UNIVERSITAT AUTÒNOMA DE BARCELONA

DEPARTAMENT DE MATEMÀTIQUES

TESI DOCTORAL:

TEOREMES DE NO FREE LUNCH EN EL CONTINU I EL MODEL DEL PONT BROWNIÀ EN OPTIMITZACIÓ

Autor: Ricard Josep Caballero Monteso

Programa de doctorat en Matemàtiques RD 778/98
Any de dipòsit 2015

Dirigida per:
Dr. Aureli Alabert Romero

Agraïments

Agraeixo el suport emocional que m'han donat per fer aquesta tesi als meus pares, als meus yeyos i a la Raquel. Per saber entendre en tot moment els "tinc molta feina de matemàtiques", "no em surt, haig de seguir-ho mirant",.... Moltes gràcies per fer-me costat.

Agraeixo la base de continguts que m'han donat per fer aquesta tesi: a tots els mestres i professors que he tingut (i als que tindré, per avançat), des de la meva mare amb les primeres nocions d'aritmètica, fins al director de la tesina sobre el procés Ornstein-Uhlenbeck, Frederic Utzet, passant per tots els professors i companys de la carrera de Matemàtiques de la UAB que he tingut i que tant m'han fet aprendre. I també a l'Alek Alkoum per ajudar-me a millorar l'anglès.

Agraeixo el suport tècnic que m'han donat en aquesta tesi a: Marco Ferrante i Alessandro Berti, Endre Csáki, Jim Pitman i a Mario Rodríguez Riotorto.

I sobretot agraeixo al director d'aquesta tesi, Aureli Alabert, tot el recolzament que m'ha donat, tot el que m'ha ensenyat, i tota la tranquil·litat i positivitat que m'ha transmès cada segon.

Índex

Introducció

En els camps teòrics de la computació i de l'optimització, el famós teorema del No Free Lunch ve a dir que tots els algorismes són igual de bons quan es fa la mitjana de la seva eficiència sobre totes les possibles funcions a optimitzar, és a dir no hi ha cap algorisme que sigui millor que un altre. Aquesta idea pot arribar a ser de fàcil comprensió en el cas discret ja que es veu de forma intuïtiva, fet que no succeeix quan es pensa en un cas continu. En el primer capítol, es generalitza a variable contínua aquest teorema utlilitzant de manera molt natural la teoria de processos estocàstics i arribant a la conclusió que no hi ha teoremes de No-Free-Lunch en el continu, excepte en determinats casos extrems de poca importància pràctica.

A diferència del primer capítol, el segon i tercer capítol són més pràctics i es considera el pont brownià com un model probabilístic per a problemes d'optimització tipus "caixa negra", en què no es té cap forma analítica de la funció, sinó que aquesta només pot ser avaluada en un nombre determinat de punts, i a més a més, considerant que cadascuna d'aquestes avaluacions és molt costosa de fer, i per tant només se'n faran unes quantes. El model probabilístic considera que la funció a optimitzar és una trajectòria d'un procés amb una determinada llei. Des del punt de vista de la complexitat computacional, això correspon a estudiar el "average performance" d'algorismes, en front de l'habitual "worst-case performance". Però això es fa sempre des del punt de vista asimptòtic, quan el nombre d'avaluacions tendeix a infinit, i un dels objectius d'aquests capítols se centra en la millora de l'estimació del valor òptim quan només es pot avaluar la funció en pocs punts. Per aconseguir aquest objectiu es comparen i analitzen diverses heurístiques adaptatives i no-adaptatives, i s'estudia la seva eficiència. A més, el camí d'elaboració de noves estratègies per acostar-se a la solució òptima del nostre problema donarà lloc a dos funcionals del pont brownià no explicitats anteriorment.

Finalment, afegir que tots tres capítols tenen la seva corresponent introducció i que el primer capítol està publicat a [1], el segon capítol està en procés de submissió i penjat en Arxiv [2], i el tercer capítol formarà part d'un article que s'està redactant.

Capítol 1

No-Free-Lunch Theorems in the Continuum

Abstract:

No-Free-Lunch Theorems state, roughly speaking, that the performance of all search algorithms is the same when averaged over all possible objective functions. This fact was precisely formulated for the first time in a now famous paper by Wolpert and Macready, and then subsequently refined and extended by several authors, usually in the context of a set of functions with finite domain and codomain. Recently, Auger and Teytaud have studied the situation for continuum domains. In this chapter we provide another approach, which is simpler, requires less assumptions, relates the discrete and continuum cases, and that we believe that clarifies the role of the cardinality and structure of the domain.

Keywords: No-Free-Lunch, Stochastic processes, Black-box optimisation

Mathematics Subject Classification (2010): 68Q25, 60G, 90C26

1.1 Introduction

In [28], Wolpert and Macready formulated rigorously a principle which was already intuitively known to the operations research practitioners: All search or optimization algorithms perform equally well when their performance is averaged against all possible objective functions. This principle has been known since then as the *No-Free-Lunch Theorem* (NFL for short).

The precise formulation of the Wolpert-Macready NFL Theorem will be stated in Section 1.2 (Theorem 1.2.1), but the basic assumptions are that we are dealing with the set of all functions $f\colon \mathscr{X} \to \mathscr{Y}$ between

two finite sets $\mathscr{X}$ and $\mathscr{Y}$, and that the "averaging" is uniform over all these functions. The measure of performance can be any function of the images $f(x_1),\dots,f(x_m)$ of the points $x_1,\dots,x_m$ sampled by the algorithm.

In [24], Schumacher, Vose and Whitley extended the result to some subsets of all functions (those called "closed under permutation"), whereas Igel and Toussaint [14] stated it for some non-uniform measures. The language of probability theory allows to formulate these statements in a unified and easier way and it is in our opinion very convenient to switch from the finite setting to the continuum. In [6] and [7], Auger and Teytaud considered for the first time this case, and their result is essentially negative: No NFL theorems exist in the continuum.

Our goal in this chapter is to improve and clarify the results of Auger and Teytaud, particularly Theorem 4.1, [7]. First of all, we show situations where NFL theorem do exist. This apparent paradox is resolved by noticing that the hypotheses imposed in [7] invalidate our examples. In fact, the authors seem to specifically look for conditions under which no NFL theorem can hold true. The theorem is indeed correct, although there is a gap in the proof, as explained in Section 1.3. We must also point out, however, that their paper contains much more material of interest on this and other matters.

The point of view adopted here is different: We establish a simple and natural definition of the NFL property and look for the necessary conditions implied by this definition. In this sense, our main result is Theorem 1.3.9. The conclusion we reach is that there are no No-Free-Lunch theorems for the set of functions having second-order moments, except for a few cases (illustrated in Examples 1.3.4).

The relevance of this theoretical discussion for the field of global *black-box* optimisation comes from the so-called *probabilistic models* (see, for instance [29, chap. 4]): In many practical optimisation problems there is little information about the objective function, with no access to derivatives or to any explicit formula; we are only allowed to ask the function for its value at a point of our choice and, after observing the value returned, we may decide on the next point to sample the function; and so on. Moreover, function evaluations can be expensive, and we are constrained to make only a small number of them. In these cases, it may be useful to think that the function has been drawn at random from some set of functions, according to some probability law (perhaps with some unknown parameters) that one specifies using prior information. Technically, we are then in the presence of a *stochastic process*, from which our function is a particular path. Different algorithms will choose different points for the successive evaluations, and some may perform better than others by exploiting better the model, *unless* there is a No-Free-Lunch theorem for that model. If this is the case, all algorithms perform the same in average

and, in particular, pure blind search is as good as any other proposal. In the present chapter we will see that the presence of the No-Free-Lunch property reduces to a few probabilistic models, which are not really important in practice.

A different approach to No-Free-Lunch can be found in Rowe, Vose and Wright [23], where the concept is discussed in purely set-theoretic terms as a symmetry property of a set of functions rather than in relation with any particular application. The authors conclude that a NFL property holds whenever the set of functions is closed under permutations (as in [24] for the finite case), no matter the cardinality of domain and codomain. Our approach here is totally different, to the point that the respective definitions of No-Free-Lunch are not equivalent in the non-finite context.

The chapter is organised as follows: In Section 1.2 we state the definitions and preliminaries both from algorithmics and from probability theory that are strictly needed in the rest of the chapter. In Section 1.3 we state the main results: We show that No-Free-Lunch cases do exist in the continuum; we impose then a hypothesis of measurability of the stochastic process involved, and we see that NFL can only appear if we are dealing with functions whose randomness is concentrated in a set of null Lebesgue measure (Theorem 1.3.5), or the model consists of a trivial constant process (Theorem 1.3.9). In Section 1.4 we justify the investigation of the existence (or not) of NFL properties in the continuum and propose some open questions.

1.2 Preliminaries

We follow approximately the notations of [14] and [7], with some convenient modifications.

1.2.1 Algorithmic concepts

Let $\mathscr{X}$ and $\mathscr{Y}$ be any two sets. The set of all functions $f\colon \mathscr{X} \to \mathscr{Y}$ can be identified with the Cartesian product $\mathscr{Y}^{\mathscr{X}}$. Denote

$$\mathscr{E}_0 := \{\emptyset\},\ \mathscr{E}_1 := \mathscr{X} \times \mathscr{Y},\ \ldots,\ \mathscr{E}_m = \mathscr{X}^m \times \mathscr{Y}^m$$

and $\mathscr{E} := \bigcup_{m\geq 0} \mathscr{E}_m$.

A *(random) algorithm A* is a mapping $A\colon \mathscr{Y}^{\mathscr{X}} \times \mathscr{E} \times \Theta \to \mathscr{E}$, where $(\Theta, \mathscr{G}, Q)$ is a probability space and, if $e = ((x_1,y_1),\ldots,(x_m,y_m)) \in \mathscr{E}_m$, then $A(f,e,\theta) \in \mathscr{E}_{m+1}$ and

$$A(f,e,\theta) = ((x_1,y_1),\ldots,(x_m,y_m),(x_{m+1},y_{m+1}))$$

with $y_{m+1} = f(x_{m+1})$. Therefore, we can think of

$$A(f,e) := ((X_1,Y_1),\dots,(X_m,Y_m))$$

as a random vector $\Theta \to \mathscr{E}$. This definition formalises the fact that the algorithm chooses the next point based on the previous points and an (optional) random mechanism represented by the probability space $(\Theta,\mathscr{G},Q)$. One may assume that f is never evaluated more than once at the same point.

A *measure of performance* of the algorithm is any function C of the values obtained by evaluating f during the algorithm. Formally,

$$C\colon \bigcup_{m\geq 1} \mathscr{Y}^m \to \mathbb{R}\,.$$

A typical measure of performance for optimization problems is the function $C(y_1,\dots,y_m) = \min\{y_1,\dots,y_m\}$, the best observed value after m evaluations. (Notice that the measures of performance we are talking about are not related to algorithmic complexity, e.g. to the number of evaluations needed to reach the end of a procedure.)

To state the basic Wolpert-Macready Theorem, rephrased in our probability-theoretic language, consider another probability space $(\Omega,\mathscr{F},P)$, and a random variable $f\colon \Omega \to \mathscr{Y}^{\mathscr{X}}$. Now f is random and $f(\omega)$, for each ω, is a specific function $\mathscr{X} \to \mathscr{Y}$. Denote by $A^m\colon \mathscr{Y}^{\mathscr{X}} \times \Theta \to \mathscr{E}_m$ the successive application, m times, of algorithm A to the initial empty sequence $\emptyset$, and by $A^m_Y\colon \mathscr{Y}^{\mathscr{X}} \times \Theta \to \mathscr{Y}^m$ its second component. Finally, let us abbreviate $y := (y_1,\dots,y_m) \in \mathscr{Y}^m$.

The number $Q\{A^m_Y(h,\theta) = y\}$ is the probability that the algorithm A produce the particular sequence of function values $y := (y_1,\dots,y_m)$ when applied to the function h. This probability is either 0 or 1 for deterministic algorithms.

Theorem 1.2.1. (Wolpert-Macready [28], Theorem 1).
Assume that $\mathscr{X}$ and $\mathscr{Y}$ are finite sets. Let $f\colon \Omega \to \mathscr{Y}^{\mathscr{X}}$ be a random variable that chooses functions $f(\omega) \in \mathscr{Y}^{\mathscr{X}}$ with the uniform discrete probability law. That means, for every $h \in \mathscr{Y}^{\mathscr{X}}$,

$$P\{\omega \in \Omega\colon\ f(\omega) = h\} = |\mathscr{Y}|^{-|\mathscr{X}|}\,,$$

where $|\cdot|$ denotes cardinality.

Then, the law of A^m_Y is the same for all algorithms. Precisely stated: let A and B be two algorithms; then,

for all $m \in \mathbb{N}$ and all $y \in \mathscr{Y}^m$,

$$[P \times Q]\{(\omega,\theta) :\ A_Y^m(f(\omega),\theta) = y\} = [P \times Q]\{(\omega,\theta) :\ B_Y^m(f(\omega),\theta) = y\}\ . \tag{1.1}$$

Since the law of f is uniform,

$$\begin{aligned}
&[P \times Q]\{(\omega,\theta) :\ A_Y^m(f(\omega),\theta) = y\} \\
&= [P \times Q]_{\bigcup_{h \in \mathscr{Y}^{\mathscr{X}}}} \big\{(\omega,\theta) :\ A_Y^m(h,\theta) = y \wedge f(\omega) = h\big\} \\
&= \sum_{h \in \mathscr{Y}^{\mathscr{X}}} P\{\omega : f(\omega) = h\} \cdot Q\{\theta :\ A_Y^m(h,\theta) = y\} \\
&= |\mathscr{Y}|^{-|\mathscr{X}|} \sum_{h \in \mathscr{Y}^{\mathscr{X}}} Q\big\{\theta :\ A_Y^m(h,\theta) = y\big\}\ .
\end{aligned}$$

Therefore, under the hypotheses of the theorem, (1.1) can be written

$$\sum_{h \in \mathscr{Y}^{\mathscr{X}}} Q\{\theta :\ A_Y^m(h,\theta) = y\} = \sum_{h \in \mathscr{Y}^{\mathscr{X}}} Q\big\{\theta :\ B_Y^m(h,\theta) = y\big\} \tag{1.2}$$

for any two algorithms A and B, and for all $y \in \mathscr{Y}^m$, $m \in \mathbb{N}$.

And still another, informal, way to formulate the result of Wolpert and Macready is that if an algorithm performs better than pure blind random search in a particular set of functions, then it must perform worse than random search *in the mean* on the complementary set.

For simplicity, we will assume that we deal with deterministic algorithms from now on, although everything can be easily extended to accommodate random algorithms. Since one can obviously write

$$\sum_{h \in \mathscr{Y}^{\mathscr{X}}} Q\{\theta :\ A_Y^m(h,\theta) = y\} = \sum_{h \in \mathscr{Y}^{\mathscr{X}}} \int_\Theta \mathbf{1}_{\{A_Y^m(h,\cdot) = y\}}(\theta)\, Q(d\theta)\ ,$$

for deterministic algorithms equality (1.2) can be written

$$\Big|\big\{h \in \mathscr{Y}^{\mathscr{X}} :\ A_Y^m(h) = y\big\}\Big| = \Big|\big\{h \in \mathscr{Y}^{\mathscr{X}} :\ B_Y^m(h) = y\big\}\Big|\ .$$

1.2.2 Probability concepts

For the sake of completeness and the reader's convenience, we summarise here, albeit in a very compact way, all concepts from measure and probability theory that are used in the sequel.

Let $(\Omega, \mathscr{F}, P)$ be a probability space and $(S, \mathscr{S})$ any measurable space. An S-valued random variable f is a measurable mapping $f\colon \Omega \to S$. A *random function* is simply a random variable with values in the space of all functions between two sets $\mathscr{X}$ and $\mathscr{Y}$, the second one equipped with some σ-field; that is, we take $S = \mathscr{Y}^{\mathscr{X}}$, and the natural choice for $\mathscr{S}$ is the *product* σ*-field*, i.e. the smallest σ-field that turns every projection $\mathscr{Y}^{\mathscr{X}} \to \mathscr{Y}$ into a measurable mapping. Random functions are also called *stochastic processes*, especially when $\mathscr{X}$ is an interval I of the real line and $\mathscr{Y}$ is the set of real numbers $\mathbb{R}$ endowed with the Borel σ-field. We assume in the rest of the chapter that $\mathscr{X} = [0,1]$ and $\mathscr{Y} = \mathbb{R}$, so that we are dealing with random functions $f(\omega)\colon [0,1] \to \mathbb{R}$.

The *law* of a stochastic process is the image measure of P through the mapping f. That means, it is the probability μ on $(S, \mathscr{S})$ such that

$$\mu(B) = P(f^{-1}(B)) \,, \quad \forall B \in \mathscr{S} \,.$$

Given a stochastic process f, the composition $f(t) := \delta_t \circ f$ of f with the Dirac delta at $t \in [0,1]$ is automatically a real random variable $f(t)\colon \Omega \to \mathbb{R}$, with respect to the Borel σ-field on $\mathbb{R}$. The random vectors of the form $(f(t_1), \ldots, f(t_m))$, with $t_1, \ldots, t_m \in [0,1]$, are the finite-dimensional projections of f. Their laws, the *finite-dimensional distributions*, determine the law of the whole process.

A stochastic process can be represented in different ways: As a function-valued random variable, as above, or as a family of real-valued random variables, $\{f(t),\, t \in [0,1]\}$, or as a mapping from the product $\Omega \times [0,1]$ into $\mathbb{R}$, defined in the obvious way: $(\omega, t) \mapsto f(t, \omega)$. A process is said to be *measurable* if, in the last representation, it is a measurable mapping $\Omega \times [0,1] \to \mathbb{R}$ when $\Omega \times [0,1]$ is endowed with the product σ-field $\mathscr{F} \times \mathscr{B}([0,1])$, where $\mathscr{B}([0,1])$ denotes the Borel σ-field of $[0,1]$. Stochastic processes mentioned in Auger–Teytaud [7] are always considered measurable. This is an important hypothesis in their results, and its role will be made clear in the present chapter. A process is said to be of *second-order* if all its variables are square-integrable, implying that they have finite expectation and variance.

With some abuse of notation, we use the same symbol f to denote several different related objects: f is a function $\Omega \to \mathbb{R}^{[0,1]}$, or $\Omega \times [0,1] \to \mathbb{R}$; for every $\omega \in \Omega$, $f(\omega)$ is a function $[0,1] \to \mathbb{R}$; for all $t \in [0,1]$, $f(t)$ is a random variable $\Omega \to \mathbb{R}$; and finally, for all t and ω, the value $f(t, \omega)$ is a real number.

We will also use occasionally the customary abbreviations a.s. for *almost surely* (i.e. true with probability 1), and a.e. for *almost everywhere* (i.e. true except a set of measure zero with respect to Lebesgue measure).

1.3 Main results

Recall, from the notations in Section 1.2, that $A_Y^m(f(\omega))$ is the random vector consisting of the images $(f(t_1),\dots,f(t_m))(\omega) \in \mathbb{R}^m$ produced by applying m iterations of algorithm A on the function $f(\omega) \in \mathbb{R}^{[0,1]}$.

Definition 1.3.1. *Let C be a performance measure, measurable on $\mathbb{R}^m$, for all m. We say that a stochastic process $f = \{f(t),\ t \in [0,1]\}$ satisfies the* No-Free-Lunch property with respect to C *if for any two algorithms A and B, and for all $m \in \mathbb{N}$, the random variables*

$$\omega \mapsto C\big(A_Y^m(f(\omega))\big) \quad \textit{and} \quad \omega \mapsto C\big(B_Y^m(f(\omega))\big)$$

have the same law.

Intuitively, the NFL property states that the information about C that we get after having sampled m points is the same no matter which algorithm we use. In particular, blind search performs in average as well as any other algorithm.

Definition 1.3.2. *We say that a stochastic process $f = \{f(t),\ t \in [0,1]\}$ satisfies the* No-Free-Lunch property *if it does so with respect to all possible performance measures C.*

Remark 1.3.3. It is easily seen (see e.g. Auger and Teytaud [7], Lemma 2.3), that if f satisfies the No-Free-Lunch property then the random variables

$$\omega \mapsto A_Y^m(f(\omega)) \quad \text{and} \quad \omega \mapsto B_Y^m(f(\omega))$$

have the same law. Conversely, if the above random variables have the same law, then f satisfies the No-Free-Lunch property for all performance measures C. One may say that NFL is a extremely strong form of stationarity. A *stationary process* has invariant laws under translations: The law of $(f(t_1),\dots,f(t_n))$ and $(f(t_1+h),\dots,f(t_m+h))$ are the same, for every h and any dimension m, provided all indices belong to the set where the process is defined, the interval $[0,1]$ in our case. It is clear that the NFL property is much stronger.

Examples 1.3.4. *We can readily show two examples of random functions enjoying the No-Free-Lunch property:*

1. *Consider a family of random variables $\{f(t),\ t \in [0,1]\}$, mutually independent and identically distributed, with any non-degenerate probability law (as a specific case, consider for instance*

Bernoulli or Gaussian variables). All n-dimensional joint distributions are the direct product measure of the individual laws, and are therefore the same. The NFL property is then trivially satisfied.

Observe that the domain $[0,1]$ *plays no role in the conclusion. We can replace it by any set of arbitrary cardinality and topological properties. Such a stochastic process exists by the classical Kolmogorov Extension Theorem (see e.g. [3]).*

2. *Consider any random variable X and define a constant process* $f(t) \equiv X$*, for all* $t \in [0,1]$*. The NFL property is immediate to check: The probability law of any vector* $(f(t_1), \ldots, f(t_n))$ *is concentrated in the diagonal of* R^n *and with the law of X in each component.*

3. *Consider* $f(t)$ *to be independent identically distributed random variables with* $P\{f(t) = 0\} = P\{f(t) = 1\} = 1/2$*, except at one only point* t_0*, where* $P\{f(t_0) = 0\} = 2/3$ *and* $P\{f(t_0) = 1\} = 1/3$*. The set of paths of the stochastic process* $\{f(t),\ t \in [0,1]\}$ *can be taken to be (ruling out sets of probability zero) the set of all functions* $[0,1] \to \{0,1\}$*, which is closed under permutation, and thus enjoys the NFL property in the sense of [23]. However, Definition 1.3.2 is not satisfied, because of the variable with a different law.*

Examples 1 and 2 show that there exist NFL situations also in the continuum case, and that the cardinality of the domain alone cannot be the responsible of the lack of No-Free-Lunch.

In Example 3 we illustrate the difference between having a symmetry condition in a set of functions and the NFL property when there is some probability distribution on that set.

It is certainly true that a continuous-time stochastic process with all variables mutually independent can hardly be of any interest in modelling a real phenomenon (in sharp contrast with the discrete case). Notice for example, that in the common case of Gaussian variables, almost all sample paths (i.e., all functions in the set we are considering, with probability 1) are unbounded from below and from above, which makes pointless to search for or to approximate the minimum value. As another example, if the variables have the uniform law in an interval $[\alpha, \beta]$, then almost all sample paths are bounded, with infimum equal to α and supremum equal to β, although the probability that these values are attained is zero. A very different problem is the case when α or β are unknown and we try to estimate them by sampling m points from independent variables distributed uniformly in $[\alpha, \beta]$; this is indeed a statistical problem of a real practical interest.

Such trivial NFL situations do not appear if we impose on f the condition of being a measurable process. This is the main result of Auger and Teytaud [7]. We reformulate it as the problem of finding a necessary condition for having NFL in a measurable process, and show that in this case we are dealing essentially with a constant process. This possibility does not appear in [7], because of the hypothesis of existence of a so-called "proper median", that the authors introduce in the definition of NFL, and that it looks somewhat artificial. We will not use this concept. We also point out that the argument in [7] is in our opinion not complete, since at some point in the proof of their Theorem 4.1 there is a confusion between the underlying randomness of the process and the eventual randomness of the algorithm applied.

