}_o\|_2}. \tag{3.2}$$

Remark that solving Problem (3.1) is equivalent to find $\mathcal{C}_s$ which minimizes $\operatorname{err}(\mathcal{C}_s)$.

In our application, errors below 0.5 can be considered good enough.

4. Applying a usual non-linear algorithm

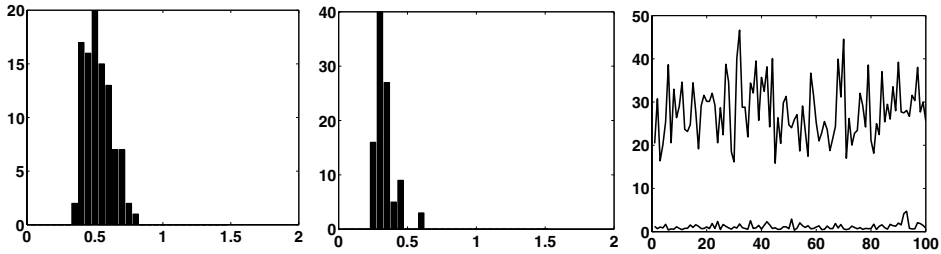
Because Problem (3.1) is a least square problem, we choose to apply the Trust-Region-Reflective algorithm which is a subspace trust-region method. This algorithm is based on the interior-reflective Newton method as described in [2,3]. In this work, the `MatLab` implementation of this algorithm (named `lsqnonlin`) is used.

Different strategies can be used to apply this algorithm. The purpose of the next subsections is to determine the most efficient one.

4.1. To split or not to split?

Some variables of Problem (3.1) are independent. This is due to the fact that our model of handwriting is a piecewise parametrization, each piece being independent to the others. Moreover, in our model of handwriting, there is no link between what happens on horizontal and vertical directions. Therefore, parameters concerning x and y also are independent and then, Problem (3.1) can be divided as a set $\mathcal{S}$ of sub-problems:

$$\mathcal{S} = \left\{ \sum_{i=\iota_d(j)}^{\iota_d(j+1)-1} (d_i - a_{dj} \sin(\omega_{dj} t_i + \phi_{dj}))^2 : d \in \{(x),(y)\}, j \in [\![0, N_d - 1]\!] \right\}. \tag{4.1}$$



(a) (b) (c)

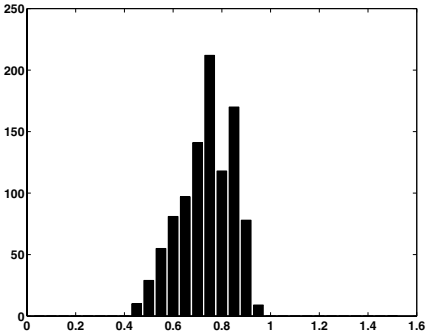
FIGURE 2. Comparison between split and non-split methods: (A) Error distribution of 100 runs of the split method; (B) Error distribution of 100 runs of the non-split method; (C) Computation time (in seconds) over 100 runs where the top curve represents split method and the bottom curve for the non-split one.

Instead of applying the algorithm to the whole problem, we might be tempted to apply it on each element of $\mathcal{S}$. In Figure 2, comparisons between the split and non-split methods are performed for a unique stroke but using 100 different starting points (they are the same ones for the two strategies); moreover all other parameters are equal. In the non-split case (2(b)) the errors are distributed below 0.5 in most of the cases, with a mean of 0.33. The CPU-time to solve one instance is about one second (2(c)). In the case of the split method (2(a)), errors are distributed with a higher dispersion (from 0.4 to 1.7) around a mean of 0.52. The CPU-time for one instance, see (2(c)), is about 28 seconds. This is almost 30 times higher than for the non-split method. Hence, in the following, we only consider the non-split method.

