Inverse problems of immunology: within-host HIV dynamics

adele.mortier

August 2015

1 Introduction

The human immunodeficiency virus (HIV), by causing a progressive failure of the human immune system, allows life-threatening opportunistic infections and cancers to thrive. Within blood, HIV is present as both free virus particles and virus within infected immune cells, specifically CD4+ T cells.

HIV infection leads to low levels of CD4+ T cells through a number of mechanisms, including apoptosis of uninfected bystander cells, direct viral killing of infected cells, and killing of infected CD4+ T cells by CD8 cytotoxic lymphocytes that recognize infected cells.

The research in the within-host dynamics of HIV infections focuses on theses mechanisms, and especially the spread of the virus *in vivo*. The latest models take into account virus's cell-to-cell interactions, T cell activation, and immune exhaustion [5]. However, a simpler model involving ordinary non-linear equations also provides satisfying results for early times, with a limited parameter set.

Since it remains quite easy to solve, it lends itself well to the study of HIV inverse problem. With this basic model and regular data of viremia and T cells loads over time after infection, we propose an algorithm which characterizes individual parameters in HIV modeling. The knowledge of these parameters could open the way to more personalized (and more efficient) treatments. In a first section, the differents parameters and the model of the problem are presented in order to justify the inverse approach.Then, in a second section, we will explain the implementation of the solving program. Thirdly, some results and plots of the outputs are shown and analysed, which leads to a concluding paragraph reviewing the global performance of the approach and the accuracy of the approximate solution. Finally, we will consider future perspectives which could enhance the model or the solving program.

2 Statement of the problem

A basic model of HIV dynamics involves only three determining variables: uninfected and infected target cells (mostly CD4+ T cell), and free HIV viruses within blood . Their concentrations are respectively noted *T(t)*, *I(t)*, *V(t)*. We can also build the vector quantity *X(t)*, which characterizes the whole immune response to the infection:

$$X(t) = \begin{bmatrix} T(t) \\ I(t) \\ V(t) \end{bmatrix} \tag{1}$$

In practice, it is relatively easy to measure this vector variable, and thus access to the solution of the direct problem of HIV dynamics with an acceptable level of precision: we simply need daily blood samples from HIV-positive patients.

Besides, interactions between the three blood concentrations *T(t)*, *I(t)*, *V(t)* can be modeled quite simply by a *SIR model with vital dynamics and constant population*[8][13][1]. It is a basic compartmental model that involves *Susceptible cells I(t)*, *I infectious virions V(t)*, and *Recovered immune cells*

T(t):

$$\frac{dX}{dt} = \begin{cases} \frac{dT}{dt} &= \lambda - d_T T - \beta V T \\ \frac{dI}{dt} &= \beta V T - \delta I \\ \frac{dV}{dt} &= pI - cV \end{cases} \tag{2}$$

Where:

- λ and d_T are respectively the birth rate and the dying rate of target cells;
- β quantifies the infection by free virus;
- δ is the dying rate of infected cells;
- p and c are respectively the production rate and the clearing rate of the virions.

These six coefficients may vary according to the metabolism of each individual. But contrary to the variable *X(t)*, they remain quite difficult to measure experimentally. That is the reason why tackling the HIV dynamics modeling with an inverse approach seems appropriate. So let q^{ex} be the 6-dimensional vector that contains all information about the immune features of a given individual:

$$q^{ex} = \begin{bmatrix} \lambda & d_T & \beta & \delta & p & c \end{bmatrix}^T \tag{3}$$

Parameter	Minimum	Mean	Maximum	Units
λ	4.3×10^{-2}	1.089×10^{-1}	2×10^{-1}	$\mu L^{-1}\ day^{-1}$
d_T	4.3×10^{-3}	1.089×10^{-2}	2×10^{-2}	day^{-1}
β	1.9×10^{-4}	1.179×10^{-3}	4.8×10^{-3}	$\mu L\ day^{-1}$
δ	1.3×10^{-1}	3.660×10^{-1}	8×10^{-1}	day^{-1}
p	9.8×10^{1}	1.427×10^{3}	7.1×10^{3}	day^{-1}
c	3	3	3	day^{-1}

Table 1: Extreme and mean values of immune coefficients [7]

The exact solution q^{ex} will be the unknown element of the inverse problem; its components shall be in the ranges specified above. The goal of the inverse problem modeling is to approximate these coefficients only by knowing the behaviour of the function $X^{q_{ex}}(t)$ through blood samples of a given individual (*i.e.*, a given q_{ex}) in a given time interval. These baseline data can be written as follows:

$$\left\{X^{q_{ex}}(t_k)\right\}_{t_k \in [t_0, t_f)}.$$

Henceforth, we will note $\left\{X^{q_{ex}}(t_k)\right\}_{t_k \in [t_0, t_f)}$ the sampled solution of (**??**) for a given q. A realistic time sampling cannot exceed for instance 5 measurements a day (exhaustion caused by blood taking, equipments required...). A frequency of 2 blood samples per day during 5 days was chosen to solve the inverse problem. As a result, sampled times are:

$$\left\{t_k\right\}_{k \in [0,9]}, \forall k \in [0, 9], t_k = {}^{k}/_{2}. \tag{4}$$

As for the *SIR* model, it allows a relatively easy solving of the direct problem for all q we may consider. Moreover, we can assume that the solution q^{ex} of the inverse problem is the value of q which minimizes the following misfit function, based on the *method of Least Squares*:

$$J : q \in \mathbb{R}^6 \longmapsto \sum_{t_k \in [t_0, t_f)} |X^q(t_k) - X^{ex}(t_k)|^2. \tag{5}$$

Consequently, the HIV dynamics inverse problem can be reduced to a problem of optimization and ordinary differential equation (*ODE*) solving. The first step will be carried out through the *Nelder-Mead method*[9], while a fifth-order *Runge-Kutta method* [10] will handle the second step. We will now explain in details the running of these two methods and their implementation.