From a purely set-theoretic point of view, the hypothesis of measurability can be judged as very restrictive (see our Example 1.3.4 (1), and consider the many particular cases that contains). However, it is very natural when dealing with the *continuum*, considered not only as a set of $\aleph_1$-cardinality, but endowed with its natural order, topological and measure-theoretic structures. This explains in part the different definitions and language that were used in [23] with respect to ours.

We start by showing that measurability and independence together collapses the process to be essentially supported on a time set of measure zero.

Theorem 1.3.5. *Let $f = \{f(t),\, t \in [0,1]\}$ be a measurable stochastic process with mutually independent random variables $f(t)$. Then, for almost all t with respect to Lebesgue measure, the random variable $f(t)$ is constant with probability 1.*

Proof. We treat first the particular case in which the process is bounded and centred. Then the result will be easily extended to the general case.

First case: Assume $\mathrm{E}[f(t)] = 0$ and that there is a constant $m \in \mathbb{N}$ such that for all t, $|f(t)| < m$.

If $f\colon \Omega \times [0,1] \to \mathbb{R}$ is a measurable mapping, then the partial mappings $f(\omega)\colon [0,1] \to \mathbb{R}$ are also measurable, for $\omega \in \Omega$ almost surely. Since, moreover, all the sample paths are bounded, it makes sense to consider their Lebesgue integrals

$$g(t) = \int_0^t f(s)\, ds\ , \quad t \in [0,1]\ .$$

From the properties of the integral, $g(t)$ is a process with continuous paths almost surely. We may leave it undefined for the exceptional set of probability zero, because this is not relevant.

The random variables $g(t)$ are also clearly bounded (e.g. by m itself), and therefore the second moment

$\mathrm{E}[g(t)^2]$ is finite. But we see that it is in fact equal to zero:

$$\mathrm{E}[g(t)^2] = \mathrm{E}\Big[\int_0^t f(s)\,ds \cdot \int_0^t f(r)\,dr\Big] = \mathrm{E}\Big[\int_0^t \int_0^t f(s)f(r)\,dsdr\Big] =$$
$$= \int_0^t \int_0^t \mathrm{E}[f(s)^2]\cdot \mathbf{1}_{\{s=r\}}\,dsdr + \int_0^t \int_0^t \mathrm{E}[f(s)]\,\mathrm{E}[f(r)]\cdot \mathbf{1}_{\{s\neq r\}}\,dsdr\,,$$

where the interchange of integral and expectation is justified by the boundedness of all functions involved, which allows to apply the Fubini theorem, and we have used the hypothesis $\mathrm{E}[f(s)\cdot f(r)] = \mathrm{E}[f(s)]\cdot\mathrm{E}[f(r)]$. Now, the first integral is equal to zero because we are integrating over the line $\{s=r\}$, which has zero Lebesgue measure, and the second one is also zero because the variables are centred.

The equality $\mathrm{E}[g(t)^2]=0$ implies that for all $\omega\in\Omega$ except maybe in a subset $N_t\subset\Omega$ of probability zero, one has $g(t,\omega)=0$. In particular, this is true for all $t\in\mathbb{Q}$ and, since $\mathbb{Q}$ is a countable set, we have that $N=\bigcup_{t\in\mathbb{Q}} N_t$ has probability zero. By the continuity of the paths, we obtain that for $\omega\in\Omega-N$, $g(t,\omega)=0$ for all $t\in[0,1]$. The integral being zero for all t, the integrand is also zero except maybe on a set of Lebesgue measure equal to zero. We conclude that a.s., and for almost all $t\in[0,1]$, $f(t,\omega)=0$.

General case:

Let $m\in\mathbb{N}$, and define:

$$\bar{f}_m(t) := f(t)\cdot \mathbf{1}_{\{|f(t)|<m\}}$$

and

$$\hat{f}_m(t) = \bar{f}_m(t) - \mathrm{E}[\bar{f}_m(t)]\,.$$

The random variables $\hat{f}(t)$ are also mutually independent. Applying the particular case above we have that for all m, the law of $\hat{f}_m(t)$ is a Dirac delta at zero, for $t\in[0,1]$ a.e., and therefore $\bar{f}_m(t)$ is constant ω-a.s, t-a.e.

Now, let $N_{t,m}\subset\Omega$ be the set where $\bar{f}_m(t)\neq f(t)$. The random variable $f(t)$ is almost surely equal to a constant $a(t,m)$ on the set $\Omega-N_{t,m}$, which tends, as $m\to\infty$, to a set $\Omega-N_\infty(t)$, whose probability is 1, given that $\lim_{m\to\infty} P\{|f(t)|\leq m\}=1$. We obtain that the constant $a(t,m)$ cannot depend on m, and conclude that for almost all $t\in[0,1]$, with respect to Lebesgue measure, the random variables $f(t)$ are degenerated. □

In other words, Theorem 1.3.5 states that, under the hypotheses of measurability and independence, randomness can only appear on a time set of zero Lebesgue measure.

For the remaining of the section, we assume that we deal with second-order processes. We will show, in Theorem 1.3.9 below, that a measurable, second-order process satisfying the NFL property is trivial: All their random variables are almost surely equal.

First, we state some preliminary results in the form of lemmas:

Lemma 1.3.6. *Let $(\Omega, \mathscr{F}, P)$ be a probability space and $f : \Omega \times [0,1] \to \mathbb{R}$ a measurable second-order stochastic process. Then the NFL property can be satisfied only if the random variables $f(t)$ are identically distributed and the covariance $\mathrm{Cov}(f(t), f(s))$ is constant for all t, $s \in [0,1]$, $t \neq s$.*

Proof. If the NFL property is satisfied, then by Remark 1.3.3 the vectors $A_Y^m(f)$ and $B_Y^m(f)$ are identically distributed for any $m \in \mathbb{N}$ and any pair of algorithms A and B.

Take $m = 1$. Given two values t and s in $[0,1]$, let A be a deterministic algorithm that chooses t as initial point and B another deterministic algorithm that chooses s as initial point. Then $f(t)$ and $f(s)$ are identically distributed.

Take now $m = 2$. Given two couples of different points (t_1, t_2) and (s_1, s_2), let A be a deterministic algorithm that choses (t_1, t_2) as the first two points and B be a deterministic algorithm that chooses (s_1, s_2). Then the random vectors $(f(t_1), f(t_2))$ and $(f(s_1), f(s_2))$ are identically distributed. This fact implies in particular that

$$\mathrm{Cov}(f(t_1), f(t_2)) = \mathrm{Cov}(f(s_1), f(s_2)) \ .$$

□

Lemma 1.3.7. *Let $X_1, \ldots, X_n$ ($n \geq 2$) be real square-integrable random variables with a common covariance $\rho = \mathrm{Cov}[X_i, X_j]$ when $i \neq j$, and define $V := \max_i \mathrm{Var}[X_i]$, the maximum of their variances. Then*

$$\rho \geq -\frac{V}{n-1}$$

Proof. We have

$$0 \leq \mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{Var}[X_i] + \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \mathrm{Cov}[X_i, X_j] \leq Vn + n(n-1)\rho$$

and the result follows at once. □

The proof of the next result is immediate:

Lemma 1.3.8. *Let $f\colon \Omega \times [0,1] \to \mathbb{R}$ be a second-order stochastic process such that the variables $f(t)$ are identically distributed, with finite common mean μ and positive common variance σ^2. Then f satisfies the NFL property if and only if the same holds for $(f-\mu)/\sigma$.*

By Lemmas 1.3.6, 1.3.7 and 1.3.8, we can restrict the search of a second-order measurable stochastic process satisfying the NFL property to the case when the variables $f(t)$ are identically distributed, with zero mean, unit variance and such that for some $\rho \geq 0$, $\mathrm{Cov}[f(t), f(s)] = \rho$ for any pair $t \neq s$.

In the next theorem we use Fourier analysis, following quite closely some arguments that can be found in Crum [12], to prove our main result:

Theorem 1.3.9. *Let $\{f(t,\omega) :\ t \in [0,1],\ \omega \in \Omega\}$ be a measurable, second-order stochastic process, with $\mathrm{E}[f(t)] = 0$ and $\mathrm{E}[f(t)^2] = 1$ for all $t \in [0,1]$, satisfying the NFL property, and defined in some probability space $(\Omega, \mathscr{F}, P)$.*

Then, the process is constant, in the sense that there exists a random variable $X\colon \Omega \to \mathbb{R}$ such that $P\{\omega \in \Omega :\ f(t,\omega) = X(\omega)\} = 1$, $\forall t \in [0,1]$.

Proof. We are going first to extend the process from $[0,1]$ to the whole real line, in order to apply Fourier transform techniques comfortably. Define:

$$f(k+t) := \begin{cases} f(t), \text{ if } t \in (0,1], k = 1,2,\ldots \\ f(t), \text{ if } t \in [0,1), k = -1,-2,\ldots \end{cases}$$

From the previous lemmas, we know that the covariance function $K(t) := \mathrm{E}[f(t+s)f(s)]$ of the extended process is equal to some ρ ($0 \leq \rho \leq 1$) for t in an interval $(-\varepsilon, \varepsilon)$, and all $s \in \mathbb{R}$, except for $t = 0$, in which is equal to 1. Our purpose is to see that in fact ρ must be equal to 1, from which the conclusion will be easily drawn.

It can be readily seen that the extended process is also measurable, using that a set $A \in \mathscr{B}(I) \otimes \mathscr{F}$, where I is any interval in $\mathbb{R}$, is also $\mathscr{B}(\mathbb{R}) \otimes \mathscr{F}$-measurable when considered as a subset of $\mathbb{R} \times \Omega$. This implies that, for almost all ω, $t \mapsto f(t,\omega)$ is a Borel measurable function, that it makes sense to consider the integrals

$$\int_{\mathbb{R}} e^{-2s^2} f(s)^2\, ds\ ,$$

and that they are measurable functions $\Omega \to \mathbb{R}$. Taking expectation and applying Fubini's theorem,

$$\mathrm{E}\Big[\int_{\mathbb{R}} e^{-2s^2} f(s)^2\, ds\Big] = \int_{\mathbb{R}} e^{-2s^2}\, \mathrm{E}[f(s)^2]\, ds = \int_{\mathbb{R}} e^{-2s^2}\, ds < \infty\,.$$

This means that, ω-a.s., $s \mapsto e^{-s^2} f(s)$ belongs to $L^2(\mathbb{R})$. It also belongs to $L^1(\mathbb{R})$, ω-a.s:

$$\mathrm{E}\Big[\int_{\mathbb{R}} e^{-s^2} |f(s)|\, ds\Big] \le \int_{\mathbb{R}} e^{-s^2}\, \mathrm{E}[f(s)^2]^{1/2}\, ds = \int_{\mathbb{R}} e^{-s^2}\, ds < \infty\,.$$

For such ω, consider the function

$$g(s) := e^{-s^2} f(s) - e^{-(s+t)^2} f(s+t)\,.$$

Since $g \in L^1(\mathbb{R})$, we may take its Fourier transform

$$\hat{g}(\xi) = \int_{\mathbb{R}} g(s) \cdot e^{-2\pi i s\xi}\, ds$$

which can be written as

$$\int_{\mathbb{R}} e^{-s^2} f(s) \cdot e^{-2\pi i s\xi}\, ds - \int_{\mathbb{R}} e^{-s^2} f(s) \cdot e^{-2\pi i (s-t)\xi}\, ds\,,$$

or

$$(1 - e^{-2\pi i t\xi}) \cdot \hat{F}(\xi)\,,$$

where $\hat{F}$ is the Fourier transform of $s \mapsto e^{-s^2} f(s)$.

Since $g \in L^2(\mathbb{R})$ also, by Plancherel's Theorem,

$$G(t,\omega) := \int_{\mathbb{R}} g(s)^2\, ds = \int_{\mathbb{R}} |\hat{g}(\xi)|^2\, d\xi = \int_{\mathbb{R}} |1 - e^{-2\pi i t\xi}|^2 \cdot |\hat{F}(\xi)|^2\, d\xi\,.$$

The integrand tends to zero as $t \to 0$, and it is dominated by $4 \cdot |\hat{F}(\xi)|^2 \in L^1(\mathbb{R})$. Therefore, $\lim_{t\to 0} G(t,\omega) = 0$, a.s.

Moreover,

$$\begin{aligned} G(t,\omega) &= \int_{\mathbb{R}} \left| e^{-s^2} f(s) - e^{-(s+t)^2} f(s+t) \right|^2 ds \\ &\le 2\Big[\int_{\mathbb{R}} e^{-2s^2} f(s)^2\, ds + \int_{\mathbb{R}} e^{-2(s+t)^2} f(s+t)^2\, ds\Big] = 4\int_{\mathbb{R}} e^{-2s^2} f(s)^2\, ds\,, \end{aligned}$$

that belongs to $L^1(\Omega)$, as we have seen before. By the Dominated Convergence Theorem again,

$$\lim_{t\to 0} \mathrm{E}[G(t,\omega)] = 0 \,.$$

On the other hand,

$$\begin{aligned} G(t,\omega) &= \int_{\mathbb{R}} e^{-2s^2} f(s)^2\, ds + \int_{\mathbb{R}} e^{-2(s+t)^2} f(s+t)^2\, ds \\ &\quad - 2\int_{\mathbb{R}} e^{-s^2-(s+t)^2} f(s) f(s+t)\, ds \\ &= 2\int_{\mathbb{R}} e^{-2s^2} f(s)^2\, ds - 2\int_{\mathbb{R}} e^{-s^2-(s+t)^2} f(s) f(s+t)\, ds \,. \end{aligned}$$

The expectation of the first term is equal to

$$2\int_{\mathbb{R}} e^{-2s^2}\, ds = \sqrt{2\pi} \,.$$

For the second, it yields

$$2\int_{\mathbb{R}} e^{-s^2-(s+t)^2} K(t)\, ds = \sqrt{2\pi} e^{-t^2/2} K(t)$$

(the interchange of integral and expectation is justified here by checking first the integrability).

We get

$$\mathrm{E}[G(t,\omega)] = \sqrt{2\pi}\big(1 - e^{-t^2/2} K(t)\big) \,.$$

Hence

$$0 = \lim_{t\to 0} \mathrm{E}[G(t,\omega)] = \lim_{t\to 0} \sqrt{2\pi}\big(1 - e^{-t^2/2} K(t)\big) \,,$$

which implies that $\lim_{t\to 0} K(t) = 1$, and we conclude that $\rho = 1$, as we wanted to see.

Finally, since $\mathrm{Cov}[f(t), f(s)] = 1$, we have $f(t) = \alpha f(s) + \beta$, for some α, β. But the variables are centred, and this implies $\beta = 0$, whereas the unit variances yield $|\alpha| = 1$. The negative value of α is impossible because the covariance is nonnegative. Hence $f(t) = f(s)$, almost surely, for all t and s, and the proof is complete. □

Corollary 1.3.1. *In view of Lemma 1.3.8, the conclusion of Theorem 1.3.9 is true without the hypotheses of null expectation and unit variance.*

Notice that the conclusion of the previous theorem and corollary does not mean that almost all sample paths are constant, because the null set N_t where the equality $f(t) = X$ fails depends on t. However,

in an optimisation setting it is natural to specify a regularity assumption on the functions, besides the probabilistic model. For example, the continuity of the paths (or simply the right or left continuity) automatically yields that the union $\cup_{t\in[0,1]}N_t$ has probability zero, and in that case one may say that the process is constant in the sense that, except on a set $N \subset \Omega$ of probability zero, all paths $f(t)$ are constant.

Summarizing the present section:

- Without measurability assumptions, we showed two examples of NFL property in the continuum: The case in which all variables are independent, identically distributed, and the case where the process is constant: $f(t) = X$ a.s, $\forall t$, for some random variable X. Both are unimportant from the optimisation practitioner's point of view, and both are ruled out in the mentioned paper by Auger and Teytaud, by imposing the measurability and the "proper median" hypotheses, respectively.

- We have shown that measurability and independence together lead to a "zero-measure time" process, and that measurability and NFL (for second-order processes) imply that the process is constant. With the three conditions together, or simply measurability, independence and stationarity, one gets easily that each variable of the process must be almost surely equal to some constant k, the same for all of them: $P\{f(t) = k\} = 1, \forall t$.

1.4 Conclusion and open questions

It is frequently argued that in realistic scenarios the hypotheses of the NFL theorems are always violated, already in the finite cases. We have shown that also in the continuum the necessary conditions for NFL are too restrictive to be found in practice.

This means that in every practical situation there must be some information on the objective function that permits, in principle, to choose algorithms that perform better than pure blind search. We believe that the usefulness of the (no)-NFL statements is precisely on the theoretical side, to highlight that any proposal of a search algorithm, supported by a benchmark of functions in which it behaves well, should be accompanied by a study of the benchmark common features that help that algorithm beat the others. In other words, as has been emphatically pointed out in a recent expository article [25]: «It is clear now that for the practitioner the correct question is not *which algorithm I have to use* but first of all *what is the geometry of the objective function*».

Before papers [6], [7] and, to some extent, [23], there was, to our knowledge, no special interest in investigating the existence of NFL theorems in the continuum. The typical argument was that only the finite case is important in practice, since the computations are always made in finite precision. But nowadays it is possible to work in arbitrary precision, so that potentially one can decide to evaluate a function at any real number having a finite quantity of non-zero bits. In this sense, from the theoretical foundations viewpoint, it is interesting to use the continuum as a mathematical model of the domain for continuous optimization problems.

There are still two questions that deserve further study concerning NFL theorems, both in the discrete and in the continuum settings:

- The first one is the consideration of noisy functions, that means, black-box functions that may answer differently when asked twice for the value at the same point t. This is not uncommon in practice, since the computation of the objective value at a feasible point may involve itself some randomness or the heuristic solution of another optimisation problem. In that case, the algorithms to consider should be allowed to sample more than once the same point.

- More importantly, the second question refers to the concept of No-Free-Lunch itself. As we have seen, the NFL property is so strong that constant processes are the only measurable processes that qualify. But if we are just concerned with minimizing a function, the relevant performance measure is $C(y_1,\dots,y_n) = \min\{y_1,\dots,y_n\}$, or perhaps some related function. Recalling Definition 1.3.1 applied to this measure, it is easy to see that NFL with respect to C is not sufficient to conclude that the laws of $\omega \mapsto A_Y^m(f(\omega))$ and $\omega \mapsto B_Y^m(f(\omega))$ have to be the same, and then our Theorem 1.3.9 need not be true. We believe that this point deserves further investigation.

1.5 Acknowledgements

We thank the reviewers of TCS for their useful comments, and especially for pointing us to the paper [23], of which the authors were completely unaware, and that provides an interesting and different point of view on the No-Free-Lunch theme.

This work has been supported by grants numbers MTM2011-29064-C03-01 from the Ministry of Economy and Competitiveness of Spain; UNAB10-4E-378, co-funded by the European Regional Development Fund (ERDF); and 60A01-8451 from the University of Padova.

Capítol 2

Minimum of conditioned Brownian bridge

Abstract:

We study the law of the minimum of a Brownian bridge, conditioned to take specific values at specific points, and the law of the location of the minimum. They are used to compare some non-adaptive optimisation algorithms for black-box functions for which the Brownian bridge is an appropriate probabilistic model and only a few points can be sampled.

Keywords: Black-box optimisation, Brownian bridge, simulation.

Mathematics Subject Classification (2010): 90C26, 60J65, 65C05

2.1 Introduction

We study the law of the minimum of a Brownian bridge conditioned to pass through given points in the interval $[0, 1]$, and the location of this minimum. Our motivation is the investigation of the performance of algorithms based on probabilistic models in expensive black-box optimisation.

The probabilistic model point of view assumes the existence of a probability space from where the function at hand has been drawn. The choice of points to sample is guided by the probabilistic properties of this random function. Eventually, the values of the function at the points already sampled can be used to decide the next sampling point (*adaptive algorithms*) or neglected (*non-adaptive* or *passive algorithms*).

We assume here that the probabilistic model is completely specified, and given by the standard Brownian bridge on the interval $[0, 1]$; that means, the function to be optimised is a path of a standard Brownian

motion process, conditioned to take certain values x_0 at $t = 0$ and x_1 at $t = 1$. More generally, one could set up a *statistical model* (a family of probabilistic models depending on some parameters) and improve sequentially the knowledge of the parameters using the values observed while sampling.

Probabilistic models try to account for heavy multimodality in the objective function. The irregularity and the independence of values over disjoint intervals of the Brownian bridge and other Markovian stochastic processes represent well this multimodality, although at a very local scale the functions found in practice are usually smooth.

Our main interest is in expensive black-box functions from which only a few points can be sampled, where it is more important to have an estimation of the absolute error committed in approximating the true minimum than the convergence, the speed of convergence or the complexity properties of the algorithm.

In this chapter we establish some facts about the law of the minimum of a Brownian bridge on the interval $[0,1]$, conditioned to hit some points in the interior. The density function of the law can be computed exactly, but we argue that it is better to use simulation to obtain its features. We then use these simulations to evaluate empirically the performance of three simple non-adaptive algorithms when only small samples are allowed. New adaptive algorithms in the same setting will be presented and compared elsewhere.

The Brownian bridge model in optimisation has been studied by several authors, from the point of view of the asymptotic properties of the algorithms (see, e.g. Locatelli [19], Ritter [22], Calvin [9, 10, 11]). We mention here just two facts:

1. *Long-run performance:* Sampling at n equidistant points and taking the value of the best sampled point as the approximation of the true minimum has an absolute error whose expectation is $O(1/\sqrt{n})$. The best adaptive algorithm is better than the best non-adaptive algorithm concerning improvement rates, but asymptotically both are $O(1/\sqrt{n})$. Thus, sampling at equidistant points is optimal in the long run.

2. *Complexity:* For algorithms using n function evaluations, the convergence to zero of the mean error cannot be $O(e^{-cn})$ for any constant c. (This order is indeed attained in unimodal functions, for example by Fibonacci search.)

We establish some notations and preliminaries in Section 2.2. Section 2.3 is devoted to computing the probability that the minimum lies in a given interval determined by two of the conditioning points. In

Section 2.4 we show how to simulate the law of global minimum of the process. In Section 2.5 we test and compare three non-adaptive algorithms from the point of view of the expected difference between the best sampled point and the true minimum of the path, when the evaluation points are few. Finally, in Section 2.6, we compute the conditional distribution of the location of the minimum of a single Brownian bridge given the value of this minimum, and we show how to use it to simulate the location of the minimum of the whole process.

2.2 Preliminaries

In the sequel, for a given stochastic process $Z := \{Z_t,\ t \in I\}$, defined on a closed interval $I \subset \mathbb{R}$, we denote by

$$m(Z) := \min_{t\in I} Z_t \qquad \text{and} \qquad \theta(Z) := \arg\min_{t\in I} Z_t$$

the random variables giving the minimum value of Z and its location, respectively. In the cases we will treat here, the minimum exists and is unique with probability 1 but, to avoid any ambiguity, one can assume that $\theta(Z)$ is the first point where the minimum is achieved.

A standard Brownian motion W on the interval $[t_0,t_1]$, starting at (t_0,a), $a \in \mathbb{R}$, is a Markov stochastic process with continuous paths, defined by the transition probability

$$p_{s,r}(x,y) = \frac{1}{\sqrt{2\pi(r-s)}} \exp\left\{\frac{-(y-x)^2}{2(r-s)}\right\}, \quad t_0 \le s < r \le t_1\ ,$$

and such that $W_{t_0} = a$ with probability 1.