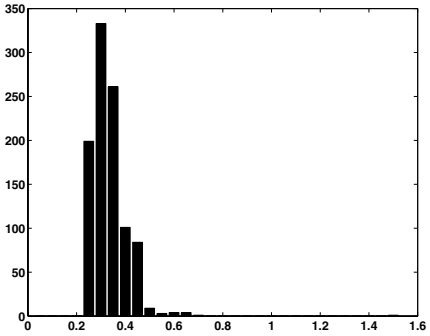
4.2. STARTING POINTS

Solving Problem 3.1, the starting point which initiates the local search algorithm, has a big impact on its convergence to a local solution. The importance of the choice of this starting point leads us to try different strategies. Thus we develop the following three strategies: (i) we can randomly generate these starting points according to a uniform distribution on our working domain; (ii) we can use starting points generated randomly according to normal laws whose parameters can be determined statistically; (iii) we can start from the point defined by the top of these normal laws (*i.e.*, the mean of the distribution of the statistically studied parameters). Note that in this case, the optimization code is run just one time.

Statistical estimations of the parameters of the normal laws used in the second strategy are based on the search of the zero-crossings findings. In order to improve the results, this estimation is done on each stroke. On each velocity profile we compute the zero crossings, that divide the signal in semi-periods. For each demi period, we can compute an amplitude by maximizing the profile on this



(a) Uniformly distributed starting points



(b) Normally distributed starting points

FIGURE 3. Error distribution for 1000 runs of the optimization algorithm on two types of starting point distribution; the errors are on the abscices, and the number of runs by interval error are on the ordinates.

semi-period. Frequency can easily be approximated from the same semi-period. We take as parameters of the normal laws associated with frequencies and amplitudes, the mean and standard deviation of the frequencies and amplitude so computed for all semi-periods of each profiles. The phases are supposed to follow a uniform law.

We run the algorithm 1000 times, on one stroke, for the first two cases (i) and (ii) and we present in Figure 3 the error distribution of these runs. Error distribution in the case (i) has a mean of 0.7 and has a higher dispertion than in the case (ii) where the mean is 0.3. The results concerning the third case (in which only one starting point is considered), provides an error value of 0.8. Thus, the error distribution is clearly much lower when the starting point is randomly chosen according to a normal distribution determined by a statistical study.

4.3. How many runs ?

In the previous subsections, it was shown that the best way to apply the Trust Region Reflective algorithm to the problem was to consider it as a single entire optimization problem (see, Eq. (3.1)) and then to run it a lot of times with different starting points randomly chosen according to a normal distributions. Now, a question remains: how many runs of this optimization algorithm do we need to reach a good enough solution?

Suppose that 1000 runs always provide approximated solutions good enough for our application. Let s denotes the best solution coming from these 1000 runs and l_i the local solution obtained using the ith starting point. Hence, we are interested by the following optimization problem:

$$\begin{cases} \min_{n\in\{1,\cdots,1000\}} n \\ \text{uc} \qquad\qquad\qquad \|err(s) - err(l_n)\| < \epsilon \end{cases}$$

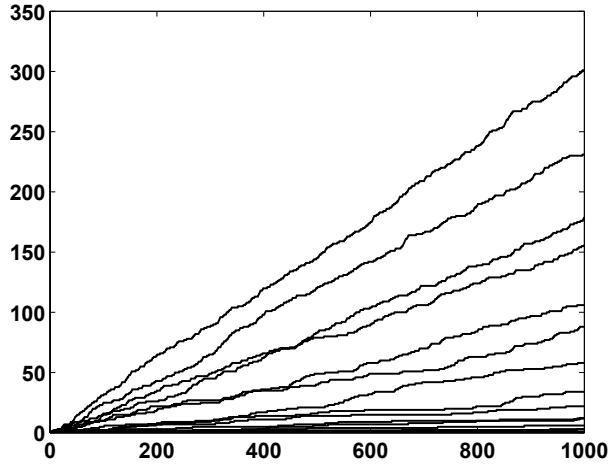


FIGURE 4. Number of accumulated encountered good approximations (satisfying the constraint) in function of the number of runs, for 15 different strokes (with $\epsilon = 0.01$).

This problem is relevant because, as we can see in Figure 4, the best found local solutions seem to be almost equals and uniformly distributed over the number of runs.