3 Structure of the program

3.1 Relevant parameters and main functions

The inverse problem solving program is implemented using C/C++ language. It is based on three major functions. Vector parameters like q or *X(t)* were stored using the C++ `std::vector` library. To run the whole program, it is necessary to set several input arguments, which are:

- `tspan`: the time interval $[t_0, t_f)$ for the measurements of *X(t)*;
- `h`: the time sampling frequency, so that $t_0 \leq t_k = k \times h < t_f$;
- `xsynth`: the sampled values of T-cells and virions concentrations $\{X^{q_{ex}}(t_k)\}_{t_k \in [t_0, t_f)}$;
- `X0`: first element of the foregoing array;
- `points`: an arbitrary set of 7 realistic q values (see Table 1 page 2);
- `q`: an element of the foregoing array;
- `simplextol`: ending condition for the *Nelder-Mead method*;
- `odetol`: ending condition for the *Runge-Kutta method*;
- `hmax`: maximum time increment in the *Runge-Kutta method*;

Parameter	Type	Preset Value
`tspan`	`vector<double>`	[0, 5)
`h`	`double`	0.5
`xsynth`	`vector<vector<double>>`	Given by the solving of direct problem using q_{ex}
`X0`	`vector<double>`	$[10^3\ 0\ 10^{-3}]$
`points`	`vector<vector<double>>`	Generated by a random law
`simplextol`	`double`	10^{-3}
`odetol`	`double`	10
`hmax`	`double`	5×10^{-3}

Table 2: Input parameters and their preset values

All these parameters should be passed to the three following functions:

- `simplex(tspan, h, xsynth, points, simplextol, odetol, hmax)`
  launches the *Nelder-Mead method*.
  It should return the minimum (local or global) of the misfit function *J(q)*, in the form of a `vector`.

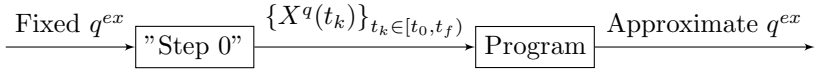
- `J(tspan, h, xsynth, q, odetol, hmax)` computes the misfit function on the basis of the *Least-Squares method* for a given q.
  Its output value should converge toward 0 by definition.

- `odesolver(q, tspan, X0, odetol, hmax)` launches a fifth-order *Runge-Kutta method* using *Dormand-Prince* coefficients[10].
  It should return the sampled function $\{X^q(t_k)\}_{t_k \in [t_0, t_f)}$ in the form of an array of `vector`.

3.2 Step-by-step processing

3.2.1 Generation of synthetic data

In practice, the blood samples $\{X^{q_{ex}}(t_k)\}_{t_k \in [t_0, t_f)}$ come from experimental measurements (which can, by the way, be erroneous in a certain extent). But here, synthetic exact data have to be generated from a fixed q^{ex}, in order to check the good convergence of the algorithm, which should be able to recalculate this value by itself.
That is why the whole program includes a "Step 0", which consists in the solving of the direct problem for a fixed q^{ex}. Afterwards, the value of q^{ex} should not be reused by the program.



3.2.2 Direct problem solving

The solving of the direct problem relies on a *Runge-Kutta method* for non-stiff ordinary differential equations, inspired by the Matlab solver `ode45.m` [6], which gave satisfying results in other studies[18]. The default method of `ode45.m` is the Dormand-Prince [10] embedded method, designed to produce an estimate of the local truncation error of a single *Runge-Kutta* step. It allows to control the error with adaptive stepsize [16] by managing two methods, one with order 5 (using the b_i^5 coefficients below) and one with order 4 (using the b_i^4).

c_i	a_{ij}						b_i^5	b_i^4
0							$\frac{35}{384}$	$\frac{5179}{57600}$
$\frac{1}{5}$	$\frac{1}{5}$						0	0
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					$\frac{500}{1113}$	$\frac{7571}{196695}$
$\frac{4}{5}$	$\frac{44}{45}$	$\frac{-56}{15}$	$\frac{32}{9}$				$\frac{125}{192}$	$\frac{393}{640}$
$\frac{8}{9}$	$\frac{19372}{6561}$	$\frac{-25360}{2187}$	$\frac{64448}{6561}$	$\frac{-212}{729}$			$\frac{-2187}{6784}$	$\frac{-92097}{339200}$
1	$\frac{9017}{3168}$	$\frac{-355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$\frac{-5103}{18656}$		$\frac{11}{84}$	$\frac{187}{2100}$
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$\frac{-2187}{6784}$	$\frac{11}{84}$	0	$\frac{1}{40}$

Table 3: Dormand-Prince coefficients

Let us consider the HIV dynamics' differential system as follows:

$$\frac{dX}{dt} = f(X). \tag{6}$$

The function f could have depended on time t, but it is not the case in the simple *SIR* model (no delays, constant coefficients). So, we can obtain the sampled function *X(t)* by successive approximations:

$$X_{n+1} = X_n + h_n \sum_{i\in[0,6]} b_i^4 k_i. \tag{7}$$

Where:

$$\forall i \in [0,6],\ k_i = f(X_n + h_n \sum_{j\in[0,6]} a_{ij}k_j),\ k_i \in \mathbb{R}^3. \tag{8}$$