A Brownian bridge B starting at (t_0,a) and ending at (t_1,b) has the law of a Brownian motion defined on the time interval $[t_0,t_1]$ starting at (t_0,a) and conditioned to take the value b at t_1. The random variable B_t, $t_0 < t < t_1$, is Gaussian with mean $a + \frac{t-t_0}{t_1-t_0}(b-a)$ and variance $\frac{(t-t_0)(t_1-t)}{t_1-t_0}$.

The following results are known or easily deduced (see e.g. Karatzas and Shreve [16, Sec. 2.8]):

Proposition 2.2.1. *Let W be a Brownian motion starting at (t_0,a), defined on the interval $[t_0,t_1]$. The density function of its minimum $m(W)$ is given by*

$$f_{m(W)}(y) = \sqrt{\frac{2}{\pi}}(t_1-t_0)\exp\left\{\frac{-(a-y)^2}{2(t_1-t_0)}\right\}\mathbf{1}_{\{y<a\}}\ . \tag{2.1}$$

Let B be a Brownian bridge from (t_0,a) to (t_1,b). The density function ot its minimum $m(B)$ is given by

$$f_{m(B)}(y) = \frac{2}{t_1-t_0}(a+b-2y)\exp\Big\{\frac{-2(a-y)(b-y)}{t_1-t_0}\Big\}\mathbf{1}_{\{y<a,y<b\}}\ . \tag{2.2}$$

□

Given $0=t_0<t_1<\cdots<t_n<t_{n+1}=1$, and real values $x_0,\ldots,x_{n+1}$, we are interested in a stochastic process $X := \{X_t,\ \in [0,1]\}$ whose law is that of a Brownian bridge starting at (t_0,x_0), ending at (t_{n+1},x_{n+1}), and conditioned to pass through all the intermediate points (t_i,x_i), $i=1,\ldots,n$.

This process can be thought as the concatenation of $n+1$ independent Brownian bridges $B^i := \{B^i_t,\ t\in [t_i,t_{i+1}]\}$, with end values x_i and x_{i+1}. In the optimisation application that we have in mind, the interior points $t_1,\ldots,t_n$ are the points sampled by the algorithm, and $x_1,\ldots,x_n$ are the observed values at those points.

The law of the minimum of the process X can be expressed in terms of the law of the minimum of its pieces, in the usual way. Despite the mutual independence of the Brownian bridges, this cannot be simplified further:

Proposition 2.2.2. *Let X be the conditioned Brownian bridge defined above, and $m(X)$ its minimum. Then, for all $y\in\mathbb{R}$,*

$$P\big\{m(X)>y\big\} = \prod_{i=0}^{n}\Big(1-\exp\Big\{\frac{-2(x_{i+1}-y)(x_i-y)}{t_{i+1}-t_i}\Big\}\Big)\mathbf{1}_{\{y<\min(x_0,\ldots,x_{n+1})\}} \tag{2.3}$$

Proof. The formula comes from the standard computation of the law of the minimum of several independent random variables:

$$F_{m(X)}(y) = 1-\prod_{i=0}^{n}(1-F_{m(B^i)}(y))\ ,$$

where $F_{m(X)}$ is the distribution function of $m(X)$, and $F_{m(B^i)}$ is the distribution function of the minimum of the Brownian bridge B^i, whose density is given by (2.2), adjusting the appropriate constants. □

Note that in the case when we do not condition to the end point (t_{n+1},x_{n+1}), we obtain a similar expression where, according to (2.1), the last factor in (2.3) is replaced by

$$1-\int_{-\infty}^{y}\sqrt{\frac{2}{\pi}}(1-t_n)\exp\Big\{\frac{-(x_n-z)^2}{2(1-t_n)}\Big\}\,dz\ .$$

It would not be difficult to deal with this situation separately (a conditioned Brownian motion), but we will keep our assumptions for simplicity. Moreover, sampling at $t = 1$ reverts to our case.

It is natural to try to compute explicitly the density $f_{m(X)}$ of the minimum of X by conditioning to each of the intervals $[t_i, t_{i+1}]$:

$$f_{m(X)}(y) = \sum_{i=0}^{n} P\{\theta(X) \in [t_i, t_{i+1}]\} \cdot f_{m(X)_{|\theta(X)\in[t_i,t_{i+1}]}}(y) \ .$$

Even though, as we will see, the probability of $\theta(X)$ lying in a given interval can be, in principle, computed exactly, the conditional densities in the second factors still depend on the rest of the process, and thus they are not simply densities of the minimum of a single Brownian bridge.

2.3 Probability that $\theta(X)$ belong to $[t_i, t_{i+1}]$

2.3.1 Analytical formulae

The probability that the minimum of X is achieved in one of the intervals $[t_i, t_{i+1}]$ can be computed exactly:

Proposition 2.3.1. *The probability that the minimum of the process X is located in the interval* $[t_i, t_{i+1}]$ *is given by:*

$$\begin{aligned} P_{\theta(X)}([t_i, t_{i+1}]) = \int_{-\infty}^{\min(x_0,\dots,x_{n+1})} & \frac{2}{t_{i+1}-t_i}(x_i + x_{i+1} - 2y) \exp\Big\{\frac{-2(x_i-y)(x_{i+1}-y)}{t_{i+1}-t_i}\Big\} \times \\ & \prod_{j\neq i}\Big(1 - \exp\Big\{\frac{-2(x_j-y)(x_{j+1}-y)}{t_{j+1}-t_j}\Big\}\Big)\, dy \end{aligned} \tag{2.4}$$

Proof. The random variables $m(B^0), \dots, m(B^n)$ are independent, because of the Markov property of Brownian motion. Therefore, their joint density is given by the product $\prod_{i=0}^{n} f_{m(B^i)}(y_i)$, where $f_{m(B^i)}$ is the density of the minimum of the i-th bridge. Denoting, for simplicity, $f_i := f_{m(B^i)}$ and F_i the corresponding distribution function,

$$\begin{aligned} P\big\{\theta(X) \in [t_i, t_{i+1}]\big\} &= \int_{\prod_{\substack{j=0\\ j\neq i}}^{n} \{y_i < y_j\}} f_i(y_i) \times \prod_{\substack{j=0\\ j\neq i}}^{n} f_j(y_j)\, dy_0 \cdots dy_n \\ &= \int_{-\infty}^{\infty} f_i(y_i) \times \Big(\prod_{\substack{j=0\\ j\neq i}}^{n} \int_{y_i}^{\infty} f_j(y_j)\, dy_j\Big)\, dy_i = \int_{-\infty}^{\infty} f_i(y) \times \prod_{\substack{j=0\\ j\neq i}}^{n} \big(1 - F_j(y)\big)\, dy \ , \end{aligned}$$

Now, the result is obtained using (2.2) and the corresponding distribution functions. □

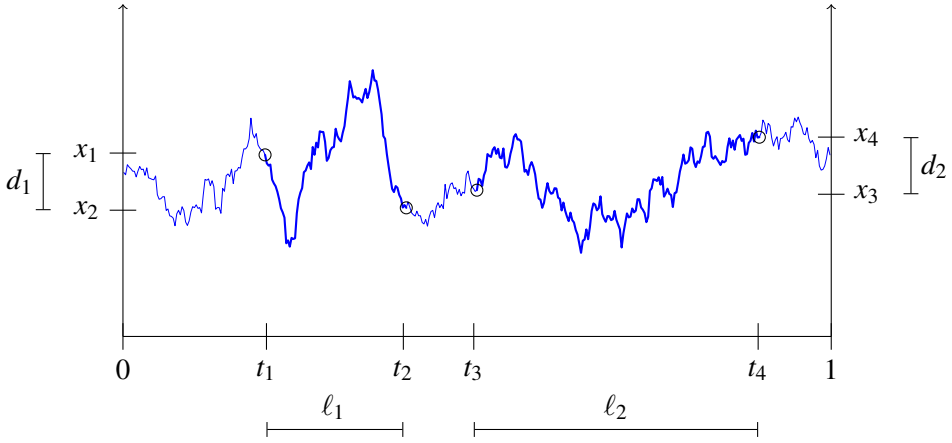

Figure 2.1: A path of Brownian motion conditioned to the circled points

The integral in (2.4) can be obtained analytically using a computer algebra system. It is a long expression that we will not copy here. Let us compare, instead, the probability of two different intervals:

Let $t_1 < t_2 < t_3 < t_4$ and consider the Brownian bridge B_1 from (t_1,x_1) to (t_2,x_2) and the Brownian bridge B_2 from (t_3,x_3) to (t_4,x_4). Denote $\ell_1 := t_2 - t_1$, $d_1 := |x_2 - x_1|$, $\ell_2 := t_4 - t_3$, $d_2 := |x_4 - x_3|$, and $\xi := x_3 \wedge x_4 - x_1 \wedge x_2$. See Figure 2.1.

We ask ourselves which of the two intervals $[t_1,t_2]$ and $[t_3,t_4]$ is more likely to contain the minimum of the process. We have

$$\begin{aligned} P\{m(B_1) < m(B_2)\} &= \int_{\{y<\bar{y}\}} f_{m(B_1)}(y) f_{m(B_2)}(\bar{y})\, dy\, d\bar{y} \\ &= \int_{-\infty}^{\infty} f_{m(B_1)}(y) \Big(\int_{y}^{\infty} f_{m(B_2)}(\bar{y})\, d\bar{y} \Big)\, dy\,. \end{aligned}$$

Taking as new variables $y - x_0 \wedge x_1$ instead of y, and $\bar{y} - x_2 \wedge x_3$ instead of $\bar{y}$, we get

$$\int_{-\infty}^{\xi\wedge 0} \frac{2}{\ell_1}(d_1 - 2y) \exp\Big\{ \frac{2y(d_1-y)}{\ell_1} \Big\} \Big(\int_{y-\xi}^{0} \frac{2}{\ell_2}(d_2 - 2\bar{y}) \exp\Big\{ \frac{2\bar{y}(d_2-\bar{y})}{\ell_2} \Big\}\, d\bar{y} \Big)\, dy\,,$$

which can be written

$$\int_{-\infty}^{\xi\wedge 0} \frac{2}{\ell_1}(d_1 - 2y) \exp\Big\{ \frac{2y(d_1-y)}{\ell_1} \Big\} \Big(1 - \exp\Big\{ \frac{2(y-\xi)(d_2-(y-\xi))}{\ell_2} \Big\} \Big)\, dy\,. \tag{2.5}$$

This integral is also computable analytically. Its value depends on five parameters $(\ell_1, d_1, \ell_2, d_2, \xi)$, which are independent from each other in a general setting. Therefore, there is no easy way to tell if it is more likely to find the minimum in one interval or the other. One observes, as the intuition suggests, that the above probability is an increasing function of ℓ_1, d_2 and ξ, and that is decreasing in ℓ_2 and d_1,

when all the other parameters are fixed.

In the case when the intervals are $[0,t_1]$ and $[t_1,1]$, then $\ell_2 = 1-\ell_1$, and ξ can be expressed in terms of d_1 and d_2, in different ways according to the relative positions $x_0 < x_1 < x_2$, $x_0 < x_2 < x_1$, or $x_1 < x_0 \wedge x_2$, so that the number of parameters reduces to three.

Example 2.3.1. *Let B_1 be the bridge from $(0,0)$ to $(0.5,0)$, and B_2 the bridge from $(0.5,0)$ to $(1,d_2)$, for $d_2 \geq 0$, and set $p := P\{m(B_1) < m(B_2)\}$. The following table illustrates how p and d_2 are related.*

p	d_2
0.5	0.0000
0.6	0.1837
0.7	0.4386
0.8	0.8384
0.9	1.6620
0.95	2.7302
0.99	6.8638

In fact, the explicit functional relationship is given by $p = \frac{1}{2} + \sqrt{\pi/8}\, d_2 \exp\{d_2^2/2\}\big(1 - \mathrm{erf}\{d_2/\sqrt{2}\}\big)$, where erf() *is the standard error function. If we keep the same first bridge, and make the second shorter and ending at zero, say from $(1-\ell_2,0)$ to $(1,0)$, the dependence between p and the length ℓ_2 is even easier: $p = 1/(2\ell_2+1)$. Both are straightforward computations from expression (2.5).*

By equating both expressions one obtains the variations in d_2 and ℓ_2 that give an equivalent raise of the probability that the first interval contain the minimum of the path.

2.3.2 Approximate computation

Despite the fact that the integrals (2.4) can be computed analytically, the time needed to solve them grows exponentially in the number n of intervals. Indeed, the exact computation involves decomposing the integrand in the sum of $O(2^n)$ terms. Each term has an elementary primitive, but in an optimisation procedure in which more and more points are sampled, and consequently the Brownian bridge is conditioned to one more point each time, the computation becomes cumbersome very quickly. For example, with just 8 intervals, the computer algebra system `maxima` takes more than three hours to obtain the result, in an Intel i7 CPU with plenty of memory at its disposal (although `maxima` only uses one of its cores). It is therefore justified to resort to an approximate method.

We remark that adding one more point to the set of conditioning points (that means, splitting one of the intervals in two), forces to recompute from scratch the probabilities of all intervals. There seems to be no way to reuse previous computations.

As we have seen, the probabilities $P\{m(B^i) < m(B^j)\}$, for each pair of indices i, j, can be computed exactly and more easily than (2.4); nevertheless, they are not useful even to find the interval with the maximal probability. An interval $[t_i, t_{i+1}]$ may satisfy $P\{m(B^i) < m(B^j)\} > 1/2$, $\forall j \neq i$, and still not be the interval with the largest probability of containing $m(X)$. For instance, if we condition the Brownian motion to pass through the points

$$(0,0),\ (0.144, 0.225),\ (0.610, 0.344),\ (1, 0.145)\ ,$$

we find that $P\{m(B^1) > m(B^2)\} = 0.5436$ and $P\{m(B^1) > m(B^3)\} = 0.5198$. However, the first interval is the least probable one to contain the minimum:

$$P_{\theta(X)}([t_0,t_1]) = 0.3124\ , \quad P_{\theta(X)}([t_1,t_2]) = 0.3374\ , \quad P_{\theta(X)}([t_2,t_3]) = 0.3502\ .$$

Even more, such an interval may not exist. For instance, conditioning to

$$(0,0),\ (0.392, 0.031),\ (0.594, -0.157),\ (1, 0.435)\ ,$$

one gets the circular relation $P\{m(B^1) < m(B^2)\} = .5018$, $P\{m(B^2) < m(B^3)\} = .5032$, $P\{m(B^3) < m(B^1)\} = .5013$.

All these arguments support the need to compute (2.5) numerically. It is easy to do it with a rigorous error bound: For some $\hat{x} < \min(x_0, \ldots, x_{n+1})$, split the integral into the two intervals $(-\infty, \hat{x}]$ and $[\hat{x}, \min\{x_0, \ldots, x_{n+1}\}]$. On the first one, the integral is bounded by

$$\int_{-\infty}^{\hat{x}} \frac{2}{t_{i+1} - t_i} (x_i + x_{i+1} - 2y) \exp\left\{ \frac{-2(x_i - y)(x_{i+1} - y)}{t_{i+1} - t_i} \right\} dy =$$
$$\exp\left\{ \frac{-2(x_i - \hat{x})(x_{i+1} - \hat{x})}{t_{i+1} - t_i} \right\} \leq \exp\left\{ \frac{-2(x_i \wedge x_{i+1} - \hat{x})^2)}{t_{i+1} - t_i} \right\}\ .$$

To make this quantity less than a fixed small ε, we can take $\hat{x} < x_i \wedge x_{i+1} - \left(\frac{t_{i+1} - t_i}{2} \log \frac{1}{\varepsilon}\right)^{1/2}$.

For the second interval, denoting the integrand by f and using for instance the standard rectangle rule with step size h, the error is bounded by $\frac{1}{2} \|f'\|_\infty \cdot h \cdot L$, where $L := \min(x_0, \ldots, x_{n+1}) - \hat{x}$.

Differentiating f and taking into account that all the exponentials take values less than 1, one obtains

$\|f'\|_\infty \le C$ with

$$C := \frac{4}{t_{i+1}-t_i}\left[1+(x_i+x_{i+1}-2\hat{x})\sum_{j=0}^{n}\frac{1}{t_{j+1}-t_j}(x_j+x_{j+1}-2\hat{x})\right],$$

and the integration step size to ensure an error less than ε must be

$$h \le \frac{2\varepsilon}{C\cdot L}.$$

A much more efficient method but with a not completely rigorous error bound is given by the `quadpack` functions present in the C Gnu Scientific Library and the Fortran SLATEC Library, which apply a Gauss-Kronrod rule [20]. With $n = 50$, the computation is completed in less than one-tenth of second, in an Intel i7 CPU at 2.40GHz with 20GB RAM, using the `quadpack` routines implemented in the computer algebra system `maxima`, with an estimated absolute error rarely bigger than 10^{-9}.

The integral of (2.4) can also be transformed into an integral on $[0,1]$ setting $y = \min_i x_i - (1-x)/x$ (this is what `quadpack` does), and the new integrand presents no singularities.

Example 2.3.2. *In Table 2.1, we show the effective computation of the probability that the minimum fall in the first interval, in several situations and with different methods. Sets 1 and 2 comprise four intervals, with end-points at* $t = (0,.1,.2,.5,1)$*, and values* $x = (0,0,0,0,0)$ *and* $x = (0,.1,.2,.3,.4)$ *respectively. Sets 3 and 4 comprise sixteen intervals, with end-points*

$$t = (0,.025,.050,.075,.100,.125,.150,.175,.200,.275,.350,.425,.500,.625,.750,.875,1),$$

and all images set to zero in set 3 and to $x = i/40,\ i = 0,\ldots,16$*, in set 4.*

The methods are: 1) the analytical computation of the integral (2.4), only in the case of fewest intervals ("exact"); 2) the `quadpack` *functions through* `maxima`*; 3) the* `romberg` *routine built-in in* `maxima`*; 4) the Riemann approximations with 10 000 subintervals, taking always their left points; 5) the Riemann approximations with the same number of subintervals, taking a random point in each one; and 6) the simulation method explained in the next section. In 3),4),5), the computations are also made after the mentioned explicit transformation to the interval [0,1]. In 6) a sample of size 10 000 is taken. For the methods including randomness, 5) and 6), we show the highest error observed after 20 realizations.*

All computations were programmed in `maxima`*. Time and memory are relative to the fastest and the more economic method in each case; we used the figures reported by* `maxima` *itself in a single run.*

	Set 1. Result: 0.05722062072176488			Set 2. Result: 0.3539550244743264		
	error	time	memory	error	time	memory
1) exact		1	1.26		57.2	199
2) quadpack	$< 10^{-16}$	1.07	1	$< 10^{-13}$	1	1
3) Romberg	$< 10^{-11}$	1.07	1.54	$< 10^{-11}$	1.07	1.63
4) Riemann left	$< 10^{-16}$	122	141	4.146×10^{-6}	121	137
5) Riemann random	1.008×10^{-6}	103	81.8	4.283×10^{-6}	100	79.0
6) simulation	4.179×10^{-3}	249	191	6.045×10^{-3}	252	188
	Set 3. Result: 0.003053658531871728			Set 4. Result: 0.3498434691309963		
	error	time	memory	error	time	memory
2) quadpack		1	1		1	1
3) Romberg	$< 10^{-12}$	3.41	2.89	$< 10^{-11}$	3.52	2.91
4) Riemann left	$< 10^{-18}$	197	294	$< 10^{-7}$	199	158
5) Riemann random	1.366×10^{-7}	143	157	8.140×10^{-6}	144	84.3
6) simulation	1.446×10^{-3}	459	462	1.06×10^{-2}	456	254

Table 2.1: See Example 2.3.2

They give therefore just a rough idea of the computational cost. In the case of 16 intervals, the "exact" computation is infeasible and we have taken the result of `quadpack` *as the base for the figures of the other methods.*

2.4 Simulating the law of the minimum

We are interested in approximating in an effective way the law of the minimum of the Brownian motion conditioned to the points $(t_0,x_0),\dots,(t_{n+1},x_{n+1})$, so that particular parameters such as its moments can also be easily estimated. To this end, taking into account the difficulty and length of the analytical computations implied by (2.3), we resort to simulation.

A minimum value for each bridge from (t_i,x_i) to (t_{i+1},x_{i+1}) can be easily simulated from its distribution function $F_{m(B_i)}$, which is explicitly invertible:

$$F^{-1}_{m(B_i)}(z) = \tfrac{1}{2}\left(x_i + x_{i+1} - \left((x_{i+1}-x_i)^2 - 2(t_{i+1}-t_i)\log z\right)^{1/2}\right), \quad z \in (0,1)\,.$$

Since $m(X) = \min\{m(B^0),\dots,m(B^n)\}$, we can simulate a minimum value of X as the minimum of the simulated minima of each bridge. The computational cost is linear in n. At the same time, the relative frequency with which each interval contributes to the global minimum constitutes another way to approximate the probabilities $P\{\theta(X) \in [t_i,t_{i+1}]\}$ of Section 2.3.2. This is what is done in row 6 of Table 2.1 for the interval $[0,t_1]$.

For example, with set 1 of Example 2.3.2, and a sample of size 10 000, we have obtained the following

confidence intervals for the probabilities of each interval to host the minimum:

interval	95% C.I.
$[0, 0.1]$	$[0.3501, 0.3715]$
$[0.1, 0.2]$	$[0.0955, 0.1169]$
$[0.2, 0.5]$	$[0.2362, 0.2576]$
$[0.5, 1]$	$[0.2758, 0.2972]$

The computations have been done in `R` with the `MultinomialCI` package, based on the algorithm of Sison and Glaz [26].

Figure 2.2 shows the result of simulating the minimum of the process in the way described above, conditioned to equispaced points with images equal to zero. The figure includes an histogram and an estimation of the density using the polynomial splines algorithm described in [27], as implemented in the `logspline` package in `R`.

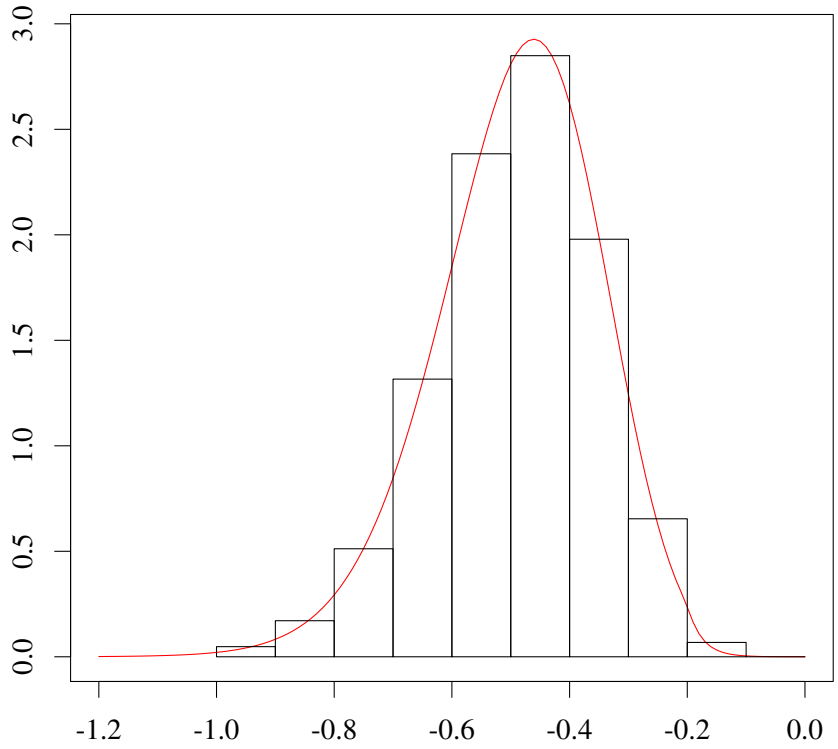

Figure 2.2: Histogram and density estimation of the minimum of a Brownian motion conditioned to pass through the points $\{(k/4, 0),\ k = 0, \ldots, 4\}$, with a sample size of 10000 observations. An asymptotic 95% symmetric confidence interval for the mean yielded $[-0.4939, -0.4884]$. The sample median was -0.4814 .