Suppose that at each run of the algorithm, we obtain a probability p to obtain such an approximation. Denote n the number of times the algorithm is run in order to have a probability q to obtain at least one good enough solution which satisfies the constraint. Thus, the condition $1-(1-p)^n > q$ must be satisfied and this yields:

$$n \geqslant \left\lceil \frac{\ln(1-q)}{\ln(1-p)} \right\rceil . \tag{4.2}$$

Considering Figure 4, even if the hypothesis that all the good approximations seem to be uniformly distributed over the number of runs, the probability p seems to be very different from one trace to another one: a stroke has a great influence on the value of p. Therefore, we unfortunately cannot accurately determine n beforehand. Moreover, an overestimation of n cannot also be determined.

4.4. Conclusion

In this section, we study different possibilities to apply in an efficient way the Trust-Region method in order to solve Problem 3.1. Thus, this algorithm has to be applied on the whole problem (because the idea to split Problem (3.1) provides bad results). Furthermore, note that the shape of the distribution of the starting points is very important and that the number of runs needed to reach a good local solution cannot be determined *a priori*.

In the following sections, the time comparisons in Section 6, the CPU-time comparisons will be done for a fixed number of runs, denoted N. The statistical study (to determine normal distributions) is supposed to add no overhead as far as time is concerned. The algorithm and the way of applying it will be referred as MS_N (Multi-Start method using N starting points).

5. The FHA heuristic method

In this section, we present an original heuristic based algorithm named FHA (for Fast Harmonic Approximation). FHA is used to find an approximated solution of Problem (3.1). This algorithm has been implemented in `MatLab`.

This section is divided into two parts. In a first subsection, we present a mathematical result enabling to quickly approximate the amplitude of a harmonic function, and, in the second part, we describe the algorithm itself.

5.1. Semi-period wise estimation of the amplitude of a sine function

Consider the following function:

$$f : x \mapsto a\sin(\omega x + \phi), \tag{5.1}$$

where a, ω and ϕ are independent to x. We can calculate the mean and the variance of f between two successive zeros:

$$M = \frac{\omega}{\pi}\int_{\frac{-\phi}{\omega}}^{\frac{\pi-\phi}{\omega}} f(x)\mathrm{d}x = \frac{2a}{\pi}, \tag{5.2}$$

$$V = \frac{\omega}{\pi}\int_{\frac{-\phi}{\omega}}^{\frac{\pi-\phi}{\omega}} (f(x)-M)^2\mathrm{d}x = \frac{a^2\left(-8+\pi^2\right)}{2\pi^2}. \tag{5.3}$$

This shows that the amplitude of a sinusoidal signal can be approximated both the sum or the variance of this signal on a semi-period (zero to zero) independently to frequencies and phases.

5.2. FHA Algorithm

5.2.1. Evaluating frequencies and phases

Between two zeros, we know from the model described in Section 2, that the parameters a, ω_x and ϕ_x are constant. Now, we show that these values can be easily computed. Let t_1 and t_2 be the times of the two subsequent zeros. We have these two following equations:

$$\omega_x(t_2 - t_1) = \pi, \tag{5.4}$$

$$\omega_x t_1 + \phi_x = 0. \tag{5.5}$$

From equation (5.4), we have ω_x and from equation (5.5), we obtain ϕ_x. It provides a very simple way to obtain those parameters, but the main drawback is that their accuracy highly depends on how we find the zero values.

5.2.2. Evaluating amplitude

The final step is to estimate the amplitude velocity between two consecutive zero-values. Denote t_i and t_{i+1} the times corresponding to two subsequent zero-values. Let us define the arc A as the part of the derivative signal between t_i and t_{i+1}:

$$A = \left(\frac{\mathrm{d}x}{\mathrm{d}t}\right) \text{ between } t_i \text{ and } t_{i+1}, \tag{5.6}$$

we approximate a thanks to results described in Section 5.1:

$$a = \frac{\pi}{2}\mathrm{mean}(A). \tag{5.7}$$

6. Comparison between MS_{50}, MS_{1000} and FHA

In this section the methods presented in Sections 4 and 5 will be compared, regarding their precisions and computational times. Both algorithms have been run on 15 traces which can be found in [1]. In order to have a better estimation of the performance of MS_{50} and MS_{1000}, they have been run 10 times for each strokes. Figure 5 presents the errors obtained for each methods. In the case of MS_{50} and MS_{1000}, best fit, worst fit and mean fit are given to get a insight of the variability of these methods.