This method also computes an error at each step, which is:

$$e_{n+1} = h_n \sum_{i\in[0,6]} (b_i^5 - b_i^4)k_i,\ e_{n+1} \in \mathbb{R}^3. \tag{9}$$

We assume that the step is validated if:

$$\begin{cases} ||e_{n+1}||_\infty \leq \epsilon \cdot \max(||X_n||_\infty, 1) \\ \epsilon = \texttt{odetol} \end{cases} \tag{10}$$

And then, we can update the time increment's value:

$$h_{n+1} = min\left(h_{max},\ 0.8 \cdot h_n \cdot \left(\frac{\epsilon \times max(||X_n||_\infty, 1)}{||e_{n+1}||_\infty}\right)^{\frac{1}{6}}\right). \tag{11}$$

In the program, the *Runge-Kutta method* shall be used at "Step 0" so as to obtain the synthetic data, and also whenever the function *J(q)* has to be computed for a given q; what occurs at several steps during the *Nelder-Mead method* for function minimization.

3.2.3 Minimization of the misfit function

By definition, the solution of the inverse problem is the only q for which *J(q) = 0*. And since J is a positive function:

$$J(q) = 0 \iff q = \text{argmin}(J). \tag{12}$$

To find the minimum of the misfit function, the *Nelder–Mead method* was implemented for a 6-dimensional space. The method uses the concept of a simplex, which is a special polytope. In our case, it is made up of 7 different q. The data structure is an array of `vector` called `points`. At each steps, `points` is updated using one of these four transformations:

$$q_h = \begin{cases} q^* = \bar{q} + \alpha \cdot \overrightarrow{q_h\bar{q}} \text{ (reflection)} & or\ \forall q \in \texttt{points} \\ q^{**} = \bar{q} - \beta \cdot \overrightarrow{q_h\bar{q}} \text{ (contraction)} & q = q + \frac{1}{2} \cdot \overrightarrow{qq_l} \\ q^{***} = \bar{q} - \gamma \cdot \overrightarrow{q^*\bar{q}} \text{ (expansion)} & \text{(reduction)} \end{cases} \tag{13}$$

Where:

$$\begin{cases} q_h = \text{argmax}_{q\in\texttt{points}}(J(q)) \\ q_l = \text{argmin}_{q\in\texttt{points}}(J(q)) \\ \bar{q} = \text{centroid}_{q\in\texttt{points}\setminus\{q_h\}}(q) \end{cases} \quad \begin{cases} \alpha = 1 \\ \beta = 1/2 \\ \gamma = 2 \end{cases}$$

However, the *Nelder–Mead method* is a heuristic search method that can converge to non-stationary points, i.e., local minima. In this case, the function `simplex` returns a wrong solution. This is all the more probable as the initial random set `points` is dispersed around the expected solution. That is why, without the use of an auxiliary method, the initial random generation of the array `points` remains a key issue.

Besides, the *Nelder–Mead method* does not take into account the unrealistic character of the q it returns. Indeed, the standard stopping condition of this algorithm is:

$$\begin{cases} \sqrt{\sum_{q\in\texttt{points}} |J(q) - J(\bar{q})|^2} \leq \varepsilon \\ \varepsilon = \texttt{simplextol} \end{cases} \tag{14}$$

It only quantifies how close to each other are the misfit function's values at the vertices of the simplex. Thus, a subsidiary method should check if the found minimum belongs to Table 1, and if it actually makes the function J converge towards 0. If these two conditions are actually fulfilled, the program can stop. If not, the *Nelder–Mead method* has to be launched once again, with an other random set `points`.

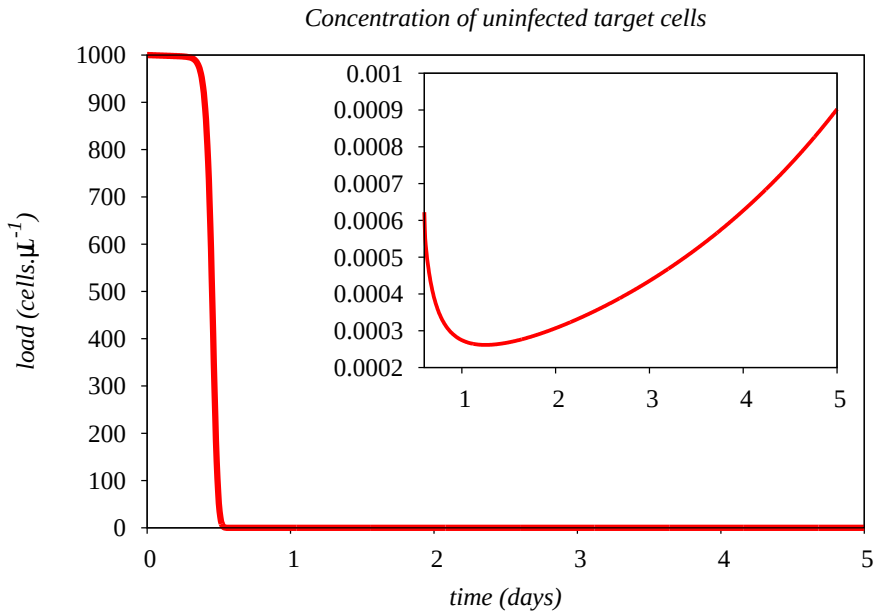
4 Results

4.1 Initial settings

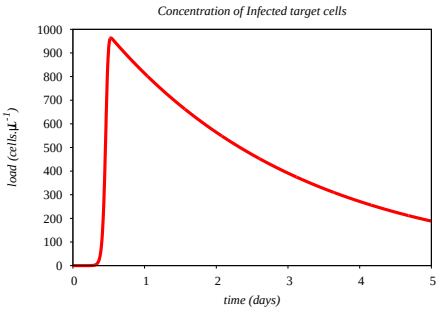
The whole program was launched with the input values of Table 2 (page 3. The value of the exact solution q_{ex} used to generate synthetic data was equal to the mean values of Table 1.