2.5 Non-adaptive optimisation

Suppose now that we have a black-box optimisation problem in which we can assume that the Brownian bridge is a good probabilistic model for the function at hand. That means, suppose that we are trying to find a point in the interval $[0,1]$ with an image as close as possible to the true minimum of a given but unknown path, drawn at random from the law of a Brownian bridge. The value of the bridge at the end-points is supposed to be given, or that they have already been sampled. It is also assumed that we are allowed to sample the path at a fixed small quantity n of points in $[0,1]$.

A non-adaptive algorithm for this optimisation problem consists in deciding beforehand the n points where we are going to sample the path. They have the same convergence order as the best adaptive algorithm as $n \to \infty$, namely $O(n^{-1/2})$, are much simpler to implement, and offer parallelisation opportunities. Therefore it is worth comparing non-adaptive algorithms in terms of the size of the error incurred for small n. In the next chapter we will discuss and compare some adaptive heuristics.

We will first consider and compare two strategies: Sampling at equidistant points $\frac{k}{n+1}$, $k = 1, \dots, n$, and sampling at random uniformly distributed points. We apply both to a bridge with values 0 and 1 at the end-points and to a symmetric bridge (same value at the end-points). We obtain approximate 95% confidence intervals for the difference between the minimal sampled value and the true minimum of the path. The results are summarised in Table 2.2. Formally, the confidence intervals estimate

$$\mathrm{E}\left[\min_{0\le i\le n+1} B_{t_i} - \min_{t\in[0,1]} B_t\right],$$

where B is the initial bridge joining $(0,x_0)$ and $(1,x_{n+1})$. In one case the t_i are fixed; in the other, they are themselves random.

The procedure for the computations is as follows:

1. Fix the number of points to sample. We have used $n = 2,4,8,16,32,64$ to see the evolution of the intervals when the number of points increases.

2. Sample a path of the Brownian bridge at point $t_1 = 1/(n+1)$; this is done by simulating a value of B_{t_1}, which is easy because its law is Gaussian. Sample at point $t_2 = 2/(n+1)$ the bridge from (t_1,x_1) to $(1,x_{n+1})$. Proceed similarly to get the values of the path at all equidistant points.

3. For the simulation at the n random points in $[0,1]$, determine first at which subinterval of all previously sampled points the new one belongs to, and sample from the corresponding bridge.

	Bridge from $(0,0)$ to $(1,1)$		Bridge from $(0,0)$ to $(1,0)$	
no. of points	95% C.I. eqd	95% C.I. rnd	95% C.I. eqd	95% C.I. rnd
2	$[0.2390, 0.2517]$	$[0.2547, 0.2729]$	$[0.3417, 0.3549]$	$[0.3791, 0.4012]$
4	$[0.2025, 0.2132]$	$[0.2163, 0.2320]$	$[0.2552, 0.2649]$	$[0.3002, 0.3194]$
8	$[0.1659, 0.1745]$	$[0.1759, 0.1891]$	$[0.1944, 0.2023]$	$[0.2183, 0.2317]$
16	$[0.1280, 0.1341]$	$[0.1376, 0.1475]$	$[0.1390, 0.1447]$	$[0.1651, 0.1760]$
32	$[0.0920, 0.0963]$	$[0.1040, 0.1111]$	$[0.0987, 0.1028]$	$[0.1198, 0.1283]$
64	$[0.0663, 0.0694]$	$[0.0778, 0.0838]$	$[0.0712, 0.0741]$	$[0.0851, 0.0910]$

Table 2.2: Approximate confidence intervals for the expectation of the error when estimating the minimum of a Brownian bridge by the minimum of the sampled values, for equidistant ('eqd') and random ('rnd') sampling.

The equidistant points of step 2 and their evaluations are included here so that both methods are in fact applied to the same path.

4. From all the $2n$ sampled points of steps 2 and 3, estimate the expectation of the minimum of the path to which they belong, with the method described in Section 2.4. We have used a simulation of size 1000 in this case, taking the mean of the values obtained.

5. For each sampling strategy, compute the difference between the best sampled point and the estimated minimum of the path.

6. Repeat steps 2–5 a number of times (we used 1000), and construct the asymptotic confidence intervals from the sets of differences obtained, for both strategies.

The results are summarised in Table 2.2 for two different initial bridges. We observe that the equidistant sampling performs better for both. One also observes that the errors are smaller for the non-symmetric bridge; this can be explained by a smaller variance of its minimum value (these variances can be computed analytically from the density (2.2)), despite the fact that many evaluations are possibly wasted in a non-promising region.

The quotient between the estimated errors with equidistant and with random sampling decreases when the number of points increases. Calvin [9] showed that when $n \to \infty$, this quotient approaches ≈ 0.8239. Equidistant sampling is therefore a better choice, if n is really fixed in advance.

A third logical non-adaptive strategy for the non-symmetric bridge is to sample at points that divide $[0,1]$ in intervals that have the same probability to contain the minimum. This is what happens with the equidistant points in the case of a symmetric bridge. We need first to find the points $0 = t_0 < t_1 < \cdots < t_n < t_{n+1} = 1$ such that

$$P\{\theta(B) \in [t_i, t_{i+1}]\} = 1/(n+1) \ , \quad \text{for all } i = 0, \ldots, n \ ,$$

no. of points	95% C.I. eqp
2	$[0.1975, 0.2144]$
4	$[0.1637, 0.1780]$
8	$[0.1275, 0.1387]$
16	$[0.0909, 0.0991]$
32	$[0.0677, 0.0741]$
64	$[0.0491, 0.0538]$

Table 2.3: The third method, dividing $[0,1]$ in intervals of equal probability to host the minimum, for a brige from $(0,0)$ to $(1,1)$.

where B is the bridge from $(0,0)$ to $(1,1)$. This can be done numerically without difficulty with the general formula (2.7) for the density of $\theta(B)$ that we are going to prove in the next section, and that in this case reduces to

$$f_{\theta(B)}(s) = \sqrt{\frac{2(1-s)}{\pi s}} \exp\Big\{\frac{-s}{2(1-s)}\Big\} \mathbf{1}_{\{0\le s\le 1\}} ,$$

With the same number of simulations used in Table 2.2, we get the results of Table 2.3.

Comparing both tables, we see that the expectation of the error for the 'eqp' strategy is the lowest of the three; however, the confidence intervals for the error are longer than with 'eqd', because the variance turns out to be larger. The variances for 'rnd' are the largest.

2.6 Simulating the law of the location of the minimum

The location of the minimum of a continuous function is an ill-posed problem: small changes in the function may result in big changes in the location of the minimum. Therefore, the information about the location of the minimum given by the sampled values is limited, and possibly of less practical importance than the information about the minimum value. Anyway, we can try to visualise this information through the law of $\theta(X) = \arg\min_{[0,1]} X_t$.

This law can be simulated with the auxiliary use of the minima of all bridges $B^0, \dots, B^n$, which in turn can be easily simulated as we have seen in Section 2.4. We prove first that conditioned to all these minima, the variables $\theta(X)$ and $\theta(B^j)$, where j is the index of the interval where the global minimum is attained, have the same law. This is the contents of the next proposition:

Proposition 2.6.1. *Denote* $x^* := \min_{0\le i\le n+1} x_i$ *and* $\Psi_j := \{y = (y_0, \dots, y_n) \in [-\infty, x^*]^{n+1} : y_j = \min_i y_i\}$. *For any Borel set* $A \subset [0,1]$,

$$P\big\{\theta(X) \in A \,/ m(B^0) = y_0, \dots, m(B^n) = y_n\big\} = P\big\{\theta(B^j) \in A \,/ m(B^j) = y_j\big\}$$

on Ψ_j almost everywhere with respect to Lebesgue measure.

Proof. First, we prove the equality

$$P\big\{\theta(X)\in A \,\big/ m(B^0)=y_0,\dots,m(B^n)=y_n\big\} = P\big\{\theta(B^j)\in A \,\big/ m(B^0)=y_0,\dots,m(B^n)=y_n\big\}\ .$$

on Ψ_j almost everywhere with respect to Lebesgue measure (y-a.e. on Ψ_j for short).

It is clear that the support of $\theta(X)$ and of $\theta(B^j)$ with respect to the conditional law is the interval $I_j := [t_j, t_{j+1}]$, y-a.e. on Ψ_j; therefore, we can assume $A \subset I_j$.

Denoting $s^* := \theta(X)$, since $X_s = B^j_s$, $\forall s \in I_j$, and $s^* \in I_j$ almost surely with respect to the conditional law, y-a.e. on Ψ_j, we have $m(X) = X_{s^*} = B^j_{s^*} = m(B^j)$. This implies $\theta(B^j) = s^*$ almost surely, y-a.e. on Ψ_j, due to the almost sure uniqueness of the location of the minimum of a Brownian bridge.

Finally, the equality

$$P\big\{\theta(B^j)\in A \,\big/ m(B^1)=y_1,\dots,m(B^n)=y_n\big\} = P\big\{\theta(B^j)\in A \,\big/ m(B^j)=y_j\big\}\ .$$

comes from the independence of B^j from all other bridges.

□

From Proposition 2.6.1, we see that to simulate the location of the minimum of X it is enough to simulate the minima of all bridges, select the lowest of them y_j, and then simulate the location $\theta(B^j)$ of the minimum of the bridge B^j conditioned only to $m(B^j) = y_j$. We need first the law of the vector $(m(B^j), \theta(B^j))$. This is stated in the next proposition. We also give the marginal law of $\theta(B^j)$, since we have not been able to find it in the literature in this generality, even though it will not be used directly in the simulation.

Proposition 2.6.2. *Writing $\ell := t_{j+1} - t_j$ and $d := |x_{j+1} - x_j|$, the minimum of the bridge B^j from (t_j, x_j) to (t_{j+1}, x_{j+1}) and its location have the joint density*

$$f_{(m(B^j),\theta(B^j))}(y,s) = \frac{(x_j-y)(x_{j+1}-y)\sqrt{2\ell}}{\sqrt{\pi(s-t_j)^3(t_{j+1}-s)^3}} \exp\Big\{\frac{d^2}{2\ell} - \frac{(x_j-y)^2}{2(s-t_j)} - \frac{(x_{j+1}-y)^2}{2(t_{j+1}-s)}\Big\}\mathbf{1}_{\{y<x_j, y<x_{j+1}, t_j\le s\le t_{j+1}\}}\ , \tag{2.6}$$

and the density of the location is

$$f_{\theta(B^j)}(s) = \Big[\frac{d}{\ell^{3/2}}\sqrt{\frac{2}{\pi h(s)}}\exp\Big\{\frac{-d^2}{2\ell}h(s)\Big\} + \frac{\ell-d^2}{\ell^2}\operatorname{erfc}\Big\{\Big(\frac{d^2}{2\ell}h(s)\Big)^{1/2}\Big\}\Big]\mathbf{1}_{\{t_j\le s\le t_{j+1}\}}\ , \tag{2.7}$$

where

$$h(s) = \begin{cases} (t_{j+1}-s)/(s-t_j)\,, & \text{if } x_{j+1} \le x_j \\ (s-t_j)/(t_{j+1}-s)\,, & \text{if } x_j \le x_{j+1} \end{cases}$$

and erfc( ) *is the complementary error function* $1-\mathrm{erf}(\,)$.

Proof. The joint law of W_t, with W a standard Brownian motion $W=\{W_s,\ s\ge 0\}$ starting at $W_0=a$, its minimum m_t up to time t and the location θ_t of this minimum, is known to have the density

$$\begin{aligned} &P_a\{W_t\in db,\ m_t\in dy,\ \theta_t\in ds\} = \\ &\frac{(a-y)(b-y)}{\pi\sqrt{s^3(t-s)^3}}\exp\Big\{-\frac{(a-y)^2}{2s}-\frac{(b-y)^2}{2(t-s)}\Big\}\mathbf{1}_{\{y<a,\ y<b,\ 0\le s\le t\}} \end{aligned} \tag{2.8}$$

(see Karatzas-Shreve [16, Prop. 2.8.15], or Csáki et al. [13], where it is extended to general diffusions); the formula is usually stated for the maximum, but (2.8) is easily deduced by symmetry, taking into account that $-W$ is a Brownian motion starting at $-a$.

Consequently, if W starts instead at time $u<t$, we have

$$\begin{aligned} &P_{(u,a)}\{W_t\in db,\ m_t\in dy,\ \theta_t\in ds\} = \\ &\frac{(a-y)(b-y)}{\pi\sqrt{(s-u)^3(t-s)^3}}\exp\Big\{-\frac{(a-y)^2}{2(s-u)}-\frac{(b-y)^2}{2(t-s)}\Big\}\mathbf{1}_{\{y<a,\ y<b,\ u\le s\le t\}}\,. \end{aligned}$$

Conditioning to $\{W_t=b\}$, one finds the joint density of the minimum m and its location θ for a Brownian bridge B joining the points (u,a) and (t,b):

$$\begin{aligned} &P_{(u,a),(t,b)}\{m\in dy,\ \theta\in ds\} = \\ &\frac{(a-y)(b-y)\sqrt{2(t-u)}}{\sqrt{\pi(s-u)^3(t-s)^3}}\exp\Big\{\frac{(b-a)^2}{2(t-u)}-\frac{(a-y)^2}{2(s-u)}-\frac{(b-y)^2}{2(t-s)}\Big\}\mathbf{1}_{\{y<a,\ y<b,\ u\le s\le t\}}\,, \end{aligned}$$

which is equivalent to (2.6).

Integrating out y, we get, if $a<b$,

$$\begin{aligned} P_{(u,a),(t,b)}\{\theta\in ds\} = \Big[&\frac{b-a}{(t-u)^2}\sqrt{\frac{2(t-u)(t-s)}{\pi(s-u)}}\exp\Big\{\frac{-(b-a)^2(t-s)}{2(t-u)(s-u)}\Big\} \\ &+\frac{(t-u)-(b-a)^2}{(t-u)^2}\,\mathrm{erfc}\Big\{(b-a)\sqrt{\frac{t-s}{2(t-u)(s-u)}}\Big\}\Big]\mathbf{1}_{\{u\le s\le t\}}\,, \end{aligned}$$

and, in case $a > b$,

$$P_{(u,a),(t,b)}\{\theta \in ds\} = \Big[\frac{a-b}{(t-u)^2} \sqrt{\frac{2(t-u)(s-u)}{\pi(t-s)}} \exp\Big\{ \frac{-(b-a)^2(s-u)}{2(t-u)(t-s)} \Big\} + \frac{(t-u)-(b-a)^2}{(t-u)^2} \operatorname{erfc}\Big\{ (a-b)\sqrt{\frac{s-u}{2(t-u)(t-s)}} \Big\} \Big] \mathbf{1}_{\{u \le s \le t\}} \,,$$

from where we get (2.7). □

Proposition 2.6.3. *The location of the minimum $\theta(B_j)$ conditioned to $m(B_j)$ has a density of the form*

$$f_{\theta(B_j)|_{m(B_j)=y}}(s) = C(y)(s-t_j)^{-3/2}(t_{j+1}-s)^{-3/2} \exp\Big\{ -\frac{A(y)}{2(s-t_j)} - \frac{B(y)}{2(t_{j+1}-s)} \Big\} \cdot \mathbf{1}_{\{t_j \le s \le t_{j+1}\}} \quad (2.9)$$

for $y < x_j$, $y < x_{j+1}$, where A, B and C are positive constants depending only on y.

Proof. This is an immediate computation from the joint density (2.6) and the marginal (2.2), yielding (2.9) with

$$A(y) = (x_j - y)^2 \,, \quad B(y) = (x_{j+1} - y)^2 \,,$$
$$C(y) = \frac{(t_{j+1}-t_j)^{3/2}(x_j - y)(x_{j+1} - y)}{\sqrt{2\pi}(x_j + x_{j+1} - 2y)} \exp\Big\{ \frac{(x_{j+1} + x_j - 2y)^2}{2(t_{j+1}-t_j)} \Big\} \,.$$

□

Notice that the density (2.7) is not bounded as $h(s)$ goes to zero at one of the end-points of the interval $[t_j, t_{j+1}]$. Therefore, it is not easy to sample exactly from it. However, the conditional density (2.9) given a value $y < \min\{x_j, x_{j+1}\}$ it is bounded, which makes it more amenable to the acceptance/rejection method (see, e.g. Asmussen and Glynn [5]). The result is a sample from the joint density of the minimum and its location, from where one obtains a sample of the marginal law of the location. This trick, together with Proposition 2.6.1, will allow us to simulate the location of the minimum of the whole process X, conditioned to pass through the given set of points.

To apply acceptance/rejection by comparison with a uniform distribution, the global maximum of the function (2.9) should be easily calculated or approximated from above. This is indeed the case: There are two obvious minima at the end-points t_j and t_{j+1}; the remaining extremal points are the roots of the 3-degree polynomial

$$3C(y)\big[(s-t_j)^2(t_{j+1}-s) - (s-t_j)(t_{j+1}-s)^2\big] + A(y)(t_{j+1}-s)^2 - B(y)(s-t_j)^2 \,,$$

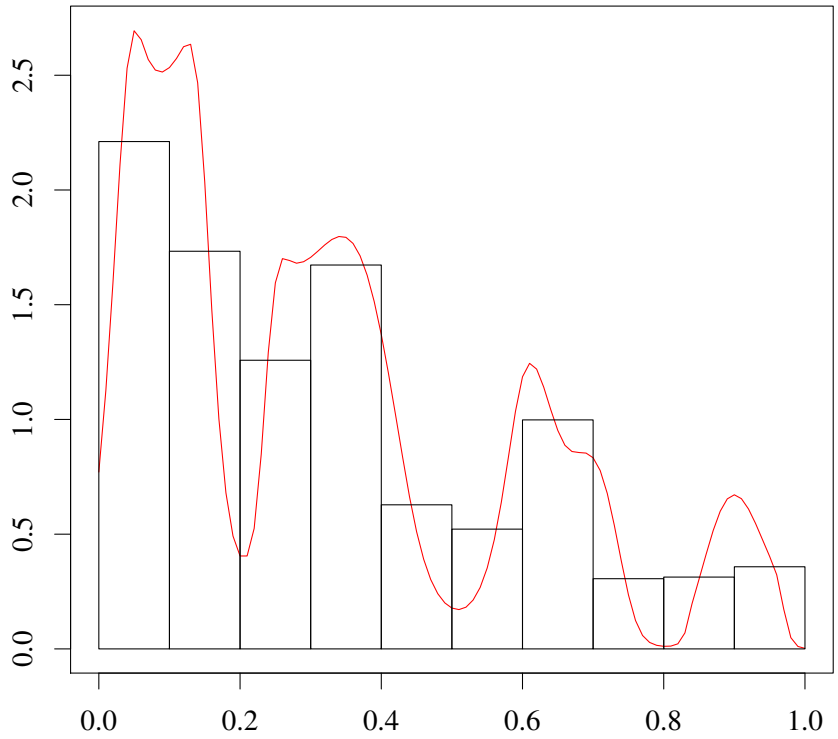

Figure 2.3: Density estimation for the location of the minimum of the Brownian bridge conditioned to the points $(0,0),(0.2,0.06),(0.5,0.16),(0.8,0.26),(1,0.20)$.

as can be seen by differentiating in s and multiplying by $2(s-t_j)^{7/2}(t_{j+1}-s)^{7/2}$. This polynomial may have one or three real roots, corresponding to a unique maximum, or to two maxima, with a minimum in between. In any case, the global maximum can be computed exactly and the acceptance/rejection method can be implemented for this density.

In Figure 2.3, we see the result of a simulation of size 10 000, with a density estimation using the `logsplines` method in `R`. As it was remarked in Section 2.2, once we are considering the minimum of the whole process, the different bridges are no longer independent; in particular, the shape of the density in each subinterval is not the one to be expected from formula (2.7), and in fact it is quite difficult to predict from the conditioning values. Hence the interest to have an exact simulation method.

As a more clear example of the last remark, consider the concatenation of two symmetric bridges, from $(0,0)$ to $(0.5,0)$, and from $(0.5,0)$ to $(1,0)$. Separately, the location of their minima follows a uniform distribution; however, the location of the global minimum follows the density simulated in Figure 2.4. The fact that the minimum of the two minima tends to take a lower value than a single minimum drifts away its location from the end-points of the subintervals.

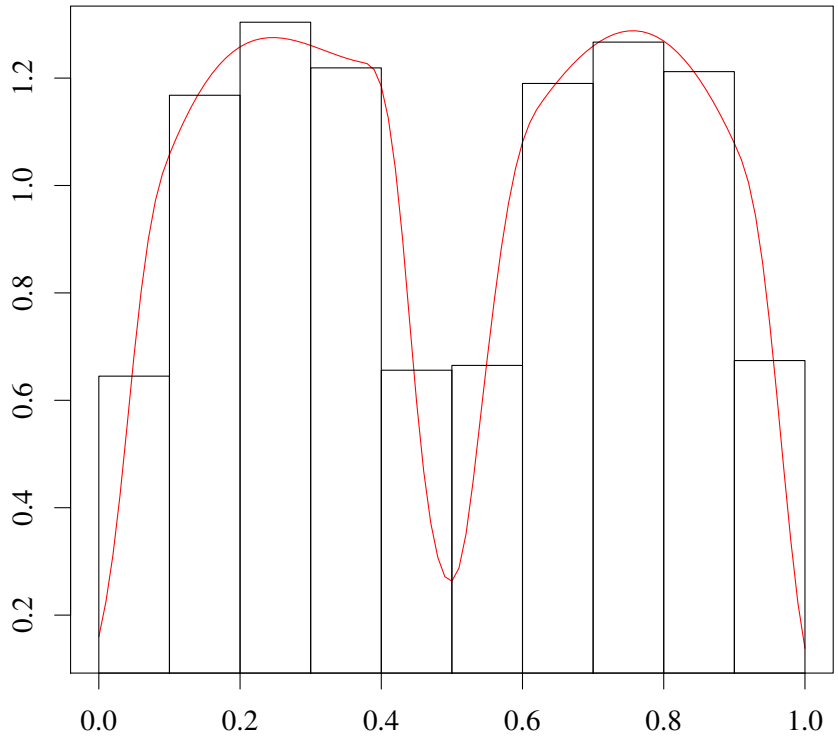

Figure 2.4: Density estimation for the location of the global minimum of two concatenated symmetric identical Brownian bridges.

2.7 Acknowledgements

This work has been supported by grants numbers MTM2011-29064-C03-01 from the Ministry of Economy and Competitiveness of Spain; and UNAB10-4E-378, co-funded by the European Regional Development Fund (ERDF).

Capítol 3

Algorismes adaptatius d'un pas per cercar el mínim del pont brownià

Abstract

Donat un pont brownià B a l'interval $[0,1]$, com a model probabilístic d'un problema d'optimització tipus "caixa negra", estudiem i comparem diverses heurístiques adaptatives per aproximar el mínim de la trajectòria. Se suposa que només podrem mostrejar el procés en pocs punts $t_i \in (0,1)$; no estem interessats per tant aquí en les propietats asimptòtiques dels algorismes, sinó en la seva eficiència amb poques iteracions.

Keywords: Black-box optimisation, Brownian bridge, simulation, adaptive algorithms.

Mathematics Subject Classification (2010): 90C26, 60J65, 65C05.