Due to the fact that all the starting points are chosen randomly, MS_{50} shows a big variability in its efficiency. However, the general behavior of the method is the same over the 15 strokes. First, considering the error, FHA provides correct and regular results which are always below an error of 0.5. Contrarily, on 9 cases over 15, MS_{50} provide a mean error above 0.5. This shows that MS_{50} can miss interesting local solutions. Concerning now the CPU-time, FHA is constant while MS_{50} highly depends on the starting points. In most cases, our heuristic method FHA is from 50 to 1000 time faster than MS_{50}.

Same results and comments which validates the efficiency of our heuristic method FHA were also obtained if we choose 1000 starting points instead of 50, see Figure 5. The accuracy of the results obtained with MS_{1000} is not so improved comparing to the one reached with MS_{50} while CPU-times increase a lot. Moreover, the fact that on some strokes MS_{50} seems to perform better that MS_{1000} (strokes 4, 6, 9 and 10) indicates that there is a high variability in results that even 10 runs cannot avoid. These strokes (4, 6, 9 and 10) are also associated with the worst results obtained by both methods. In all other cases, the results are better for MS_{1000}, even if MS_{50} and MS_{1000} seem to perform equally. Although for each stroke, the minimum error for MS_{1000} is lower than the minimum error obtained with MS_{50}. Note that when the starting points are not inefficient, this provides a very low convergence of these local optimization based algorithms.

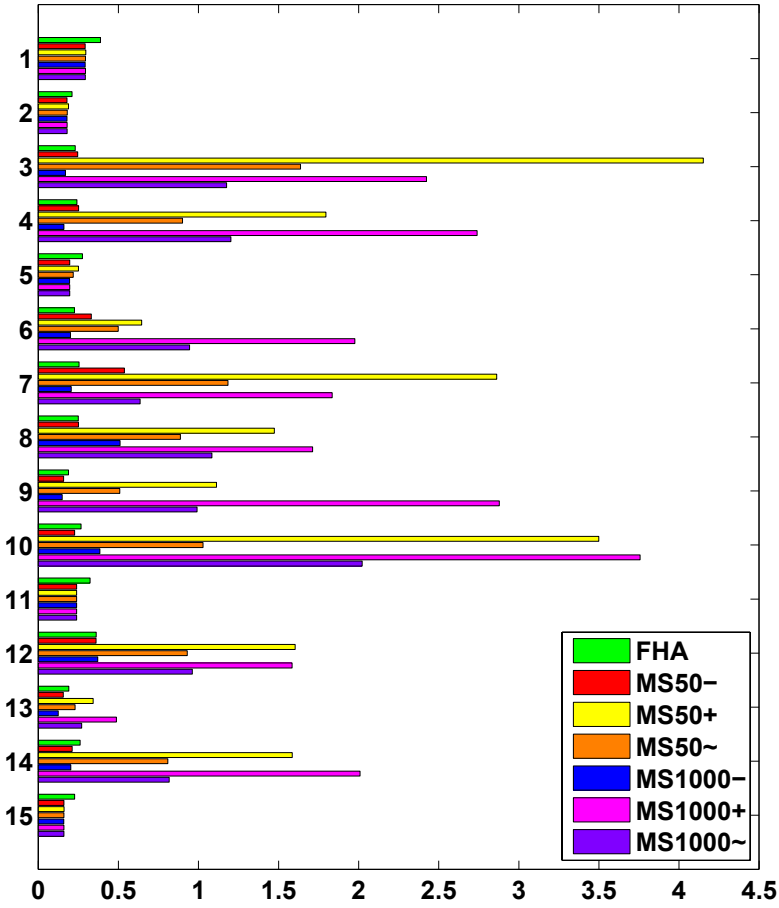
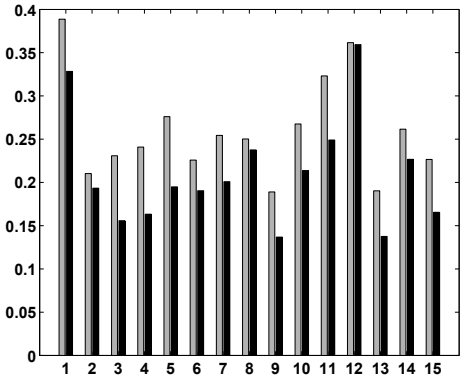