$$q^{ex} = \begin{bmatrix} 1.089 \times 10^{-1} \\ 1.089 \times 10^{-2} \\ 1.179 \times 10^{-3} \\ 3.660 \times 10^{-1} \\ 1.427 \times 10^{3} \\ 3 \end{bmatrix}$$

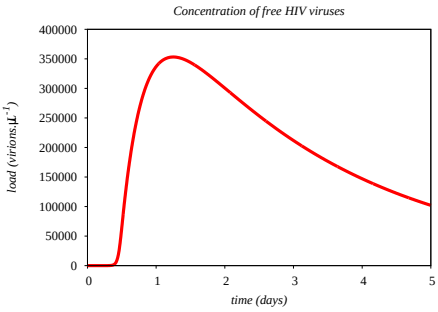
The synthetic data returned by `odesolver` could also be plotted using gnuplot:



(a) Uninfected target cells $T^{q_{ex}}(t)$



(b) Infected target cells $I^{q_{ex}}(t)$



(c) Free HIV viruses $V^{q_{ex}}(t)$

Figure 1: Plots of synthetic data generated with the set of coefficients q^{ex}

As for the generation of the set `points`, a uniform random distribution over an adjustable interval around the expected solution was chosen as a first approach. Considering each coefficient, this interval represents a given percentage of the total bandwidth between extrema of Table 1 (page 2). For a given coefficient, e.g. λ, it can thus be written:

$$[\lambda^{rel}_{min}, \lambda^{rel}_{max}] \text{ with } \begin{cases} \lambda^{rel}_{min} = \lambda^{ex} - \sigma \cdot (\lambda^{ex} - \lambda^{abs}_{min}) \\ \lambda^{rel}_{max} = \lambda^{ex} + \sigma \cdot (\lambda^{abs}_{max} - \lambda^{ex}) \end{cases} \tag{15}$$

Two series of results were computed, the first with $\sigma = 1/3$, the second with $\sigma = 2/3$.

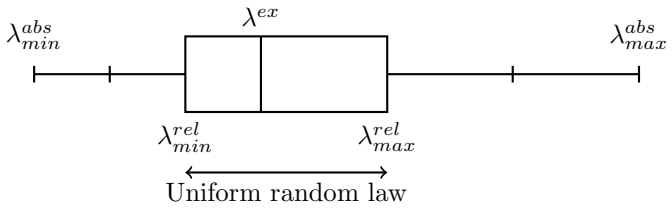


Figure 2: Adjustable interval with $\sigma = 1/3$

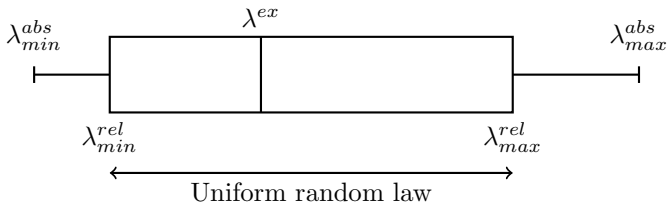
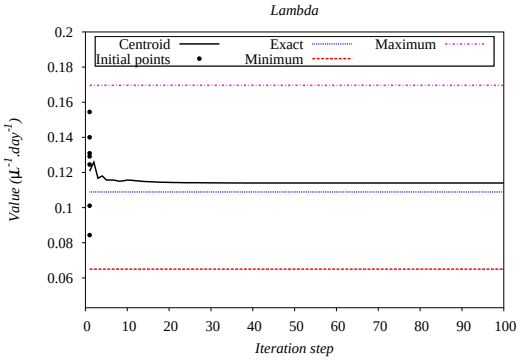


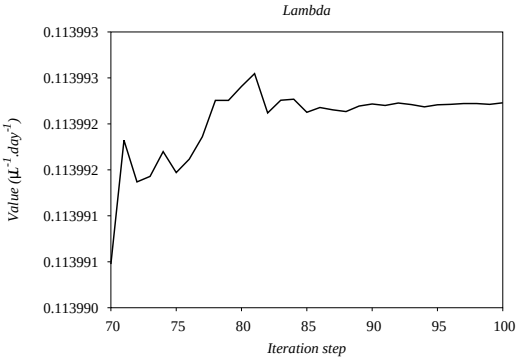
Figure 3: Adjustable interval with $\sigma = 2/3$