3.1 Introducció

Al capítol anterior estudiàvem lleis de probabilitat relacionades amb el pont brownià condicionat a uns quants punts interiors de l'interval de definició, i les usàvem per comparar tres estratègies no-adaptatives l'objectiu de les quals és aproximar-se al mínim de la trajectòria del pont.

En aquest capítol veurem algunes estratègies adaptatives. En principi, donada una quantitat fixada de punts a mostrejar, és teòricament possible fer el plantejament típic de la programació dinàmica: El primer punt a mostrejar és aquell que optimitza el valor òptim dels punts que es mostrejaran després (principi d'optimalitat de Bellman). Havent-hi aleatorietat pel mig, la condició de "millor" punt a mostrejar s'ha de decidir d'alguna manera que no el faci dependre de les diferents possibilitats de l'atzar.

Usualment, la influència de l'atzar s'elimina prenent esperances, tot i que en alguna situació pràctica poden ser convenients altres possibilitats.

Per exemple, si es volen mostrejar dos punts en total, el punt t_1 òptim seria

$$t_1 := \arg\min \mathrm{E}\left[\min_{t_2} \mathrm{E}\left[L(X_{t_2}, m(X)) \,/_{X_{t_1}}\right]\right].$$

A la pràctica, això vol dir: Per a cada possible elecció t_1, trobar el t_2 que minimitza l'esperança condicionada interior, resultant en una funció de (t_1, X_{t_1}), i després minimitzar en t_1 l'esperança d'aquesta funció. Per altra banda, L és alguna funció de pèrdua que representi una avaluació de l'aproximació obtinguda. Per exemple, $X_{t_2} - m(X)$ o $(X_{t_2} - m(X))^2$ poden ser eleccions naturals.

És bastant evident que la dificultat computacional creix exponencialment amb el nombre de punts a mostrejar. Bellman anomenava això la maledicció de la dimensionalitat (*the curse of dimensionality*). De fet, en el nostre cas, dos punts ja representen un repte considerable.

Un plantejament suboptimal més pràctic és decidir només el següent punt a mostrejar, prescindint dels que caldrà escollir després. En el camp de l'optimització aquest tipus estratègies s'anomenen *golafres* (greedy algorithms). En l'àmbit dels models probabilístics, alguns autors les donen el nom de *one-step Bayesian methods*. En aquest marc, el n-èsim punt s'escull com aquell on s'assoleix

$$\min_{t_n} \mathrm{E}\left[L(X_{t_n}, m(X)) \,/_{X_{t_1}, \ldots, X_{t_{n-1}}}\right],$$

o amb alguna estratègia heurística que substitueixi el càlcul exacte d'aquest mínim. Un avantatge dels algorismes golafres per a aquest problema és que no hi ha necessitat de fixar a priori la quantitat de punts que es mostrejaran. Igual que en el capítol anterior, no estem interessats en les propietats asimptòtiques dels algorismes, sinó en comparar diverses idees quan la funció desconeguda que es vol minimitzar es pot modelar, almenys a una escala no massa fina, com un pont brownià, i l'avaluació de la funció és massa cara per fer-ne un mostreig massiu.

Subratllem que les funcions de pèrdua no tenen per què dependre precisament del valor X_{t_n} i del mínim del procés $m(X)$. Si denotem, com en el capítol anterior, $\theta(X)$ la variable aleatòria que dóna l'argument del mínim, podem estar interessats en acostar-nos a aquest punt, i una mesura de l'acostament podria ser $t - \theta(X)$, escollint el t per tal que l'esperança d'aquest diferència sigui mínima.

Així, més en general, es poden considerar funcions de pèrdua $L(s,x,y)$ relacionades de diverses maneres amb les variables $s = \theta(X)$, $x = X_t$, $y = m(B)$, com ara $|t-s|$, $(t-s)^2$, $x-y$, $(x-y)^2$, i d'altres.

Comencem amb un apartat de preliminars sobre fets bàsics del moviment brownià i el pont brownià; moltes d'aquestes propietats s'han usat ja en el capítol anterior. Després, a l'Apartat 3, considerem les funcions de pèrdua $|t-s|$ i $(t-s)^2$, en relació a un sol pont brownià. Als dos apartats següents considerem diverses heurístiques d'optimització relacionades amb aquestes funcions. A l'Apartat 4 estudiem diversos mètodes per seleccionar un subinterval del procés condicionat X i hi busquem els òptims de les funcions de pèrdua anteriors. En l'Apartat 5, en canvi, la idea és trobar aquest òptim dins de tots els subintervals determinats pels punts mostrejats anteriorment, i després escollir entre ells el millor candidat a ser mostrejat.

A l'Apartat 6 pensem en les funcions de pèrdua que involucren la distància entre el valor del procés en el punt a mostrejar i el valor del mínim d'un pont brownià B. Per això ens cal calcular la llei conjunta de $(B_t, m(B))$. Aquest càlcul ocupa bona part de l'apartat. Finalment, veiem com minimitzar $E[B_t - m(B)]$ o $\mathrm{E}\left[(B_t - m(B))^2\right]$ no porta a cap algorisme efectiu, i en canvi sí ho fa la minimització de $\mathrm{E}\left[\min(B_t,0) - m(B)\right]$.

En el que segueix, quan no hi hagi confusió, abreviem el mínim d'un procés $m := m(X)$ i l'argument del mínim $\theta := \theta(X)$.

3.2 Preliminars

3.2.1 Definicions i caracteritzacions del pont brownià

El pont brownià és un brownià condicionat a un punt final. Comencem amb les definicions i propietats del brownià que farem servir més endavant.

Definició 3.2.1. *Un procés estocàstic W_t és un procés de Wiener (o moviment brownià) si satisfà les següents condicions:*

1. $P\{\omega : W_0(\omega) = 0\} = 1$.
2. *Per a tot $0 \le s < t$, la variable $W_t - W_s$ té distribució normal de mitjana zero i variància $t-s$.*
3. *W_t té increments independents.*
4. *Totes les trajectòries de W_t són funcions contínues.*

(vegeu, per exemple, [18]).

Definició 3.2.2. *Sigui $X = \{X_t, t \in [0,1]\}$ un procés estocàstic. Es defineixen el temps de passatge per*

$$T_b := \inf\{t \in [0,1]; X_t = b\} ,$$

el mínim del procés en $[0,t]$ per

$$m_t := \min_{s\in[0,t]} X_s ,$$

el primer instant on s'assoleix el mínim per

$$\theta_t := \inf\{0 \leq s \leq t; X_s = m_t\} ,$$

el màxim del procés en $[0,t]$ per

$$M_t := \max_{s\in[0,t]} X_s ,$$

i el primer instant on s'assoleix el màxim per

$$\tau_t := \inf\{0 \leq s \leq t; X_s = M_t\} .$$

Especifiquem l'estructura de probabilitat de transició gaussiana que utilitzarem més endavant per a la definició del pont brownià.

Definició 3.2.3. *Una família d'aplicacions $p(t,\cdot,\cdot) : \mathbb{R} \times \mathscr{B}(\mathbb{R}) \to [0,1]$ amb $0 \leq t < \infty$, és una probabilitat de transició gaussiana a $(\mathbb{R}, \mathscr{B}(\mathbb{R}))$ si:*

1. *$\forall x \in \mathbb{R}$, l'aplicació*

$$\begin{array}{rccc} p(t,x,\cdot): & \mathscr{B}(\mathbb{R}) & \longrightarrow & [0,1] \\ & B & \mapsto & p(t;x,B) \end{array}$$

   és una llei normal $\mathscr{N}(x,t)$.

2. *$\forall B \in \mathscr{B}(\mathbb{R})$, l'aplicació*

$$\begin{array}{rccc} p(t;\cdot,B): & \mathbb{R} & \longrightarrow & [0,1] \\ & x & \mapsto & p(t;x,B) \end{array}$$

   és mesurable.

La probabilitat de transició gaussiana de x a y té densitat, donada per la següent expressió (abusant una mica la notació):

$$p(t;x,y) := \frac{1}{\sqrt{2\pi t}} \exp\Big\{ -\frac{(x-y)^2}{2t} \Big\} .$$

Proposició 3.2.1. *Si W_t és un moviment brownià estàndard, aleshores $\hat{W}_t = -W_t$ és també un brownià estàndard.*

Demostració. $\hat{W}_t = -W_t$ té increments independents amb la mateixa distribució que els de W_t i té també trajectòries contínues. □

Definició 3.2.4. *El procés X_t és un pont brownià de $a \in \mathbb{R}$ fins a $b \in \mathbb{R}$ en [0,T], denotat per $B_t^{(0,a)(T,b)}$, si és un procés continu definit en $[0,T]$, amb les següents distribucions en dimensió finita:*

$$P(X_{t_1} \in dx_1, ..., X_{t_n} \in dx_n) = \prod_{i=1}^{n} p(t_i - t_{i-1}; x_{i-1}, x_i) \frac{p(T - t_n; x_n, b)}{p(T; a, b)} dx_1 ... dx_n \ .$$

on $0 = t_0 < t_1 < ... < t_n < T, x_0 = a, (x_1, ..., x_n) \in \mathbb{R}^n$ i $p(t; x, y)$ és la densitat de transició gaussiana de x a y.

Notació. Seguirem la següent notació:

- W_t és un moviment brownià.
- $W_t^{(t_1,x_1)}$ és un moviment brownià que comença en $t_1 \in \mathbb{R}_+$ amb $W_{t_1} = x_1$.
- $B_t^{(t_1,x_1)(t_2,x_2)}$ és un pont brownià que comença en $t_1 \in \mathbb{R}_+$ amb $B_{t_1}^{(t_1,x_1)(t_2,x_2)} = x_1$ i acaba en t_2 amb $B_{t_2}^{(t_1,x_1)(t_2,x_2)} = x_2$.
- $B_t^{(t_1,x_1)'}$ és un pont brownià que comença a $(0,0)$ i acaba en t_1 amb $W_{t_1} = x_1$.

El pont brownià estàndard $X = \{X_t : t \in [0,1]\}$ és el que va de $(0,0)$ a $(1,0)$ i compleix les següents propietats, que es dedueixen de la definició anterior 3.2.4:

- $X_0 = 0$ i $X_1 = 0$ amb probabilitat 1.
- X és un procés gaussià.
- $E[X_t] = 0$ per a $t \in [0,1]$.
- $Cov[X_s, X_t] = \min\{s,t\} - st$ per a $s,t \in [0,1]$.
- Amb probabilitat 1, $t \to X_t$ és continu a $[0,1]$.

Aquestes propietats caracteritzen el pont brownià estàndard.

Proposició 3.2.2. *Sigui $Z = \{Z_t : t \in [0,1]\}$ un moviment brownià estàndard, i sigui $X_t = Z_t - tZ_1$ per a $t \in [0,1]$. Llavors $X = \{X_t : t \in [0,1]\}$ és un pont brownià estàndard.*

Demostració.

- $X_0 = Z_0 = 0$ i $X_1 = Z_1 - Z_1 = 0$.
- Per combinació lineal és un procés gaussià.
- $E[X_t] = E[Z_t] - tE[Z_1] = 0$, per a $t \in [0,1]$.
- $\mathrm{Cov}[X_s, X_t] = \mathrm{Cov}(Z_s - sZ_1, Z_t - tZ_1) = \mathrm{Cov}(Z_s, Z_t) - t\,\mathrm{Cov}(Z_s, Z_1) - s\,\mathrm{Cov}(Z_1, Z_t) + st\,\mathrm{Cov}(Z_1, Z_1) = \min\{s,t\} - st$ per a $s,t \in [0,1]$.
- Amb probabilitat 1, $t \to X_t$ és continu a $[0,1]$.

□

Notem que en el cas del pont brownià estàndard X, X_t té una distribució normal d'esperança 0 i variància $t(t-1)$ per a $t \in [0,1]$. Així doncs, la variància creix i decreix a l'interval $[0,1]$ agafant el màxim de $1/4$ a $t = 1/2$.

Proposició 3.2.3. *Sigui $Z = \{Z_t : t \in [0,\infty)\}$ un moviment brownià estàndard. Definim $X_1 = 0$ i $X_t = (1-t)Z(\frac{t}{1-t})$ per a $t \in [0,1)]$. Llavors $X = \{X_t : t \in [0,1]\}$ és un pont brownià estàndard.*

Demostració.

- $X_0 = Z_0 = 0$ per definició i $X_1 = 0$.
- Per combinació lineal és un procés gaussià.
- $E[X_t] = (1-t)E[Z(\frac{t}{1-t})] = 0$, per a $t \in [0,1]$.
- Si $s,t \in [0,1)]$ amb $s < t$ aleshores $s/(1-s) < t/(1-t)$ i per tant

  $\mathrm{Cov}[X_s, X_t] = \mathrm{Cov}[(1-s)Z(\frac{s}{1-s}), (1-t)Z(\frac{t}{1-t})] = (1-s)(1-t)\frac{s}{1-s} = s(1-t)$.
- Amb probabilitat 1, $t \to X_t$ és continu a $[0,1]$ i amb probabilitat 1, $X_t = (1-t)Z(\frac{t}{1-t}) \to 0$ quan $t \downarrow 0$.

□

Proposició 3.2.4. *Sigui $X = \{X_t : t \in [0,1]\}$ un procés brownià estàndard. Si condicionem per $X_1 = 0$, el procés $\{X_t : t \in [0,1]\}$ és un pont brownià estàndard.*

Demostració. Es basa en les propietats de la normal multidimensional. □

Proposició 3.2.5. *El procés $B_t = a(1-t) + bt + W_t - W_1 t$ és un pont brownià $B_t^{(0,a)(1,b)}$.*

Demostració. S'aplica la proposició 3.2.2. □

Proposició 3.2.6. *Si $B_t^{(t_0,x_0)(t_1,x_1)}$ és un pont brownià, aleshores $-B_t^{(t_0,x_0)(t_1,x_1)} \sim B_t^{(t_0,-x_0)(t_1,-x_1)}$*

Demostració. Segueix de les proposicions 3.2.1 i 3.2.5. □

El pont brownià en un punt segueix una llei gaussiana fàcilment deduïble:

Proposició 3.2.7. *El pont brownià $B_t^{(t_0,x_0)(t_1,x_1)}$ té funció de densitat*

$$f(y) = \frac{1}{\sqrt{2\pi \frac{(t-t_0)(t_1-t)}{t_1-t_0}}} \exp\left\{ -\frac{(x_0 + \frac{(t-t_0)(x_1-x_0)}{t_1-t_0} - y)^2}{2\frac{(t-t_0)(t_1-t)}{t_1-t_0}} \right\}.$$

3.2.2 Densitats conjuntes del moviment brownià amb el màxim i el mínim

Proposició 3.2.8. *La distribució conjunta de (W_t, M_t) té la següent densitat:*

$$f_{W,M}(x,y) = \sqrt{\frac{2}{\pi}} \frac{(2y-x)}{t^{3/2}} \exp\Big\{ -\frac{(2y-x)^2}{2t} \Big\} \mathbb{1}_{\{x<y\}} \mathbb{1}_{\{y>0\}} .$$

Demostració. Sigui $y > 0, x < y$, $\widehat{W}_t = 2y - W_t$ el moviment brownià reflexat a y i $\Phi(z)$ la funció de distribució de la llei normal estàndard. Aleshores:

$$\begin{aligned}
P(W_t \le x, M_t \ge y) &= P(T_y \le t, W_t \le x) \quad \text{(ja que } \{M_t \ge y\} = \{T_y \le t\}) \\
&= P(T_y \le t, \widehat{W}_t \ge 2y - x) \\
&= P(T_y \le t, W_t \ge 2y - x) \quad \text{(perquè } T_y \text{ és el mateix per a } W_t \text{ i per a } \widehat{W}_t) \\
&= P(W_t \ge 2y - x) \quad \text{(perquè } y - x > 0, \text{ i } \{W_t \ge 2y - x\} \subset \{T_y \le t\}) \\
&= 1 - \Phi\Big(\frac{2y-x}{\sqrt{t}}\Big).
\end{aligned}$$

I ara es deriva. □

Corollari 3.2.9. *M_t té la següent densitat:*

$$f_M(y) = \sqrt{\frac{2}{\pi t}} e^{\frac{-y^2}{2t}} \mathbb{1}_{\{y>0\}}.$$

Demostració.

$$f_M(y) = \int_{-\infty}^{\infty} f_{W,M}(x,y)dx = \int_{-\infty}^{y} \sqrt{\frac{2}{\pi}} \frac{(2y-x)}{t^{3/2}} \exp\Big\{ -\frac{(2y-x)^2}{2t} \Big\} dx = \sqrt{\frac{2}{\pi t}} e^{\frac{-y^2}{2t}} .$$

□

Observació. La densitat de M_t coincideix amb la del valor absolut d'una normal $\mathcal{N}(0,t)$.

Corollari 3.2.10. *La distribució conjunta de (W_t, m_t) té la següent densitat:*

$$f_{W,m}(x,y) = \sqrt{\frac{2}{\pi}} \frac{(x-2y)}{t^{3/2}} \exp(-\frac{(x-2y)^2}{2t}) \mathbb{1}_{\{y<0\}} \mathbb{1}_{\{x>y\}}.$$

Demostració. Sigui $W'_t = -W_t$ també un moviment brownià per les propietats de simetria. Aleshores com que $\min_{s\leq t} W_s = -\max_{s\leq t} -W_s$ tenim que $P(W_t \geq x, \min_{s\leq t} W_s \leq y) = P(W'_t \leq -x, \max_{s\leq t} -W'_s \geq -y) = 1 - \Phi\Big(\frac{-2y+x}{\sqrt{t}}\Big)$. I ara es deriva. □

Corollari 3.2.11. *m_t té la següent densitat:*

$$f_m(y) = \sqrt{\frac{2}{\pi t}} e^{\frac{-y^2}{2t}} \mathbb{1}_{\{y<0\}}.$$

Demostració.

$$f_m(y) = \int_{-\infty}^{\infty} f_{W,M}(x,y)dx = \int_{y}^{\infty} \sqrt{\frac{2}{\pi}} \frac{(x-2y)}{t^{3/2}} \exp\Big\{ -\frac{(x-2y)^2}{2t} \Big\} dx = \sqrt{\frac{2}{\pi t}} e^{\frac{-y^2}{2t}} .$$

□

Observem que les lleis del màxim i del mínim del moviment brownià són essencialment iguals, per simetria.

Corollari 3.2.12. *El mínim m'_t del pont brownià $B_t^{(t_1,x_1)'}$ té la següent densitat:*

$$f_{m'}(y) = \frac{2}{t_1}(x_1 - 2y) \exp\Big\{ \frac{2x_1 y - 2y^2}{t_1} \Big\} \mathbb{1}_{\{y<x_1\}} \mathbb{1}_{\{y<0\}}.$$

Demostració. Condicionem per $\{W_{t_1} = x_1\}$ la densitat conjunta del corol·lari 3.2.10:

$$\begin{aligned} f_{m'}(y) &= \frac{2}{t_1}(x_1 - 2y)\exp\left\{\frac{{x_1}^2 - (x_1-2y)^2}{2t_1}\right\}\mathbb{1}_{\{y<x_1\}}\mathbb{1}_{\{y<0\}} \\ &= \frac{2}{t_1}(x_1 - 2y)\exp\left\{\frac{2x_1y - 2y^2}{t_1}\right\}\mathbb{1}_{\{y<x_1\}}\mathbb{1}_{\{y<0\}}\ . \end{aligned}$$

□

Quan aquesta expressió apareix en una integral més endavant, va millor utilitzar la primera igualtat.

Corol·lari 3.2.13. *El mínim m'_t del pont brownià $B_t^{(t_1,x_1)(t_2,x_2)'}$ té la següent densitat:*

$$f_{m'}(y) = \frac{2}{t_2 - t_1}(x_2 + x_1 - 2y)\exp\left\{\frac{-2(x_1-y)(x_2-y)}{t_2-t_1}\right\}\mathbb{1}_{\{y<x_1\}}\mathbb{1}_{\{y<x_2\}}\ .$$

Demostració. Utilitzem el corol·lari anterior 3.2.12:

$$\begin{aligned} f_{minB^{(0,x_1)(t_2,x_1+x_2)}}(y) =& f_{minB^{(0,0)(t_2,x_2)}}(y - x_1) \\ =& \frac{2}{t_2}(x_2 - 2(y-x_1))\exp\left\{(\frac{2x_2(y-x_1) - 2(y-x_1)^2}{t_2}\right\}\mathbb{1}_{\{y-x_1<x_2\}}\mathbb{1}_{\{y-x_1<0\}} \\ =& \frac{2}{t_2}(x_2 + x_1 + x_1 - 2y)\exp\left\{(\frac{2(x_2+x_1)(y-x_1) - 2x_1(y-x_1) - 2(y-x_1)^2}{t_2}\right\} \\ & \cdot \mathbb{1}_{\{y<x_1+x_2\}}\mathbb{1}_{\{y<x_1\}}\ . \end{aligned}$$

I canviant la notació,

$$\begin{aligned} f_{minB^{(0,x_1)(t_2,x_2)}}(y) &= \frac{2}{t_2}(x_2 + x_1 - 2y)\exp\left\{\frac{2x_2(y-x_1) - 2x_1(y-x_1) - 2(y-x_1)^2}{t_2 - t_1})\right\}\mathbb{1}_{\{y<x_2\}}\mathbb{1}_{\{y<x_1\}} \\ &= \frac{2}{t_2}(x_2 + x_1 - 2y)\exp\left\{\frac{-2(x_1-y)(x_2-y)}{t_2}\right\}\mathbb{1}_{\{y<x_2\}}\mathbb{1}_{\{y<x_1\}}\ . \end{aligned}$$

□

3.3 Funcions de pèrdua involucrant t i $\theta(X)$

Suposem que tenim un pont brownià B en $[0,1]$ i volem avaluar el pont en un punt $t \in (0,1)$, de forma que ens aproximem el més possible, segons un cert criteri, al vertader argument del mínim $\theta := \theta(B)$.

Dos possibles criteris venen donats per les funcions de pèrdua $|t-\theta(B)|$ i $(t-\theta(B))^2$. La deducció teòrica del punt t que minimitza l'esperança d'aquestes funcions no és difícil, i el resultat és esperable.

Comencem amb la primera. Si $f(s)$ denota la densitat de l'argument del mínim, la funció a minimitzar respecte t és la integral de $|t-s|$:

$$g(t) = \int_0^1 |t-s| f(s) ds = \int_0^t (t-s) f(s) ds + \int_t^1 (s-t) f(s) ds \ .$$

Si derivem sota el signe de la integral i igualem a zero obtenim:

$$\begin{aligned} g'(t) &= tf(t) + \int_0^t f(s) ds - tf(t) - tf(t) - \int_t^1 f(s) ds + tf(t) \\ &= \int_0^t f(s) ds - \int_t^1 f(s) ds = 0. \end{aligned}$$

Per tant estem buscant el t tal que

$$\int_0^t f(s) ds = \int_t^1 f(s) ds,$$

i com que $f(s)$ és una densitat, t és la mediana de θ, que denotarem $\mathbb{M}[\theta]$.

Aquesta mediana la podem calcular numèricament amb `maxima`. Sigui θ l'argument del mínim del pont, mitjançant la rutina `quad_qaqs` es calcula la probabilitat $P_s(\theta \in [0,s])$ i després s'aplica la rutina `find_root` per resoldre l'equació $P_s = 0.5$ amb $s \in (0,1)$. La rutina global que calcula la mediana es troba a l'Annex B.1.