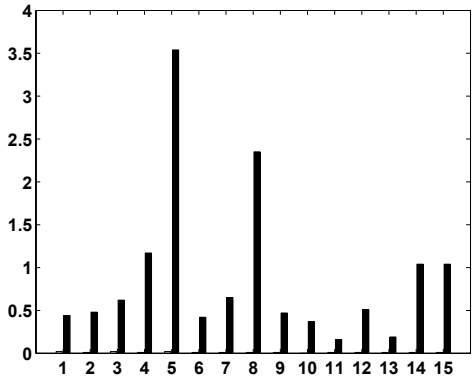
FIGURE 5. Error of the solutions found by FHA, MS_{50} and MS_{1000}. The two latters where run 10 times: for each stroke, we give the worst solution (denoted with a +), the best (-) and the mean ( ).

Remark 6.1. The reason why our heuristic algorithm FHA works so well, is because it is directly relied to the property of our model where the parameters change at zero velocity points. This makes if possible to simplify and quickly compute frequencies and phases.

In Figure 6, numerical results corresponding to the use as a starting point of the solution previously obtained by the FHA algorithm are reported. We remark that these new optimal solutions from the FHAo method (for FHA which is optimized) are more accurate than FHA. However, the corresponding CPU-times are much more important.



(a) Comparison of minimal values between FHA (in grey) and FHAo (in black)



(b) Time comparison between the two methods (in seconds). For FHA, CPU-time are between 0.01s and 0.03s

FIGURE 6. Comparison between FHA in grey and FHAo in black (solution of FHA taking as a starting point)

7. Conclusion

In this work, we presented an efficient heuristic based algorithm for solving a particular non-linear optimization problem dedicated to the modeling of handwriting. Comparing to a multistart local optimization based method, it appears that the heuristic FHA algorithm is faster and gives more regular and efficient results. This optimization problem has many local optima and even a multistart algorithm with 1000 starting points owns some difficulties providing a good local minimum (see Fig. 5). This demonstrates the benefits of our FHA algorithm which can also be used to construct efficient starting point for a local optimization code.

References

[1] G. André, `www.irit.fr/~Gaetan.Andre/publications.php`.
[2] T.F. Coleman and Y. Li, An interior trust region approach for nonlinear minimization subject to bounds. *SIAM J. Opt.* **6** (1993) 418–445.
[3] T.F. Coleman and Y. Li, On the convergence of interior-reflective Newton methods for nonlinear minimization subject to bounds. *Math. Program.* **67** (1994) 189–224.
[4] E. Gilet, Modélisation bayésienne d'une boucle de perception action : Application a l'écriture (Bayesian Modelisation of a sensori-motor loop: application to reading and handwriting). Thesis, Joseph, Fourier University, Grenoble, France (2009).
[5] J.M. Hollerbach, An oscillatory theory of handwriting. *Biol. Cybern.* **156** (1981) 139–156.
[6] M. Longcamp *et al.*, The imprint of action: motor cortex involvement in visual perception of handwritten letters. *NeuroImage* **23** (2006) 681–688.
[7] R. Plamondon *et al.*, Modelling velocity profiles of rapid movements: a comparative study. *Biol. Cybern.* **69** (1993) 119–128.

[8] R. Plamondon, On-Line and Off-Line, Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **22** (2000) 63–84.
[9] T. Plötz and G. Fink, Markov models for offline handwriting recognition: a survey. *Int. J. Doc. Anal. Recogn. (IJDAR)* **12** (2009) 169–298.
[10] P. Viviani and T. Flash, Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. *J. Exp. Psychol. Hum. Percept. Perform.* **21** (1995) 32–53.