4.2 Convergence in the Nelder-Mead algorithm

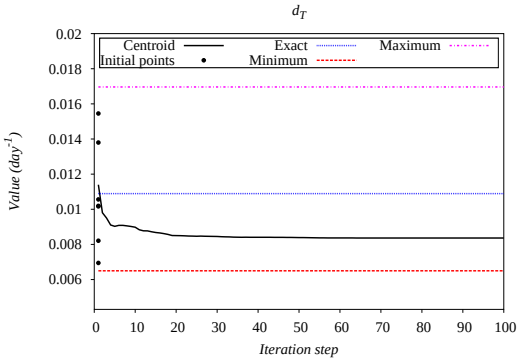
The *Nelder-Mead method* does not always return a global minimum for the function J, and may be launched several times before giving a relevant solution of the inverse problem. In this section, we focus on the results of the *Nelder-Mead method* when it has been run successfully, so as to estimate the accuracy of the convergence. The behaviours of eight random set were analysed step-by-step during the *Nelder-Mead method*, by plotting the changing coordinates of the simplex's centroid. Three sets used initial random points from a "$1/3$-adjusted interval", while the five other sets were constructed on the basis of a "$2/3$-adjusted interval". Here are the main results for one of these foregoing sets:



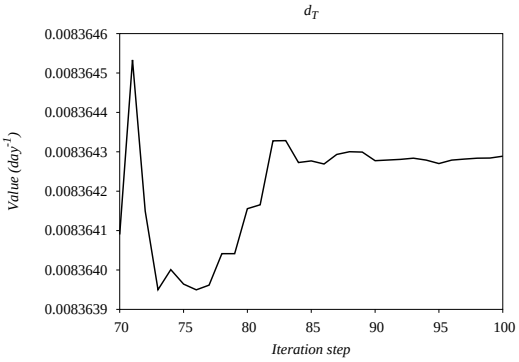
(a) λ - global convergence



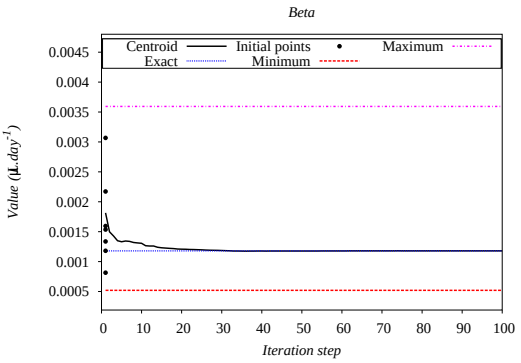
(b) λ - later behaviour



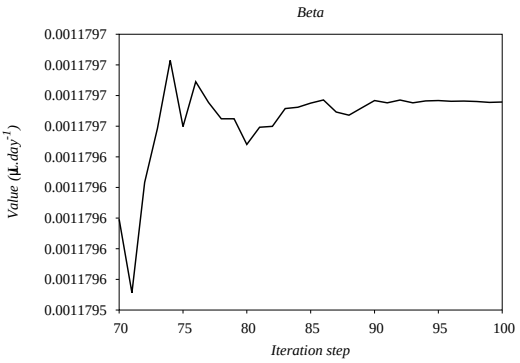
(c) d_T - global convergence



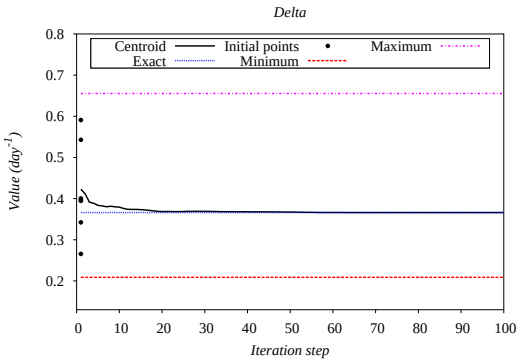
(d) d_T - later behaviour



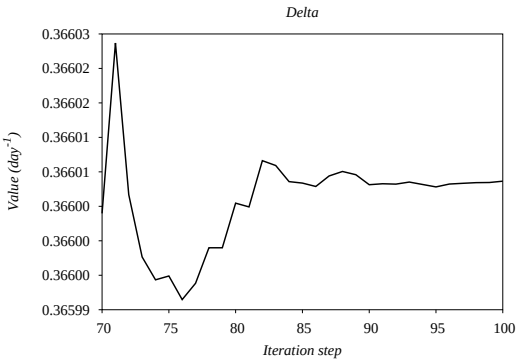
(e) β - global convergence



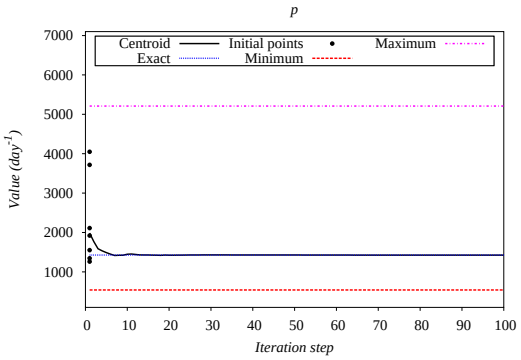
(f) β - later behaviour



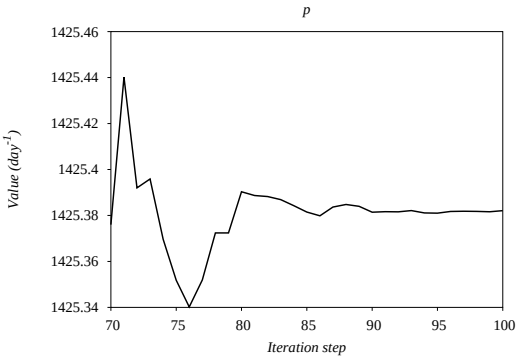
(g) δ - global convergence



(h) δ - later behaviour



(i) p - global convergence



(j) p - later behaviour

Figure 3: Convergence of the different coefficients using an initial "2/3-adjusted interval"