La següent funció d'interès és $(t-s)^2$. Hem de minimitzar la funció

$$\begin{aligned} g(t) &= \int_0^1 (t-s)^2 f(s) ds = t^2 \int_0^1 f(s) ds - 2t \int_0^1 s f(s) ds + \int_0^1 f(s) s^2 ds \\ &= t^2 - 2t\,\mathrm{E}[\theta] + \mathrm{E}[\theta^2]. \end{aligned}$$

I derivant i igualant a zero,

$$t = \mathrm{E}[\theta].$$

Que la mediana i l'esperança minimitzen les integrals de $|t-s|$ i $|t-s|^2$ respectivament es un fet conegut,

per a qualsevol densitat $f(s)$ per la que existeixin moments de segon ordre. El que ens interessa és el càlcul efectiu de mediana i esperança amb la densitat que ens ocupa. Hem vist com l'hem fet per la mediana. Veiem ara el cas de l'esperança.

Hem usat el paquet d'integració numèrica `quadpack` implementat en `maxima` per obtenir el resultat. Les rutines d'aquest paquet donen una estimació de l'error que no és estricta. És possible acotar estrictament l'error procedint de la manera següent: suposant $x_2 \leq x_1$, hem de calcular la integral (vegeu la fórmula de la densitat en el capítol anterior)

$$\int_{t_1}^{t_2} \Big[s \frac{-(x_2-x_1)}{(t_2-t_1)^2} \sqrt{\frac{2(t_2-t_1)(s-t_1)}{\pi(t_2-s)}} \exp\Big\{ -\frac{(x_2-x_1)^2(t_2-s)}{2(t_2-t_1)(s-t_1)} \Big\} + s \frac{((t_2-t_1)-(x_2-x_1)^2)}{(t_2-t_1)^2} \operatorname{erfc}\left(-(x_2-x_1)\sqrt{\frac{(t_2-s)}{2(t_2-t_1)(s-t_1)}} \right) \Big] ds.$$

El segon sumand no dóna problemes numèrics. Es pot integrar numèricament, per trapezis o qualsevol mètode similar, perquè l'integrand és acotat. Pel primer sumand,

$$\lim_{s\to t_1} s \frac{-(x_2-x_1)}{(t_2-t_1)^2} \sqrt{\frac{2(t_2-t_1)(s-t_1)}{\pi(t_2-s)}} \exp\Big\{ -\frac{(x_2-x_1)^2(t_2-s)}{2(t_2-t_1)(s-t_1)} \Big\} = 0 \,,$$

però en canvi en t_2 hi ha una singularitat integrable.

Per tant el càlcul de l'esperança de l'argument del mínim $B_t^{(t_1,x_1)(t_2,x_2)}$ a $[t_1,t_2]$ passa per veure com fer mitjançant integració numèrica una integral de la forma

$$\int_{t_1}^{t_2} s \sqrt{\frac{s-t_1}{t_2-s}} \exp\Big\{ -K\frac{t_2-s}{s-t_1} \Big\} ds \,,$$

per a certa constant K.

Com que hi ha una singularitat en t_2, integrarem en $[t_1,\overline{t_2}]$, per a un $\overline{t_2}$ proper a t_2, i en la part de la singularitat $[\overline{t_2},t_2]$ aproximarem per Taylor la funció exponencial.

Fent $h(z) = e^z \simeq 1+z$, l'error en el punt $z<0$ és igual a $\frac{h''(c)z^2}{2}$, per a un cert $c \in [z,0]$. Per tant, l'error màxim per a z en un interval $[\bar{z},0]$ està acotat per

$$\sup_{c,z\in[\bar{z},0]} \frac{h''(c)z^2}{2} = \frac{\bar{z}^2}{2}.$$

Pont	$\mathrm{E}[m]$	$\mathrm{E}[\theta]$	$\mathrm{E}[X_{\mathrm{E}[\theta]}]-\mathrm{E}[m]$	$\mathbb{M}[\theta]$	$\mathrm{E}[X_{\mathbb{M}[\theta]}]-\mathrm{E}[m]$
$(0,0)(1,0.0)$	-0.62665	0.50000	0.62665	0.50000	0.62665
$(0,0)(1,0.1)$	-0.57963	0.44203	0.62383	0.42526	0.61215
$(0,0)(1,0.2)$	-0.53797	0.39240	0.61645	0.36059	0.61008
$(0,0)(1,0.3)$	-0.50091	0.34972	0.60582	0.30552	0.59256
$(0,0)(1,0.4)$	-0.46783	0.31286	0.59297	0.25917	0.57149
$(0,0)(1,0.5)$	-0.43818	0.28090	0.57863	0.22046	0.54841
$(0,0)(1,0.6)$	-0.41151	0.25309	0.56336	0.18828	0.52447
$(0,0)(1,0.7)$	-0.38744	0.22878	0.54758	0.16158	0.50054
$(0,0)(1,0.8)$	-0.36565	0.20747	0.53162	0.13942	0.47718
$(0,0)(1,0.8)$	-0.34586	0.18871	0.51569	0.12098	0.45474
$(0,0)(1,1.0)$	-0.32783	0.17216	0.49999	0.10559	0.43342

Taula 3.1: Per a diferents ponts, diferència entre l'esperança del mínim del pont i el valor esperat del procés de l'esperança de l'argument del mínim i de la mediana del l'argument del mínim

Si fem $\overline{z} = -K\frac{t_2-\overline{t_2}}{t_2-t_1}$, podem trobar $\overline{t_2}$ per tal que l'error no superi un ε prefixat:

$$\varepsilon = K^2\left(\frac{t_2-\overline{t_2}}{\overline{t_2}-t_1}\right)^2 \Rightarrow \overline{t_2} = \frac{t_2+t_1\sqrt{\frac{2\varepsilon}{K^2}}}{1+\sqrt{\frac{2\varepsilon}{K^2}}}\,.$$

Podem tabular l'esperança i la mediana de l'argument del mínim del pont brownià per a diferents ratios entre la diferencia $|x_1-x_0|$ dels valors inicial i final i la longitud de l'interval. És suficient considerar ponts entre $(0,0)$ i $(1,x_1)$, amb $x_1 \geq 0$.

Exemple 3.3.1. *Sigui $B_t^{(t_0,x0)(t_1,x_1)}$ un pont brownià de (t_0,x_0) a (t_1,x_1). Aquesta taula il·lustra les diferències, per a diferents ponts en l'interval $[0,1]$, entre l'esperança del mínim del pont, calculat analíticament amb la fórmula (3.3) i el valor esperat del procés en el punt on hi ha calculats numèricament: a) l'esperança de l'argument del mínim; b) la mediana del l'argument del mínim, truncats a cinc decimals*

Es pot considerar, més en general, les funcions de pèrdua associades a integrar $|t-s|^{2n}$ respecte la densitat $f(s)$ del mínim del pont. Acabem de veure el casos particulars $n=0$ i $n=1$. Per n general

$$\begin{aligned} g(t) &= \int_0^1 (t-s)^{2n} f(s)ds = t^{2n}\int_0^1 f(s)ds - 2nt^{2n-1}\int_0^1 sf(s)ds \\ &\quad + \ldots + (-1)^{2n-i}\binom{2n}{2n-i}t^{2n-i}\int_0^1 s^i f(s)ds + \ldots + \int_0^1 s^{2n}f(s)ds \\ &= t^{2n} - 2nt^{2n-1}I_1 + \ldots + I_{2n}\,, \end{aligned}$$

Pont	Polinomi	Arrels
$(0,0)(1,0.0)$	$4t^3-6.0t^2+4.0t-1.0$	0.50000
$(0,0)(1,0.1)$	$4t^3-5.30444t^2+3.32212t-0.79245$	0.45729
$(0,0)(1,0.2)$	$4t^3-4.70886t^2+2.77165t-0.63091$	0.42119
$(0,0)(1,0.3)$	$4t^3-4.19669t^2+2.32259t-0.50461$	0.38981
$(0,0)(1,0.4)$	$4t^3-3.75439t^2+1.95463t-0.40539$	0.36207
$(0,0)(1,0.5)$	$4t^3-3.37090t^2+1.65181t-0.32711$	0.33726
$(0,0)(1,0.6)$	$4t^3-3.03709t^2+1.40155t-0.26507$	0.31488
$(0,0)(1,0.7)$	$4t^3-2.74544t^2+1.19386t-0.21569$	0.29458
$(0,0)(1,0.8)$	$4t^3-2.48969t^2+1.02082t-0.17623$	0.27607
$(0,0)(1,0.9)$	$4t^3-2.26463t^2+0.87609t-0.14456$	0.25914
$(0,0)(1,1.0)$	$4t^3-2.06592t^2+0.75456t-0.11904$	0.24360

Taula 3.2: Taula que mostra per a cada pont el polinomi del qual se li ha de calcular l'arrel i la seva arrel real.

on hem utilitzat la notació $I_i=\int_0^1 s^i f(s)ds$.

Si derivem i igualem a zero, veiem que cal trobar les arrels del polinomi

$$p(t)_{2n}=2nt^{2n-1}-2n(2n-1)t^{2n-2}I_1+\ldots+I_{2n-1}.$$

Exemple 3.3.2. *Farem només el cas $n=2$ per trobar valors de t no relacionats ni amb la mediana ni amb l'esperança. El polinomi en qüestió és*

$$p(t)_4=4t^3-12t^2I_1+12tI_2-8I_3$$

Vegeu la Taula 3.2, on hem fet el càlcul per als mateixos ponts de la taula anterior. Observem que com més creix l'exponent de la funció de pèrdua $(t-s)^{2n}$ més van els punts òptims cap a la dreta.

Observació 3.3.1. *A la vista de la densitat de l'argument del mínim, observem que la moda no és útil per obtenir un punt interior on sigui raonable mostrejar. Efectivament, excepte en el cas en què els valors inicial i final del pont són iguals, en què la densitat de l'argument és uniforme, la densitat no és acotada, i tendeix a infinit en algun dels dos extrems de l'interval. Concretament, allà on hi ha el valor més gran.*

3.4 Mètodes de selecció d'interval

En aquest apartat i el següent descriurem i estudiarem empíricament unes quantes heurístiques adaptatives relativament senzilles per atacar el problema d'aproximar el mínim de la trajectòria d'un pont

brownià.

En el capítol anterior hem plantejat tres mètodes que generen a priori els n punts a mostrejar. Quan es mostreja d'aquesta manera es parla d'algorismes *no-adaptatius* o *passius*. Això té l'avantatge que es poden paral·lelitzar els càlculs i així reduir el temps total invertit. En canvi, quan s'escullen els punts de manera successiva, tenint en compte les imatges dels punts prèviament mostrejats, es parla d'algorismes *adaptatius*, pels quals és d'esperar un error més petit que per als passius, per a un mateix n.

El criteri de comparació que hem usat en el capítol anterior, i que aquí seguirem usant, és el d'avaluar la diferència

$$\mathrm{E}\left[\min_{0\le i\le n+1} B_{t_i} - m(B)\right] . \tag{3.1}$$

on B és el pont original, connectant $(0,x_0)$ i $(1,x_{n+1})$. Aquesta esperança s'aproxima per simulació: en primer lloc, l'expressió (3.1) es pot escriure com

$$\mathrm{E}\left[\mathrm{E}\left[\min_{0\le i\le n+1} B_{t_i} - m(B) \,/\, \{B_{t_i},\ i=1,\dots,n\}\right]\right] \tag{3.2}$$

(notem que ara els punts t_i on s'avaluarà cada trajectòria son també aleatoris). Posant $x_i(\omega)$ els valors observats del pont en els punts $t_i(\omega)$, (3.2) és igual a

$$\mathrm{E}\left[\min_{0\le i\le n+1} x_i - \mathrm{E}\left[m(B) \,/\, \{B_{t_i}=x_i,\ i=1,\dots,n\}\right]\right] .$$

L'esperança (i l'interval de confiança asimptòtic associat) s'estimen mitjançant la mitjana i la desviació tipus mostrals. L'esperança condicionada s'estima simulant independentment molts valors de $m(B)$ per al pont brownià condicionat a passar per tots els punts (t_i,x_i), $i=1,\dots,n$; el procediment és el descrit a l'Apartat 4 del capítol anterior. Remarquem que no cal mai simular tota la trajectòria del pont brownià, sinó només els punts (t_i,x_i) a mesura que es necessiten.

Recordem que els mètodes no-adaptatius del capítol anterior, que anomenàvem *random*, *equidistant* i *equiprobable* consistien respectivament en mostrejar de manera aleatòria uniforme, de manera regular, i de manera que l'interval quedés dividit en subintervals amb la mateixa probabilitat de contenir el mínim. Els resultats rellevants són a les taules 2 i 3 del capítol anterior.

En aquest apartat l'estratègia que seguirem consisteix en donat un pont inicial $B_t^{(t_0,x_0)(t_1,x_1)}$, mostrejar en un punt $t^* = t_2 \in (t_0,t_1)$, donat per algun dels mètodes de l'apartat 3.3, $t^* = \mathrm{E}[\theta]$ o $t^* = \mathbb{M}[\theta]$. Al avaluar en t_2, tenim dos ponts concatenats $B_t^{(t_0,x_0)(t_2,x_2)}$ i $B_t^{(t_2,x_2)(t_1,x_1)}$; utilitzem aleshores algun dels mètodes de selecció d'interval que tot seguit descrivim per determinar en quin dels dos intervals, (t_0,t_2) o bé (t_2,t_1),

mostregem el proper t_3. I així successivament.

Suposem per tant, donat un procés X com una concatenació de ponts $B_t^{(t_0,x_0)(t_1,x_1)}$, $B_t^{(t_1,x_1)(t_2,x_2)}$, ..., $B_t^{(t_n,x_n)(t_{n+1},x_{n+1})}$, que denotarem B^0, B^1, ..., B^n i plantegem tres mètodes per escollir en quin interval mostrejarem després usant les idees de l'apartat anterior. Aquests tres eleccions seran:

1. L'interval amb la màxima probabilitat de contenir el mínim del procés X.

2. L'interval en què l'esperança del valor mínim és més petita.

3. L'interval en què la mediana del valor mínim és més petita.

1. Explorar a l'interval que tingui màxima P_i:

Donats els intervals (t_i,t_{i+1}), calcularem la probabilitat P_i que $\theta \in (t_i,t_{i+1})$ de la manera explicada a l'Apartat 3.2 del segon capítol, i explorarem mitjançant $t = \mathrm{E}[\theta(B^i)]$ o $t = \mathbb{M}[\theta(B^i)]$ del pont de l'interval que tingui la màxima P_i. Aquestes probabilitats P_i s'han de recalcular cada vegada que s'afegeix un nou punt intermedi t.

2. Explorar a l'interval que tingui mínima esperança de la imatge del mínim.

Calculem $\mathrm{E}[m(B^i)]$ per a cada interval (t_i,t_{i+1}) de la següent manera. Per exemple, sigui $m = \min B^0 = \min_{s\in[t_0,t_1]} B_s^{(t_0,x_0)(t_1,x_1)}$. Aleshores

$$\mathrm{E}[m] = \min(x_0,x_1) - \frac{1}{4}\sqrt{2\pi(t_1-t_0)}\exp\left\{\frac{(x_0-x_1)^2}{2(t_1-t_0)}\right\}\left(1+\mathrm{erf}\left(\frac{(2\min(x_1,x_0)-x_0-x_1)\sqrt{2}}{2\sqrt{t_1-t_0}}\right)\right). \tag{3.3}$$

En efecte,

$$\begin{aligned}&\int x\frac{-2(2x-x_0-x_1)\exp\left\{-\frac{(2x_1-2x)(x_0-x)}{t_1-t_0}\right\}}{t_1-t_0}dx\\ &= x\exp\left\{\frac{-2(x-x_1)(x-x_0)}{t_1-t_0}\right\} - \frac{1}{4}\sqrt{2\pi(t_1-t_0)}\exp\left\{\frac{(x_0-x_1)^2}{2(t_1-t_0)}\right\}\left(1+\mathrm{erf}\left(\frac{(2x-x_0-x_1)\sqrt{2}}{2\sqrt{t_1-t_0}}\right)\right).\end{aligned}$$

I per tant, l'esperança buscada és

$$\begin{aligned}
\mathrm{E}[m] &= \int_{-\infty}^{\min(x_0,x_1)} x \frac{-2(2x-x_0-x_1)\exp\left\{-\frac{(2x_1-2x)(x_0-x)}{t_1-t_0}\right\}}{t_1-t_0} dx \\
&= \min(x_0,x_1)\exp\left\{\frac{-2(\min(x_0,x_1)-x_1)(\min(x_0,x_1)-x_0)}{t_1-t_0}\right\} - \frac{1}{4}\sqrt{2\pi(t_1-t_0)} \\
&\quad \exp\left\{\frac{(x_0-x_1)^2}{2(t_1-t_0)}\right\} \mathrm{erf}\left(\frac{(2\min(x_0,x_1)-x_0-x_1)\sqrt{2}}{2\sqrt{t_1-t_0}}\right) \\
&= \min(x_0,x_1) - \frac{1}{4}\sqrt{2\pi(t_1-t_0)}\exp\left\{\frac{(x_0-x_1)^2}{2(t_1-t_0)}\right\}\left(1+\mathrm{erf}\left(\frac{(2\min(x_0,x_1)-x_0-x_1)\sqrt{2}}{2\sqrt{t_1-t_0}}\right)\right) .
\end{aligned}$$

I explorarem mitjançant $\mathrm{E}[\theta]$ o $\mathbb{M}[\theta]$ a l'interval que tingui min $\mathrm{E}[m_i]$. Quan afegim un nou punt, s'aprofiten els càlculs de totes les $\mathrm{E}[m_i]$ menys la de l'interval on s'ha explorat.

3.Explorar a l'interval que tingui mínima mediana.

Calculem $\mathbb{M}[m(B^i)]$ per a cada interval (t_i, t_{i+1}) de la següent manera. Per exemple, per a la mediana de m en el primer interval

$$\mathbb{M}[m] = \frac{(x_0+x_1) - \sqrt{(x_0+x_1)^2 - 4(x_0x_1 + \frac{\ln(0.5)(t_1-t_0)}{2})}}{2} ,$$

on hem utilitzat que la funció de distribució de m és

$$F(x) = \mathbb{1}_{\{x\geq\min(x_0,x_1)\}} + \exp\left\{-\frac{(2x_1-2x)(x_0-x)}{t_1-t_0}\right\}\mathbb{1}_{\{x<\min(x_0,x_1)\}},$$

i trobat la solució més petita de l'equació

$$-\frac{(2x_1-2x)(x_0-x)}{t_1-t_0} = \ln(0.5) .$$

I explorarem mitjançant $\mathrm{E}[\theta]$ o $\mathbb{M}[\theta]$ a l'interval que tingui min $\mathbb{M}[m_i]$. Quan afegim un nou punt, s'aprofiten els càlculs de totes les $\mathbb{M}[m_i]$ menys la de l'interval on s'ha explorat.

La Taula 3.3 resumeix els resultats experimentals dels sis mètodes sobre les mateixes trajectòries, ja que al simular cada nou valor x_i de la trajectòria es té en compte els valors x_{i-1} i x_{i+1} que ha simulat

	Pont de (0,0) a (1,1)		
nre. de punts	95% I.C. esp/esp	95% I.C. med/esp	95% I.C. P_i/esp
2	[0.19189,0.22528]	[0.19086,0.22581]	[0.19086,0.22581]
4	[0.12381,0.14807]	[0.12631,0.16466]	[0.12609,0.15832]
8	[0.07699,0.10465]	[0.09060,0.13027]	[0.08089,0.10837]
16	[0.04700,0.07662]	[0.05011,0.07996]	[0.03633,0.05739]
32	[0.03191,0.06835]	[0.03529,0.07191]	[0.01521,0.02161]
64	[0.03066,0.05745]	[0.03907,0.06910]	[0.01541,0.02289]
nre. de punts	95% I.C. esp/med	95% I.C. med/med	95% I.C. P_i/med
2	[0.19011,0.22709]	[0.18963,0.22285]	[0.18674,0.22012]
4	[0.13113,0.16937]	[0.12822,0.16661]	[0.13338,0.17054]
8	[0.07506,0.10559]	[0.08187,0.12006]	[0.07784,0.10387]
16	[0.05096,0.08832]	[0.05354,0.09260]	[0.03699,0.05230]
32	[0.04853,0.09335]	[0.05900,0.10620]	[0.01986,0.03739]
64	[0.04144,0.07626]	[0.04320,0.07802]	[0.01577,0.02806]

Taula 3.3: De dalt a baix i d'esquerra a dreta, intervals de confiança aproximats per a l'esperança de l'error quan estimem el mínim d'un pont brownià pel mínim dels valors simulats per als mètodes: interval en què l'esperança del mínim és més petita i esperança de l'argument del mínim ('esp/esp'), interval en què la mediana del mínim és més petita i esperança de l'argument del mínim ('med/esp'), interval amb la màxima probabilitat de contenir el mínim del procés X i esperança de l'argument del mínim ('P_i/esp'), interval en què l'esperança del mínim és més petita i mediana de l'argument del mínim ('esp/med'), interval en què la mediana del mínim és més petita i mediana de l'argument del mínim ('med/med'), interval amb la màxima probabilitat de contenir el mínim del procés X i mediana de l'argument del mínim ('P_i/med').

el mètode anterior (el primer que simula punts de la trajectòria és el mètode 'esp/esp'). A més quan s'estima l'esperança del mínim de tota la trajectòria es fan servir tots els punts que ha fet servir cada mètode. Per tant si hi ha i mètodes que fan servir n_i punts cadascun, la simulació del mínim de la trajectòria utilitzarà $\sum_i n_i$ punts. Aquestes rutines es troben a l'Annex B.2. S'observa que per a pocs punts (de 2 a 8) és millor agafar la mediana que l'esperança de l'argument, en canvi per a molts punts (16, 32, 64) es canvien els papers. També que sempre és millor el mètode de selecció d'interval de màxima P_i. Finalment, es veu que no milloren gaire els resultats al passar de usar 32 punts a agafar-ne el doble, 64.

Les raons de perquè és millor el mètode de selecció de la màxima P_i, les trobem en els histogrames del nombre de punts que avaluen realment els processos, que podem veure a la Figura 3.1, fets en el cas $n = 20$ punts i amb 100 simulacions. El mètode de la màxima P_i és el que més punts mostreja ja que arriba als 20 punts la majoria de vegades, cosa que no fan la resta de mètodes que seleccionen l'interval amb altres criteris. Intuïtivament podríem dir que el mètode de màxima P_i explora més punts perquè d'alguna manera quan se selecciona l'interval es reparteixen la probabilitat, i l'interval que havia quedat segon, en el procés de selecció, pot passar a ser primer. En canvi en els criteris que s'escull segons

l'esperança o mediana més baixa del mínim, un dels dos nous intervals que ha aparegut és probable que segueixi guanyant. Degut a que els sis mètodes acaben mostrejant en punts molt propers entre sí, els programes contenen una condició de parada de la cerca quan la distància entre els t_i's successius és inferior a 10^{-5}.

3.5 Mètodes d'escollir el millor candidat

En aquest apartat, considerem tres mètodes en els quals se selecciona un punt de cadascun dels intervals en què està dividida la trajectòria pels punts ja mostrejats, i després s'escull un d'ells per avaluar la funció. els anomenem *mètodes de candidats.*

Donats una sèrie de candidats c_i a següent t a explorar, ens plantegem quin agafar d'entre els c_i. El criteri que farem servir és agafar el que tingui l'esperança de la imatge més baixa:

$$t^* := \min_{c_i} \mathrm{E}[X_{c_i}] \ .$$

Donat un pont $B_t^{(t_i,x_i),(t_{i+1},x_{i+1})}$ i $c_i \in [t_i, t_{i+1}]$, sabem que

$$\mathrm{E}[X_{c_i}] = \frac{x_{i+1} - x_i}{t_{i+1} - t_i}(c_i - t_i) + x_i \ .$$

Per tant, l'elecció del millor candidat és molt simple.