	λ	d_T	β
q_0	0.154463741976175	0.0137987693919694	0.00306756851914025
q_1	0.124552591530096	0.0101646476638082	0.00217212147384071
q_2	0.129165118360953	0.0154521238848026	0.00133585816014104
q_3	0.0843335296690778	0.0101911601102736	0.00153535704621542
q_4	0.101093867814163	0.0105581818089339	0.00159435319071038
q_5	0.13094432711773	0.0082087957396163	0.00118044224372108
q_6	0.140032027141128	0.0069429062776577	0.00081380324716971
	δ	p	c
q_0	0.394438123722037	2112.33823664052	3
q_1	0.265646758425652	1345.04651020844	3
q_2	0.542804467909787	1921.44129764702	3
q_3	0.342242723878699	1261.42225409711	3
q_4	0.590855861079745	3717.01220740379	3
q_5	0.400313343709627	1551.61409344768	3
q_6	0.397900550350454	4048.0901821955	3

Table 4: Coefficients of the initial random set `points`, generated over a "2/3-adjusted interval"

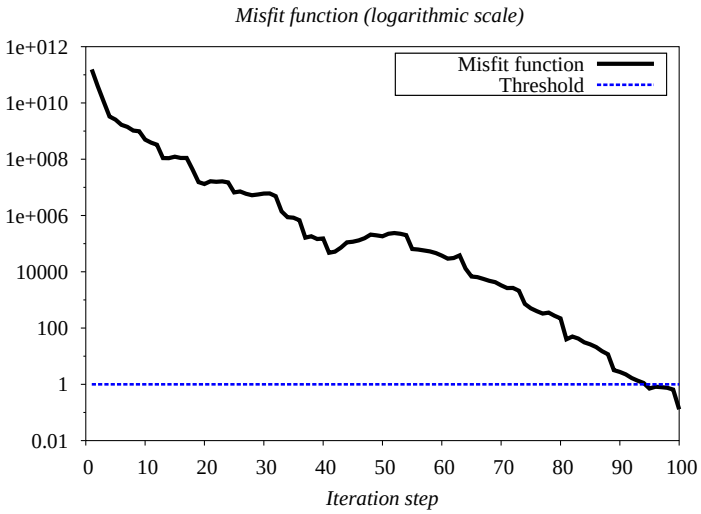
	λ		d_T		β	
Step number	Value	Relative error (%)	Value	Relative error (%)	Value	Relative error (%)
1	0.120758863	10.89	0.011395613	4.64	0.001814283	53.88
10	0.115669636	6.22	0.008978078	-17.56	0.001305623	10.74
30	0.114106539	4.78	0.008449692	-22.41	0.001185291	0.53
60	0.113997897	4.68	0.008368867	-23.15	0.001178254	-0.06
100	0.113992729	4.68	0.008364289	-23.19	0.001179676	0.06
	δ		p		c	
Step number	Value	Relative error (%)	Value	Relative error (%)	Value	Relative error (%)
1	0.42271688	15.50	1984.812433	39.09	3	0.00
10	0.379353123	3.65	1447.04152	1.40	3	0.00
30	0.368954912	0.81	1429.956621	0.21	3	0.00
60	0.36633236	0.09	1426.260772	-0.05	3	0.00
100	0.366008649	0.00	1425.38208	-0.11	3	0.00

Table 5: Coefficients of the simplex's centroid $\bar{q}$ at different steps of the *Nelder-Mead method*

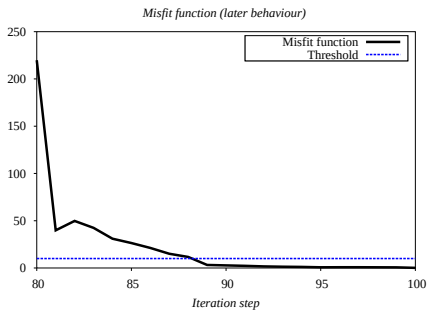
Overall, we can note a good convergence of the different coefficients towards their exact preset value, with a final relative error lower than 0.05%. The only exception seems to be d_T, whose values

become apparently further from the exact one. Besides, the final relative errors seemed to be all the more dispersed that the initial interval allowed for the random points generation is wide.

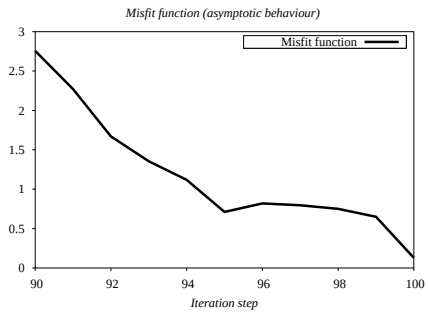
Another way to estimate the convergence of the program is to focus on the behaviour of the misfit function for each computation step. With the initial data of Table 4 (page 11), we obtain the following plots:



(k) Global behaviour (base 10 logarithm)