Al capítol anterior, la successió de punts t venia determinada pel criteri de selecció de l'interval, de manera que en el pas k-èsim teníem k intervals i es tractava d'escollir en quin interval mostrejàvem per obtenir el k-èsim punt. Ara, en canvi, avaluem en els k intervals, i per tant tenim k punts candidats $c_1, \ldots, c_k$; calculant les esperances $\mathrm{E}[X_{c_k}]$ obtindrem el següent punt t on mostrejarem.

Els criteris que estudiarem, mitjançant 100 simulacions de trajectòries per als mètodes i 100 simulacions del mínim de cada trajectòria, per donar candidats c_i a cada pas són:

1. Criteri de l'esperança de l'argument del mínim a cada interval.
2. Criteri de la mediana de l'argument del mínim a cada interval.
3. Criteri del punt mig de cada interval.

Es poden aprofitar els candidats del pas anterior, ja que a cada nou pas només s'afegeixen els dos candidats corresponents als dos nous subintervals que produeix el punt t acabat d'escollir.

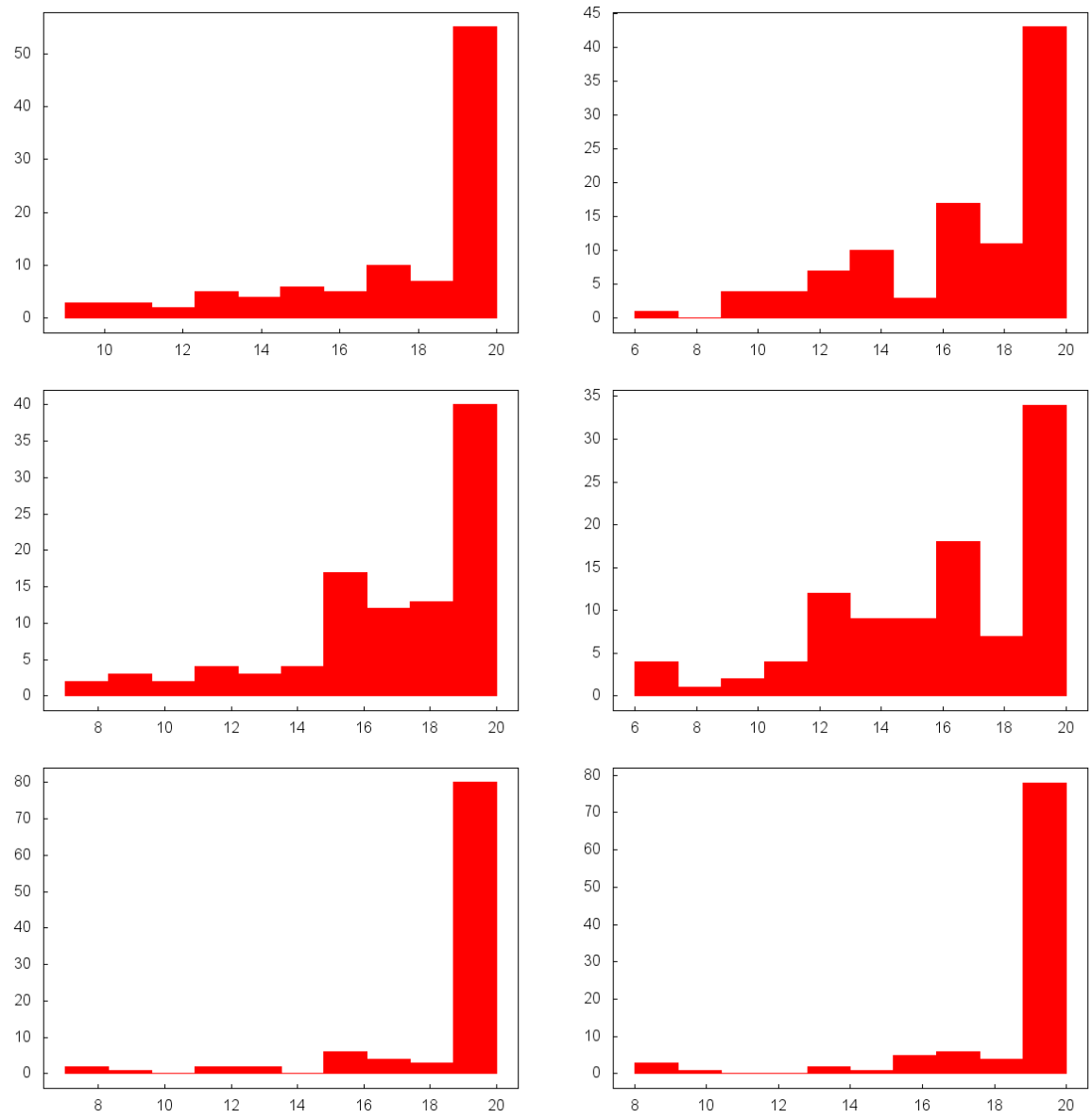

Figura 3.1: De dalt a baix i d'esquerra a dreta, en el cas $n = 20$ i amb 100 simulacions, punts realment avaluats pels mètodes: interval de mínima esperança i esperança de l'argument, interval de mínima esperança i mediana de l'argument, interval de mínima mediana i esperança de l'argument, interval de mínima mediana i mediana de l'argument, interval de màxima P_i i esperança de l'argument, i interval de màxima P_i i mediana de l'argument.

	Pont de (0,0) a (1,1)		
nre. de punts	95% I.C. eqd	95% I.C. rnd	95% I.C. eqp
2	[0.23952,0.29783]	[0.23858,0.29890]	[0.19145,0.24396]
4	[0.20951,0.26066]	[0.20575,0.26600]	[0.15315,0.19947]
8	[0.13745,0.17531]	[0.14389,0.18827]	[0.11355,0.14690]
16	[0.10741,0.13345]	[0.12546,0.16667]	[0.08799,0.11843]
32	[0.08367,0.10241]	[0.08636,0.11297]	[0.06356,0.08983]
64	[0.06365,0.07632]	[0.07186,0.09270]	[0.04420,0.05960]
nre. de punts	95% I.C. cand-esp-arg	95% I.C. cand-med-arg	95% I.C. cand-punt-mig
2	[0.18526,0.25395]	[0.18700,0.25421]	[0.21769,0.27024]
4	[0.16882,0.23551]	[0.16914,0.23653]	[0.14795,0.18290]
8	[0.14361,0.21277]	[0.14560,0.21689]	[0.08561,0.11302]
16	[0.13927,0.21331]	[0.14456,0.22030]	[0.05435,0.08717]
32	[0.13684,0.20506]	[0.14476,0.21300]	[0.07067,0.10968]
64	[0.14397,0.21436]	[0.14666,0.21795]	[0.05764,0.09074]

Taula 3.4: De dalt a baix i d'esquerra a dreta, intervals de confiança aproximats per a l'esperança de l'error quan estimem el mínim del pont brownià pel mínim dels valors samplejats 100 vegades, per als mètodes (els tres primers definits al Capítol 2): equidistant ('eqd'), random ('rnd'), equiprobable ('eqp'), candidats provinents de l'esperança de l'argument del mínim ('cand-esp-arg'), candidats provinents de la mediana de l'argument del mínim ('cand-med-arg') i candidats provinents del punt mig de l'interval ('cand-punt-mig') .

A la Taula 3.4 es poden comparar els resultats obtinguts dels mètodes no adaptatius del capítol anterior amb els mètodes de candidats amb tots sis mètodes sobre les mateixes trajectòries (el primer que simula punts de la trajectòria és el mètode equidistant). Aquestes rutines es troben a l'Annex B.3.

Observem que a partir de 16 punts no sembla ser millor mostrejar més punts. Però això és degut a què amb 100 simulacions hi ha encara molt soroll i no es possible veure'n l'evolució ja a partir de 16 punts. En efecte, en les dues primeres columnes, la desviació tipus és de gairebé 0.2, i produeix un interval de longitud aproximadament 0.08. L'interval de 16 punts ja intersecta molt amb el de 8. En canvi, a la tercera, el mètode de bisecció, apart de la velocitat de càlcul que aporta, té una desviació tipus de l'ordre de 0.1, produint intervals la meitat de llargs. Comença a no detectar-se millora a partir de 32 punts.

En conclusió, ens caldrà fer simulacions en un nombre de aproximadament 10 000 per poder apreciar la millora amb tants punts. Remarquem també que a diferència dels mètodes de selecció d'interval, tots tres mètodes de candidats han avaluat els 64 punts; no es produeix el bloqueig dels algorismes per mostrejos massa propers.

Ha quedat clar que el mètode de bisecció és, dels tres criteris de candidats, el més ràpid en reduir l'error i per tant el que obté millors resultats.

Arribats aquí, és lògic preguntar-se si subdividint els intervals en una proporció diferent de $p = 0.5$, que és el que fem amb la bisecció, seria possible trobar una proporció p que encara donés millors resultats. La intuïció ens diria que seria millor agafar una proporció propera al t_i que tingui la imatge més baixa,

p	I. C. 95%
0.05	[0.24125,0.26774]
0.10	[0.19477,0.22490]
0.15	[0.17937,0.21112]
0.20	[0.15920,0.19322]
0.25	[0.10702,0.13224]
0.30	[0.11956,0.14604]
0.35	[0.10324,0.12788]
0.40	[0.09887,0.12491]
0.45	[0.09748,0.12056]
0.50	[0.08101,0.10037]
0.55	[0.07107,0.08901]
0.60	[0.07543,0.09403]
0.65	[0.08437,0.10296]
0.70	[0.09532,0.11131]
0.75	[0.13869,0.15727]
0.80	[0.15625,0.17726]
0.85	[0.20485,0.22307]
0.90	[0.24042,0.26262]
0.95	[0.28822,0.31312]

Taula 3.5: Intervals de confiança aproximats per a l'esperança de l'error quan estimem el mínim del pont brownià pel mínim dels valors samplejats 100 vegades, en el cas $n = 10$ punts.

però experimentant observem que no és així. Les Taules 3.5 i 3.6 mostren l'evolució dels intervals de confiança quan subdividim cada interval (t_i, t_{i+1}) pel punt t^* que està a distància $p \cdot (t_{i+1} - t_i)$ de l'extrem de l'interval que té la imatge més petita, variant aquesta p.

Les taules estan fetes per el cas de $n = 10$ punts, i la proporció òptima p sembla ser al voltant de 0.55 o 0.56. El marge d'error, que és de l'ordre de 0.0005, no permet resoldre de manera més fina.

3.6 Funcions de pèrdua amb X_t i $m(X)$

Per treballar amb funcions de pèrdua del tipus $L(X_t, m(X))$, primer ens cal la densitat conjunta del pont brownià i el seu mínim, que no es troba a la literatura. Volem agrair al professor Jim Pitman la seva valuosa ajuda en la idea emprada aquí per calcular aquesta densitat conjunta. El resultat final és a la fórmula (3.7).

Suposarem, per simplificar alguns paràmetres, que el pont brownià parteix de $(0,0)$ i arriba a un punt $(0,b)$. A partir de la densitat trobada és fàcil passar a la situació general.

Si B és aquest pont, i denotem com sempre $m(B) := \min_{s\in[0,1]} B(s)$, i $\theta(B) := \arg\min_{s\in[0,1]} B_s$, la idea és passar per la densitat conjunta triple de $(B_t, m(B), \theta(B))$ i calcular aquesta mitjançant la marginal de

p	I. C. 95%
0.50	[0.08788,0.09001]
0.51	[0.08558,0.08764]
0.52	[0.08623,0.08827]
0.53	[0.08517,0.08719]
0.54	[0.08445,0.08641]
0.55	[0.08325,0.08514]
0.56	[0.08320,0.08509]
0.57	[0.08419,0.08604]
0.58	[0.08395,0.08577]
0.59	[0.08407,0.08586]
0.60	[0.08547,0.08724]
0.61	[0.08635,0.08807]
0.62	[0.08803,0.08974]

Taula 3.6: Intervals de confiança aproximats per a l'esperança de l'error quan estimem el mínim del pont brownià pel mínim dels valors samplejats 10000 vegades, en el cas $n = 10$ punts.

$(m(B), \theta(B))$, que ja tenim (vegeu la Proposició 6.2 del Capítol 2), i la corresponent condicionada; és a dir:

$$\begin{aligned} f_{(B_t, m(B))}(x,y) &= \int_0^1 f_{(B_t, m(B), \theta(B))}(x,y,s)ds \\ &= \int_0^1 f_{B_t|_{(m(B),\theta(B))=(y,s)}}(x) f_{(m(B),\theta(B))}(y,s)ds. \end{aligned} \tag{3.4}$$

Per calcular el primer factor de l'integrand, usarem un resultat de Asmussen, Glynn i Pitman [4] que descompon el moviment brownià en dos ponts de Bessel empalmats esquena contra esquena en el punt on el pont assoleix el màxim. Vegeu la Figura 3.2.

Lema 3.6.1 (Proposició 2 en [4])**.**

Sigui W un moviment brownià estàndard, començant en $(0,0)$.

Sigui $\tau(W)$ l'instant (únic quasi segur) en $[0,1]$ on W assoleix el seu màxim $M(W) := \max_{t\in[0,1]} W_t$.

Aleshores:

1. *La llei del procés $\{M(W) - W_{s-u},\ 0 \le u \le s\}$, condicionat a $\tau(W) = s$, $M(W) = m$, $W_1 = m - y$, és un pont de Bessel d'ordre 3 de $(0,0)$ a (s,m).*

2. *La llei del procés $\{M(W) - W_{s+u},\ 0 \le u \le 1-s\}$, condicionat a $\tau(W) = s$, $M(W) = m$, $W_1 = m - y$, és un pont de Bessel d'ordre 3 de $(0,0)$ a $(1-s,y)$.*

3. *Els dos ponts de Bessel són independents.*

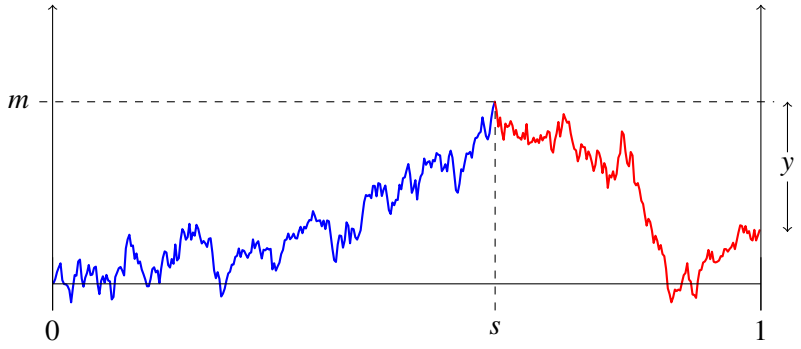

Figura 3.2: Descomposició del moviment brownià en el seu punt màxim (s,m) en dos ponts de Bessel independents.

D'altra banda, un procés de Bessel començant en 0, $Y \sim BES(3)$, és conegut que té densitat a l'instant t

$$f_{Y_t}(y) = \sqrt{\frac{2}{\pi}} t^{-\frac{3}{2}} y^2 \exp\Big\{ -\frac{y^2}{2t} \Big\} ,$$

cosa que es dedueix també fàcilment del fet que $Y_t = \sqrt{tZ}$, on $Z \sim \chi^2(3)$. Es tracta d'un procés de Markov amb densitat de transició

$$p_t(y,z) = \frac{z^{3/2}}{t y^{1/2}} \exp\Big\{ -\frac{y^2+z^2}{2t} \Big\} I_{1/2}\Big(\frac{yz}{t}\Big) \tag{3.5}$$

(vegeu [21, Prop 6.3.1], [15, secció 2.7], o també [8]), on $I_\nu(x)$ és la "modified Bessel function" d'ordre ν, que es defineix com

$$I_\nu(x) = \sum_{n=0}^{\infty} \frac{(x/2)^{\nu+2n}}{n!\Gamma(\nu+n+1)} .$$

Pel que fa al cas $\nu = 1/2$, és té que

$$\lim_{x\to 0} I_{1/2}(x) \sim \frac{1}{\Gamma(3/2)} (x/2)^{1/2} = \sqrt{\frac{2}{\pi}} x^{1/2} ,$$

i d'aquí es pot comprovar que $\lim_{y\to 0} p_t(y,z) = p_t(0,z) = f_{Y_t}(z)$. De fet, es té exactament l'expressió explícita

$$I_{1/2}(x) = \sqrt{\frac{2}{\pi x}} \sinh(x)$$

que utilitzarem més avall, i d'aquí que, quan $x \to \infty$, $I_{1/2}(x) \sim \frac{e^x}{\sqrt{2\pi x}}$. Les funcions I_ν estan definides per a tot $x > 0$, o més en general, per tot x complex amb $|\arg(x)| < \frac{\pi}{2}$.

La llei d'un pont de Bessel d'ordre 3 de $(0,0)$ a un punt genèric (r,c), que denotarem $BESB(3,r,c)$, a l'instant u, amb $0 < u < r$, es pot obtenir calculant la conjunta (Y_u, Y_r), i després condicionant a $Y_r = c$.

Usant el nucli (3.5), tenim que

$$\begin{aligned} f_{(Y_u,Y_r)}(y,c) =& f_{Y_{r|_{Y_u=y}}}(c)\cdot f_{Y_u}(y) \\ =& \frac{c^{3/2}}{(r-u)y^{1/2}}\exp\Big\{-\frac{y^2+c^2}{2(r-u)}\Big\}I_{1/2}\Big(\frac{yc}{r-u}\Big)\cdot\mathbf{1}_{\{c>0\}} \\ & \cdot\sqrt{\frac{2}{\pi}}u^{-\frac{3}{2}}y^2\exp\Big\{-\frac{y^2}{2u}\Big\}\cdot\mathbf{1}_{\{y\geq 0\}}\ . \end{aligned}$$

Ara condicionem. Per $c>0$,

$$\begin{aligned} f_{Y_u|_{Y_r=c}}(y) =& \frac{f_{(Y_u,Y_r)}(y,c)}{f_{Y_r}(c)} \\ =& \Big(\frac{r}{u}\Big)^{3/2}\frac{y^{3/2}}{(r-u)c^{1/2}}\exp\Big\{-\frac{y^2}{2u}+\frac{c^2}{2r}-\frac{y^2+c^2}{2(r-u)}\Big\}\cdot I_{1/2}\Big(\frac{yc}{r-u}\Big)\cdot\mathbb{1}_{\{y\geq 0\}}\ . \end{aligned} \tag{3.6}$$

Passem ara al càlcul de $f_{B_t|_{(m(B),\theta(B))=(y,s)}}(x)$. Hem de distingir els dos casos $t<s$, $t>s$. Comencem pel primer.

Del Lema 3.6.1, deduïm que si C és un pont brownià de $(0,0)$ a $(1,b)$, llavors $\{M(C)-C_{s-u},\ 0\leq u\leq s\}$ condicionat a $M(C)=y$, $\tau(C)=s$, és un pont de Bessel de $(0,0)$ a (s,y), $BESB(3,s,y)$.

El que ens interessa és la llei del pont brownià B de $(0,0)$ a $(1,b)$, condicionat al mínim $m(B)=y$ i a la localització del mínim $\theta(B)=s$, per $y\leq\min(0,b)$, $s\in[0,1]$, $b\in\mathbb{R}$.

Definim $C:=-B$, que serà un pont de $(0,0)$ a $(1,-b)$. Tindrem que

$$\begin{aligned} m(B) &= -M(-B) = -M(C) \\ \theta(B) &= \tau(-B) = \tau(C)\ . \end{aligned}$$

Considerem el procés $R_u := m(B)-B_{s-u} = -M(C)+C_{s-u} = -(M(C)-C_{s-u})$. Si el condicionem a $M(C)=-m(B)=-y$, $\tau(C)=\theta(B)=s$, tenim, segons el Lema 3.6.1, un pont de Bessel de $(0,0)$ a $(s,-y)$, canviat de signe.

Per tant, la seva densitat surt de (3.6) fent els canvis

$$r:s\ ,\quad c:-y\ ,\quad z:-z\ ,$$

obtenint-se

$$g(z)=\Big(\frac{s}{u}\Big)^{3/2}\frac{(-z)^{3/2}}{(s-u)(-y)^{1/2}}\exp\Big\{-\frac{z^2}{2u}+\frac{y^2}{2s}-\frac{z^2+y^2}{2(s-u)}\Big\}I_{1/2}\Big(\frac{zy}{s-u}\Big)\mathbb{1}_{\{z\leq 0\}}\ .$$

Com que volem la llei de $\{B_t, 0 \le t \le s\}$ sota el mateix condicionament, si posem $u = s-t$, tindrem $B_t = B_{s-u} = m(B) - R_{s-t}$. Hem de fer els canvis

$$u : s-t \ , \quad z : -z$$

a la fórmula anterior,

$$h(z) = \Big(\frac{s}{s-t}\Big)^{3/2} \frac{z^{3/2}}{t(-y)^{1/2}} \exp\Big\{ -\frac{z^2}{2(s-t)} + \frac{y^2}{2s} - \frac{z^2+y^2}{2t} \Big\} I_{1/2}\Big(\frac{-zy}{t}\Big) \mathbb{1}_{\{z\ge 0\}}$$

i després $z : z-y$, per obtenir la densitat de B_t:

$$j(z) = \Big(\frac{s}{s-t}\Big)^{3/2} \frac{(z-y)^{3/2}}{t(-y)^{1/2}} \exp\Big\{ -\frac{(z-y)^2}{2(s-t)} + \frac{y^2}{2s} - \frac{(z-y)^2+y^2}{2t} \Big\} I_{1/2}\Big(\frac{-(z-y)y}{t}\Big) \mathbb{1}_{\{z\ge y\}} \ .$$

Pel que fa a l'altre cas, $t > s$, procedim de manera molt similar, usant la segona conclusió del Lema 3.6.1. Si C és un pont brownià de $(0,0)$ a $(1,b)$, llavors $\{M(C) - C_{s+u},\ 0 \le u \le 1-s\}$ condicionat a $M(C) = y$ i $\tau(C) = s$ segueix la llei $BESB(3, 1-s, y-b)$.