(l) Later behaviour



(m) Asymptotic behaviour

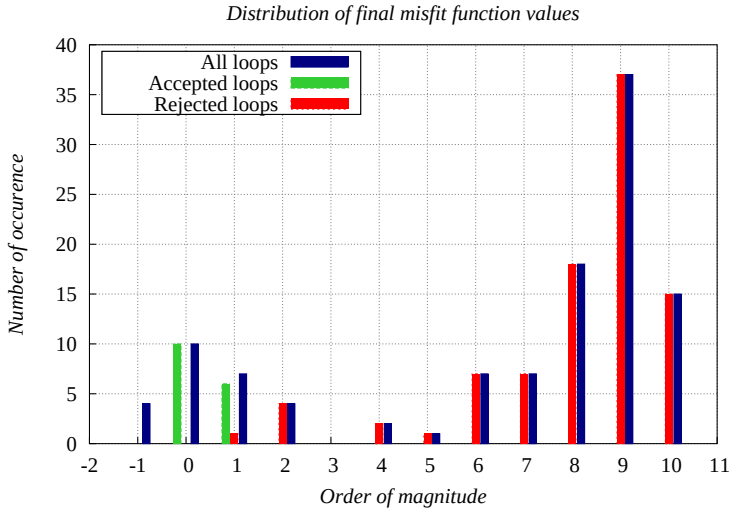
Figure 4: Values of the misfit function at each computation step

Most of the time, the simplex algorithm stopped at the 100^{th} iteration step (maximum preset number of allowed iterations), without having fallen below the tolerance `simplextol`. However, we can assume that a hundred step provides a quite satisfying accuracy insofar as the misfit function seems to reach a stationary state, with a value close to unity. Moreover, we will note in the next section that when a wrong solution is returned by the *Nelder-Mead method*, the misfit error is very likely to stay at an invariably high level, long before the end of the algorithm.

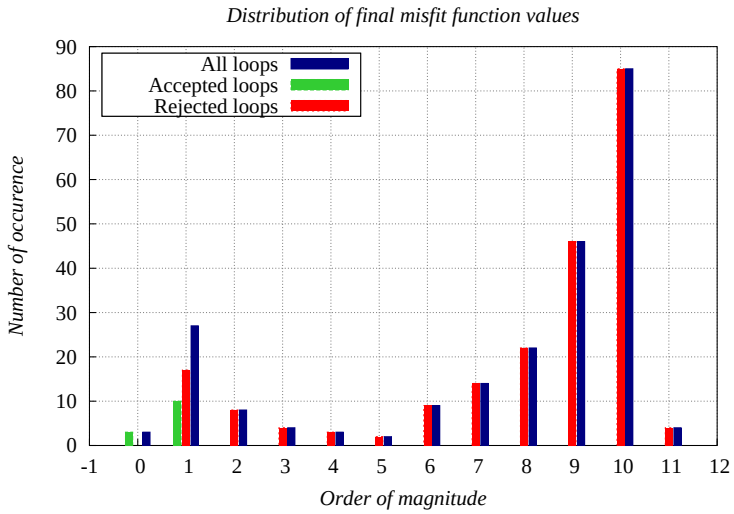
4.3 Loop performance

When the simplex is "trapped" in the area of the global minimum, a few steps lead to good results. However, when the initial set `points` is too far from the solution, the simplex may converge towards

a local minimum, which implies a fast stabilisation of the misfit function at high levels. To estimate the global performance of the program, it should be interesting to study both failures and successes of the *Nelder-Mead method*.



(a) Case of a "1/3-adjusted interval"



(b) Case of a "2/3-adjusted interval"

Figure 5: Distribution of the final values (at step 100) of the misfit function by orders of magnitude, for all *Nelder-Mead* loops (valid or non-valid).

Even if the allowed interval for the random initial points is quite narrow (1/3 of the maximal experimental bandwith for each coefficient considered), the global performance of the loop remains relatively bad: in many cases, the misfit function stays at very high levels, because the simplex has been trapped in a non-stationnary minimum. However, the valley-shape of the histogram suggests that the convergence can occur, and, when it occurs, is correctly detected. Indeed, only a few loops that could possibly return a valid minimum (i.e., with a final misfit function in the order of 10^2 or

10^1) were effectively rejected by the program. As a result, the limit of 100 steps for the *Nelder-Mead method* seems to be sufficient to discriminate with a satisfying accuracy whether the solution found may be right or wrong.

For a wider initial interval (2/3 of the maximal experimental bandwith for each coefficient considered), the same trend in the distribution of final misfit values is noted, but with a deeper valley. Moreover, the global performance of the program sinks: the program needs an increasing number of unsuccessful attempts before finding a potential minimum. The limit of 100 steps seems too become too short: we observe more rejected loops in the critical interval $[10^1, 10^2]$. Running the *Nelder-Mead method* for a longer time (e.g. 200 steps) could ensure the convergence in these cases.

Trial	Number of loops
1	4
2	2
3	4
4	7
5	4
6	2
7	2
8	5
9	17
10	1
11	1
12	12
13	1
14	3
15	6
16	4
17	1
18	7
19	29
20	1
Mean	5,65
Performance	17.7%

(a) Case of a "1/3-adjusted interval"

Trial	Number of loops
1	34
2	56
3	3
4	3
5	4
6	28
7	5
8	28
9	6
10	18
11	11
12	14
13	17
Mean	17.5
Performance	5.73%

(b) Case of a "2/3-adjusted interval"

Table 6: Number of unsuccessful loops before finding a relevant solution

5 Conclusion

The program has managed to find the coefficients of the initial differential system with a good precision. Nevertheless, its performance remains quite low, because of a total random choice of initial conditions, and a heuristic method that is also too sensible to these conditions. Finally, the trials using synthetic data (which do not include the possible errors caused by experimental measurement) can only give an overview of the operating effectiveness of the whole program.

6 Future perspective

A certain number of improvements of the program could be examined, in order to increase its performance in the search of extrema, or its ability to take into account the later fluctuations of viral loads and T-cell blood concentrations.

6.1 Model

First of all, the *SIR* simple model does not integrate the influence of latency in human body, which is yet an important long-term factor during the infection. Some studies [21] aimed to deal with this effects, by considering for instance the long-term role of peripheral virion reservoirs. As a result, the time becomes an explicit variable of the global differential system; nevertheless, the solving method does not have to change a lot, since the *Runge-Kutta* complete algorithm is precisely designed for this eventuality.

6.2 Simplex search algorithm

Furthermore, the *Nelder-Mead method* could be optimized in itself; there are indeed many variants of this algorithm, more or less complex [17]. In this regard, some methods are best suited for constrained optimization [4][19]; at each step, the points out of preset limits are reset in the good interval. This could be of great advantage, insofar as it prevents the program from looping a long time around unrealistic minima. It is also feasible to better control and optimize the adjustment of the interval where it is worth running the *Nelder-Mead method*. The classic search algorithm may thus be combined with some *Trust interval* algorithms [2][3] that seek to approximate the misfit function over a changing neighborhood for a faster convergence. Finally, the super modified simplex method [12], provides optimized transformations (reflection, expansion, contraction or reduction) by updating the coefficient (α, β or γ) with a second-order polynomial curve-fitting.

6.3 Auxiliary methods

Besides, some auxiliary loops including *Monte-Carlo method*[15] could be able to increase the probability of success during the *Nelder-Mead* algorithm, by presetting the more favorable initial conditions. Some of these techniques are based on the *Metropolis algorithm* [14][20] [11], which samples the probability distribution of the *model space* (ie, the possible solutions of inverse problem) by using of modified random walk. Combining this statistical approach with the classic simplex search could provide a more realistic model, with a better performance and a reasonable complexity (*Monte Carlo* all alone remains time-consuming)

List of Tables

List of Figures

References

[1] Ruy M. Ribeiro Alan S. Perelson. Modeling the within-host dynamics of hiv infection. *BMC Biology*, 2013.

[2] Philippe L. Toint Andrew R. Conn, Nicholas I. M. Gould. *Trust Region Methods*, volume 1. Society for Industrial and Applied Mathematics, 2000.

[3] Tadej Tuma Arpad Burmen, Iztok Fajfar. Combined simplex-trust-region optimization algorithm for automated ic design.

[4] M. J. Box. A new method of constrained optimization and a comparison with other methods. *The Computer Journal*, 8(1):42–52, 1965.

[5] Elisabetta Groppelli Pierre Pellegrino Ian Williams Persephone Borrow Benjamin M. Chain Clare Jolly Changwang Zhang, Shi Zhou. Hybrid spreading mechanisms and t cell activation shape the dynamics of hiv-1 infection. *PLOS Computational Biology*, 2015.

[6] Marc Compere. Ode solvers for octave and matlab @ONLINE, 1999. `https://sites.google.com/site/comperem/home/ode_solvers`.

[7] Mrinal Raghupathi Stephen Pankavich Eric Jones, Peter Roemer. Analysis and simulation of the three-component model of hiv dynamics. *SIAM Undergraduate Research Online*, 2013.

[8] Herbert W. Hethcote. *Applied Mathematical Ecology*, volume 18, chapter Three Basic Epidemiological Models, pages 119–144. Springer-Verlag, 1989.

[9] R. Mead J. A. Nelder. A simplex method for function minimization. *The Computer Journal*, 1965.

[10] P.J. Prince J.R. Dormand. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, pages 19–26, 1980.

[11] Albert Tarantola Klaus Mosegaard. Monte carlo sampling of solutions to inverse problems. *Journal of Geophysical Research*, 100(B7):12431—-12447, July 1995.

[12] M. B. Denton M. W. Routh, P. A. Swartz. Performance of the super modified simplex. *Analytical Chemistry*, 49(9):1422–1428, 1977.

[13] Yunzhen Cao Eric S. Daar David D. Ho Alan S. Perelson Max A. Stafford, Lawrence Corey. Modeling plasma virus concentration during primary hiv infection. *Journal of Theoretical Biology*, 2000.

[14] S. Ulam Nicholas Metropolis. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949.

[15] Gianluca Iaccarino Pietro M. Congedo, Jeroen A.S. Witteveen. Simplex-simplex approach for robust design optimization. In *EUROGEN 2011, International Conferences on Evolutionary Computing for Industrial Applications. - ECCOMAS Thematic Conference*, Capua, Italy, 2011.

[16] B.P. Sommeijer P.J. Van Der Houwen. Parallel iteration of high-order runge-kutta methods with stepsize control. *Journal of Computational and Applied Mathematics*, 29:11–127, January 1990.

[17] Catherine Porte. Méthodes directes d'optimisation, méthodes dérivées de la méthode simplex. *Techniques de l'ingénieur - qualité au laboratoire*, page 229, 2015.

[18] Peter Roemer. Stochastic modeling of the persistence of hiv: Early population dynamics. *Trident Scholar Project report*, (420), May 2013.

[19] Beata Walczak Roma Tauler, Stephen D. Brown. *Comprehensive Chemometrics: Chemical and Biochemical Data Analysis*, chapter Three Basic Epidemiological Models, pages 556–557. Elsevier Science, 2009.

[20] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*, chapter 2, pages 41–54. Society for Industrial and Applied Mathematics Philadelphia, 2005.

[21] John W. Odhiambo Waema R. Mbogo, Livingstone S. Luboobi. Stochastic model for langerhans cells and hiv dynamics in vivo. *ISRN Applied Mathematics*, January 2014.