Com abans, sigui B un pont brownià de $(0,0)$ a $(1,b)$, i definim $C = -B$, que serà un pont de $(0,0)$ a $(1,-b)$. Tindrem, igual que abans, que

$$m(B) = -M(-B) = -M(C)$$
$$\theta(B) = \tau(-B) = \tau(C)$$

i ara considerem $R_u := m(B) - B_{s+u} = -M(C) + C_{s+u} = -(M(C) - C_{s+u})$. Aquest procés R condicionat a $M(C) = -m(B) = -y$, $\tau(C) = \sigma(B) = s$, serà un pont de Bessel de $(0,0)$ al $(1-s, -y+b)$, canviat de signe. La seva densitat a l'instant u és, fent els canvis

$$r : 1-s \ , \quad c : -y+b \ , \quad z : -z$$

a la fórmula (3.6),

$$g(z) = \Big(\frac{1-s}{u}\Big)^{3/2} \frac{(-z)^{3/2}}{(1-s-u)(-y+b)^{1/2}} \exp\Big\{ -\frac{z^2}{2u} + \frac{(-y+b)^2}{2(1-s)} - \frac{z^2+(-y+b)^2}{2(1-s-u)} \Big\}$$
$$\cdot I_{1/2}\Big(\frac{-z(-y+b)}{1-s-u}\Big) \mathbb{1}_{\{z\le 0\}}.$$

Sigui ara $u = t-s$, de manera que $B_t = B_{s+u} = m(B) - R_{t-s}$. La llei de $-R_{t-s}$, fent els canvis $z : -z$ i $u : t-s$ tindrà densitat

$$h(z) = \Big(\frac{1-s}{t-s}\Big)^{3/2} \frac{z^{3/2}}{(1-t)(-y+b)^{1/2}} \exp\Big\{ -\frac{z^2}{2(t-s)} + \frac{(-y+b)^2}{2(1-s)} - \frac{z^2+(-y+b)^2}{2(1-t)} \Big\}$$
$$\cdot I_{1/2}\Big(\frac{z(-y+b)}{1-t}\Big) \mathbb{1}_{\{z\geq 0\}} \, .$$

I la densitat de $m(B) - R_{t-s}$, amb el canvi $z : z-y$, queda

$$j(z) = \Big(\frac{1-s}{t-s}\Big)^{3/2} \frac{(z-y)^{3/2}}{(1-t)(-y+b)^{1/2}} \exp\Big\{ -\frac{(z-y)^2}{2(t-s)} + \frac{(-y+b)^2}{2(1-s)} - \frac{(z-y)^2+(-y+b)^2}{2(1-t)} \Big\}$$
$$\cdot I_{1/2}\Big(\frac{(z-y)(-y+b)}{1-t}\Big) \mathbb{1}_{\{z\geq y\}} \, .$$

Tornant a (3.4) queda:

$$f_{(B_t,m(B))}(x,y) = \int_0^1 f_{(B_t,m(B),\theta(B))}(x,y,s)ds = \int_0^1 f_{B_t|_{(m(B),\theta(B))=(y,s)}}(x) f_{(m(B),\theta(B))}(y,s)ds$$
$$= \int_0^t \Big(\frac{1-s}{t-s}\Big)^{3/2} \frac{(x-y)^{3/2}}{(1-t)(-y+b)^{1/2}} \exp\Big\{ -\frac{(x-y)^2}{2(t-s)} + \frac{(-y+b)^2}{2(1-s)} - \frac{(x-y)^2+(-y+b)^2}{2(1-t)} \Big\}$$
$$\cdot I_{1/2}\Big(\frac{(x-y)(-y+b)}{1-t}\Big) \frac{\exp\{\frac{b^2}{2}\}\sqrt{2}}{\sqrt{\pi s^3(1-s)^3}} y(y-b) \exp\Big\{ \frac{-y^2}{2s} - \frac{(y-b)^2}{2-2s} \Big\}$$
$$\cdot \mathbb{1}_{\{x\geq y\}} \mathbb{1}_{\{y<\min(0,b)\}} ds$$
$$+ \int_t^1 \Big(\frac{s}{s-t}\Big)^{3/2} \frac{(x-y)^{3/2}}{t(-y)^{1/2}} \exp\Big\{ -\frac{(x-y)^2}{2(s-t)} + \frac{y^2}{2s} - \frac{(x-y)^2+y^2}{2t} \Big\}$$
$$\cdot I_{1/2}\Big(\frac{-(x-y)y}{t}\Big) \frac{\exp\{\frac{b^2}{2}\}\sqrt{2}}{\sqrt{\pi s^3(1-s)^3}} y(y-b) \exp\Big\{ \frac{-y^2}{2s} - \frac{(y-b)^2}{2-2s} \Big\}$$
$$\cdot \mathbb{1}_{\{x\geq y\}} \mathbb{1}_{\{y<\min(0,b)\}} ds \, .$$

I ara utilitzant que

$$I_{1/2}\Big(\frac{(x-y)(-y+b)}{1-t}\Big) = \frac{1}{2}\sqrt{\frac{2(1-t)}{\pi(x-y)(-y+b)}} \Big(\exp\Big\{\frac{(x-y)(-y+b)}{1-t}\Big\} - \exp\Big\{\frac{-(x-y)(-y+b)}{1-t}\Big\}\Big)$$

i

$$I_{1/2}\Big(\frac{-(x-y)y}{t}\Big)=\frac{1}{2}\sqrt{\frac{2t}{-\pi(x-y)y}}\Big(\exp\Big\{\frac{-(x-y)y}{t}\Big\}-\exp\Big\{\frac{(x-y)y}{t}\Big\}\Big),$$

i simplificant, obtenim

$$\begin{aligned}
f_{B_t,m(B)}(x,y)=&\int_0^1 f_{(B_t,m(B),\theta(B))}(x,y,s)ds\\
=&\mathbb{1}_{\{x\geq y\}}\mathbb{1}_{\{\min(0,b)>y\}}\frac{\exp\{\frac{b^2}{2}\}(x-y)}{\pi}\\
&\times\Bigg[\frac{-y}{\sqrt{1-t}}\Big(\exp\Big\{-\frac{(x-b)^2}{2(1-t)}\Big\}-\exp\Big\{-\frac{(x+b-2y)^2}{(2(1-t))}\Big\}\Big)\\
&\quad\int_0^t\frac{1}{s^{3/2}(t-s)^{3/2}}\exp\Big\{-\frac{y^2}{2s}-\frac{(x-y)^2}{2(t-s)}\Big\}ds\\
&+\frac{b-y}{\sqrt{t}}\Big(\exp\Big\{-\frac{x^2}{2t}\Big)-\exp\Big\{\frac{(x-y)^2}{2t}\Big\}\Big)\\
&\quad\int_t^1\frac{1}{(1-s)^{3/2}(s-t)^{3/2}}\exp\Big\{-\frac{-(b-y)^2}{2(1-s)}-\frac{(x-y)^2}{2(s-t)}\Big\}ds\Bigg]\ .
\end{aligned}\tag{3.7}$$

La densitat (3.7) que acabem d'obtenir involucra dues integrals en s que no tenen primitiva elemental, però l'integrand és una funció relativament "bona" per a la integració numèrica: malgrat els denominadors, que s'anul·len en els extrems, no hi ha singularitats gràcies a les exponencials. El comportament en t prop del 0 i el 1 no és tan bo, com és d'esperar perquè per $t=0$ i $t=1$ no hi pot haver densitat.

Per calcular l'esperança $\mathrm{E}[L(B_t,m(B))]$ de qualsevol funció L de les dues variables cal fer la integral

$$\int_{-\infty}^{0\wedge b}\int_y^{\infty}L(x,y)\cdot f_{(B_t,m(B))}(x,y)\,dx\,dy\ .$$

que només pot ser tractada numèricament. La integral triple involucrada es pot calcular per aplicació recursiva de rutines com ara les contingudes en el paquet `quadpack`, que ja hem utilitzat en altres casos. No obstant, per una banda, l'error que es comet és més difícil d'estimar, puix que cada avaluació d'una integral externa ve afectada pel l'error de les integrals més internes; per altra banda, els paquets de càlcul simbòlic tenen dificultats amb el pas de paràmetres quan cal cridar aquestes rutines recursivament.

Aquest és el cas del `maxima`, amb el que ha estat impossible avaluar la integral, fins i tot per $L\equiv 1$. Amb `octave` ha estat possible utilitzant la seva rutina `triplequad`, però el temps d'execució és gran. El que hem fet, doncs, en aquest cas, és usar `quadpack` directament en `C`. El programa resultant és a l'Annex

B.4. El programa permet modificar fàcilment la funció que defineix L, i els paràmetres t i b.

Una funció de pèrdua que sembla natural si es vol trobar un punt prometedor dins de l'interval de definició d'un pont és $L(x,y) = x - y$ o bé també $L(x,y) = (x-y)^2$. Però aquestes funcions no proporcionen cap nou punt a l'interior. En efecte:

Suposem sense pèrdua de generalitat que tenim el pont brownià de $(0,0)$ a $(1,b)$, amb $b \geq 0$. Aleshores, $\mathrm{E}[B_t - m(B)] = \mathrm{E}[B_t] - \mathrm{E}[m(B)]$ i la primera esperança descriu la recta $t \mapsto bt$. Per tant, el mínim és en $t = 0$ i no ens proporciona un nou punt. El mateix s'observa experimentalment amb $L(x,y) = (x-y)^2$.

No obstant, hi ha una funció de pèrdua més natural: Si el pont, avaluat en un cert $t \in (0,1)$, produeix un valor positiu, aleshores el millor punt entre els dos extrems i aquest t és l'extrem esquerre. Per tant, és més lògic buscar el mínim de $\mathrm{E}[\min(B_t,0) - m(B)]$, i aquesta funció de t sí té un mínim a l'interior de $(0,1)$. Veiem-ho posant, per simplificar, $b = 1$.

Només cal estudiar $\varphi(t) := \mathrm{E}[\min(B_t,0)]$, que és

$$\varphi(t) = \int_{-\infty}^{0} \frac{x}{\sqrt{2\pi t(1-t)}} \exp\left\{\frac{-(t-x)^2}{2t(1-t)}\right\} dx\,, \quad t \in (0,1)\,, \tag{3.8}$$

(vegeu l'Apartat 2 del Capítol 2), i $\varphi(0) = \varphi(1) = 0$.

La integral es pot calcular analíticament, i en resulta

$$\varphi(t) = \frac{t}{2}\,\mathrm{erfc}\left(\sqrt{\frac{t}{2(1-t)}}\right) - \sqrt{\frac{t(1-t)}{2\pi}} \exp\left\{\frac{-t}{2(1-t)}\right\}\,,$$

d'on la seva derivada és

$$\varphi'(t) = \frac{1}{2}\,\mathrm{erfc}\left(\sqrt{\frac{t}{2(1-t)}}\right) - \frac{1-2t}{\sqrt{8\pi t(1-t)}} \exp\left\{\frac{-t}{2(1-t)}\right\}\,, \quad t \in (0,1)\,. \tag{3.9}$$

Observem que $\varphi(t)$ és contínua en $[0,1]$. Per altra banda, $\lim_{t\to 0} \varphi'(t) = -\infty$, i $\lim_{t\to 1} \varphi'(t) = 0$. I també veiem que per $\frac{1}{2} < t < 1$, els dos sumands de (3.9) són estrictament positius. De fet, $\varphi'(\frac{1}{2}) = \frac{1}{2}\,\mathrm{erfc}\left(\frac{1}{\sqrt{2}}\right) > 0$. Per tant, com que φ' no pot ser creixent en tot $(0,1)$, la funció φ no és convexa.

No obstant, sí que es dedueix, per continuïtat, que φ és estrictament creixent en algun interval $(\frac{1}{2} - \varepsilon, 1]$. Veurem ara que φ és estrictament convexa en $[0, \frac{1}{2}]$, i això ens donarà l'existència d'un únic mínim local en $[0,1]$, que també serà el mínim global.

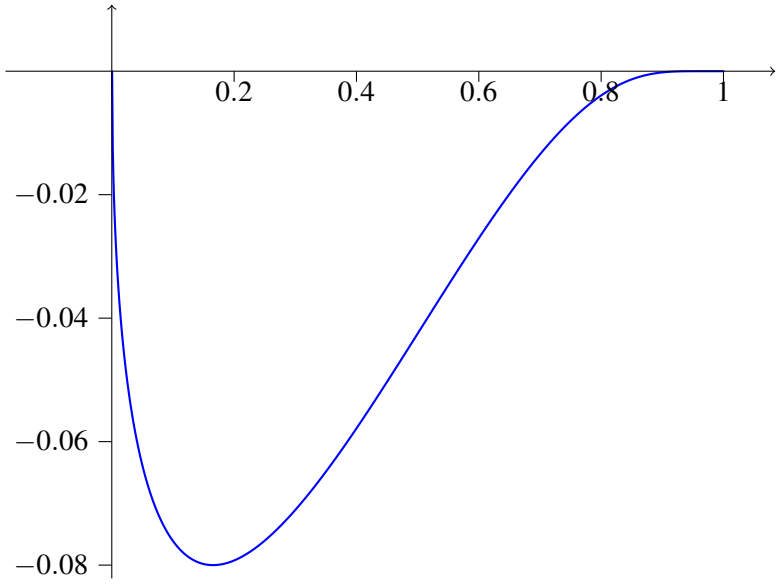

Figura 3.3: Funció φ definida per (3.8).

A tal fi, derivem altre cop:

$$\begin{aligned}\varphi''(t) = &\frac{1}{\sqrt{2\pi}}\exp\left\{\frac{-t}{2(1-t)}\right\}\times \\ &\times\left[\frac{1}{\sqrt{t(1-t)}} - \frac{1}{\sqrt{4t(1-t)^3}} + \frac{(1-2t)^2}{\sqrt{16t^3(1-t)^3}} + \frac{1-2t}{\sqrt{16t(1-t)^3}}\right].\end{aligned} \tag{3.10}$$

Els dos últims sumands de (3.10) són clarament positius per $t \leq \frac{1}{2}$. I la suma dels dos primers es veu fàcilment que també és positiva si i només si $t \leq \frac{1}{2}$. Això acaba el raonament. La Figura 3.3 il·lustra la gràfica de φ, calculada amb `R`.

A la Taula 3.7 hi ha el càlcul numèric de $\mathrm{E}\left[\min(B_t,0) - m(B)\right]$, en el cas particular $B_t \sim B_t^{(0,0)(1,1)}$ i un increment de $\Delta t = 0.05$, i s'observa que el mínim s'obté al voltant de $t = 0.15$. A la Taula 3.8 hem fet el mateix per al cas de funció quadràtica $\mathrm{E}\left[(\min(B_t,0) - m(B))^2\right]$. S'observa experimentalment que també hi ha un sol mínim global, i en aquest cas està prop de $t = 0.20$. En ambdós casos hem usat el codi en `C` esmentat abans a l'Annex B.4.

Per tant, aquest funció de pèrdua sí produeix un algorisme per generar un punt raonable a l'interior d'un subinterval $[t_i, t_{i+1}]$ i per tant es pot combinar amb alguna de les estratègies vistes als apartats 3.4 i 3.5. Deixem aquest estudi per a un treball futur.

t	$\mathrm{E}[(\min(B_t,0)-m(B))]$
0.05	0.26361
0.10	0.25156
0.15	0.24799
0.20	0.24871
0.25	0.25207
0.30	0.25718
0.35	0.26350
0.40	0.27064
0.45	0.27828
0.50	0.28617
0.55	0.29407
0.60	0.30171
0.65	0.30886
0.70	0.31523
0.75	0.32051
0.80	0.32443
0.85	0.32680
0.90	0.32773
0.95	0.32782

Taula 3.7: Càlcul numèric de $\mathrm{E}[\min(B_t,0)-m(B)]$, en el cas particular $B_t \sim B_t^{(0,0)(1,1)}$ i un increment de $\Delta t = 0.05$, amb un error màxim estimat de 0.00092.

t	$\mathrm{E}[(\min(B_t,0)-m(B))^2]$
0.05	0.11880
0.10	0.10377
0.15	0.09680
0.20	0.09452
0.25	0.09548
0.30	0.09880
0.35	0.10391
0.40	0.11035
0.45	0.11775
0.50	0.12576
0.55	0.13408
0.60	0.14237
0.65	0.15030
0.70	0.15749
0.75	0.16356
0.80	0.16813
0.85	0.17092
0.90	0.17205
0.95	0.17216

Taula 3.8: Càlcul numèric de $\mathrm{E}[(\min(B_t,0)-m(B))^2]$, en el cas particular $B_t \sim B_t^{(0,0)(1,1)}$ i un increment de $\Delta t = 0.05$, amb un error màxim estimat de 0.00076.

3.7 Conclusions i comentaris

Els algorismes estudiats en aquest capítol responen a dues filosofies:

- Seleccionar primer un interval prometedor segons un cert criteri, i després seleccionar un punt d'aquest interval, amb un altre cert criteri (Apartat 3.4), o bé
- Seleccionar un punt prometedor de cada interval, i entre aquests seleccionar el millor (Apartat 3.5).

En tots ells, la funció objectiu s'avaluarà només en el punt finalment seleccionat, i el valor de la funció en aquest punt, juntament amb els valors ja obtinguts prèviament, guiarà la selecció del punt següent. Computacionalment, la selecció del punt pot involucrar una simulació, en base al model probabilístic que s'està postulant; però se suposa que l'avaluació de la funció objectiu és molt més costosa que aquesta simulació.

Tant en una com en l'altra filosofia, la mediana $\mathbb{M}[\theta(B^i)]$ i l'esperança $\mathrm{E}[\theta(B^i)]$ de l'argument dels ponts brownians B^i en cada interval $[t_i, t_{i+1}]$ tenen un paper rellevant, i corresponen a minimitzar les funcions de pèrdua $\mathrm{E}\left[|t-\theta(B^i)|\right]$ i $\mathrm{E}[\left|t-\theta(B^i)|\right]^2$ *dins* de cada subinterval.

Hom podria demanar-se la possibilitat d'usar aquestes funcions de pèrdua de manera "global"; és a dir, donats ja n punts a l'interior de l'interval inicial $[0,1]$, i per tant un procés X amb la llei del pont brownià condicionat als punts ja mostrejats $(t_1,x_1),\ldots,(t_n,x_n)$, buscar $t = \arg\min_{[0,1]} \mathrm{E}\left[|t-\theta(X)|\right]$ o bé $t = \arg\min_{[0,1]} \mathrm{E}\left[|t-\theta(X)|^2\right]$, que donarien lloc a la mediana $\mathbb{M}[\theta(X)]$ i l'esperança $\mathrm{E}[\theta(X)]$.

El problema d'aquesta estratègia és que $\theta(X)$ és multimodal sempre, i en aquesta situació no tenen sentit pràctic la mediana i l'esperança com a mesures centrals. Com a exemple extrem, si X passa pels punts $(0,0),(0.5,0),(1,0)$, tant la mediana com l'esperança donaran $t = 0.5$ altre cop, i l'algorisme no podria avançar.

Sí podria tenir sentit, en canvi, usar en sentit global la funció de pèrdua anàloga a la que hem vist a l'Apartat 3.6, $\mathrm{E}\left[\min(B_t,0) - m(B)\right]$, que seria aleshores

$$\mathrm{E}\left[\min(X_t, x_0, \ldots, x_{n+1}) - m(X)\right] .$$

No hem estudiat aquesta possibilitat, però podria tenir interès.

Una altra possibilitat que no hem estudiat és l'ús d'estratègies mixtes, combinant les proposades. Com s'ha vist, hi ha algorismes (de fet tots, tard o d'hora) que tendeixen a explotar regions que es creuen

prometedores en detriment de l'exploració d'altres que poden tenir una probabilitat no negligible de contenir el punt òptim. La barreja d'estratègies o els mètodes de "recomençar" mostrejant primer un punt a l'atzar podrien mitigar en part el problema.

Bibliografia

[1] Aureli Alabert, Alessandro Berti, Ricard Caballero, and Marco Ferrante. No-free-lunch theorems in the continuum. *Theoretical Computer Science*, 600:98 – 106, 2015.

[2] Aureli Alabert and Ricard Caballero. On the minimum of a conditioned Brownian bridge.

[3] Robert B. Ash. *Real analysis and probability*. Academic Press, New York-London, 1972. Probability and Mathematical Statistics, No. 11.

[4] Søren Asmussen, Peter Glynn, and Jim Pitman. Discretization error in simulation of one-dimensional reflecting Brownian motion. *Ann. Appl. Probab.*, 5(4):875–896, 1995.

[5] Søren Asmussen and Peter W. Glynn. *Stochastic simulation: algorithms and analysis*, volume 57 of *Stochastic Modelling and Applied Probability*. Springer, New York, 2007.

[6] Anne Auger and Olivier Teytaud. Continuous lunches are free! In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 916–922, New York, NY, USA, 2007. ACM.

[7] Anne Auger and Olivier Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, 2010.

[8] Andrei N. Borodin and Paavo Salminen. *Handbook of Brownian motion—facts and formulae*. Probability and its Applications. Birkhäuser Verlag, Basel, second edition, 2002.

[9] James M. Calvin. Average performance of passive algorithms for global optimization. *J. Math. Anal. Appl.*, 191(3):608–617, 1995.

[10] James M. Calvin. Average performance of a class of adaptive algorithms for global optimization. *Ann. Appl. Probab.*, 7(3):711–730, 1997.

[11] James M. Calvin. Lower bound on complexity of optimization of continuous functions. *J. Complexity*, 20(5):773–795, 2004.

[12] M. M. Crum. On positive-definite functions. *Proc. London Math. Soc. (3)*, 6:548–560, 1956.

[13] Endre Csáki, Antónia Földes, and Paavo Salminen. On the joint distribution of the maximum and its location for a linear diffusion. *Ann. Inst. H. Poincaré Probab. Statist.*, 23(2):179–194, 1987.

[14] Christian Igel and Marc Toussaint. A no-free-lunch theorem for non-uniform distributions of target functions. *J. Math. Model. Algorithms*, 3(4):313–322, 2004.

[15] Kiyosi Itô and Henry P. McKean, Jr. *Diffusion processes and their sample paths*. Springer-Verlag, Berlin-New York, 1974. Second printing, corrected, Die Grundlehren der mathematischen Wissenschaften, Band 125.

[16] Ioannis Karatzas and Steven E. Shreve. *Brownian motion and stochastic calculus*, volume 113 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1991.

[17] Fima C. Klebaner. *Introduction to stochastic calculus with applications*. Imperial College Press, London, second edition, 2005.

[18] Hui-Hsiung Kuo. *Introduction to stochastic integration*. Universitext. Springer, New York, 2006.

[19] M. Locatelli. Bayesian algorithms for one-dimensional global optimization. *J. Global Optim.*, 10(1):57–76, 1997.

[20] Robert Piessens, Elise de Doncker-Kapenga, Christoph W. Überhuber, and David K. Kahaner. *QUADPACK*, volume 1 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1983. A subroutine package for automatic integration.

[21] Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1991.

[22] Klaus Ritter. Approximation and optimization on the Wiener space. *J. Complexity*, 6(4):337–364, 1990.

[23] J. E. Rowe, M. D. Vose, and A. H. Wright. Reinterpreting no free lunch. *Evolutionary Computation*, 17(1):117–129, 2009.

[24] C. Schumacher, M. D. Vose, and L. D. Whitley. The no free lunch and problem description length. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 565–570. Morgan Kaufmann, 2001.

[25] Loris Serafino. Derivatives: What does the no free lunch theorem actually say? *Notices of the AMS*, 61(7):750–755, 2014.

[26] Cristina P. Sison and Joseph Glaz. Simultaneous confidence intervals and sample size determination for multinomial proportions. *J. Amer. Statist. Assoc.*, 90(429):366–369, 1995.

[27] Charles J. Stone, Mark H. Hansen, Charles Kooperberg, and Young K. Truong. Polynomial splines and their tensor products in extended linear modeling. *Ann. Statist.*, 25(4):1371–1470, 1997. With discussion and a rejoinder by the authors and Jianhua Z. Huang.

[28] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 1(1):67–82, 1997.

[29] Anatoly Zhigljavsky and Antanas Žilinskas. *Stochastic global optimization*, volume 9 of *Springer Optimization and Its Applications*. Springer, New York, 2008.

Annex

Apèndix A

Rutines del Capítol 2: Minimum of conditioned Brownian bridge

A.1 Simulacions de trajectòries

creabrownia(punts)

```
/*procediment que crea un brownia en una llista amb el nombre de punts indicats*/
creabrownia(punts):=block([aux,aux2],
aux:create_list(random_normal(0,sqrt(1/punts)),i,1,punts),aux[1]:0,
for i:2 thru punts do(aux[i]:aux[i-1]+aux[i]),
return(aux)
);
```

dibuixapont(A)

```
/*procediment per dibuixar el procés a partir d'un vector A*/
/*d'imatges del procés*/
dibuixapont(A):=block([col,aux2],
n:punts,
col:length(A),
aux:create_list((i-1)/(col-1),i,1,col),
plot2d ([discrete, aux, A],
[style,lines], [point_type,diamond], [color,red])
```

```
);
```

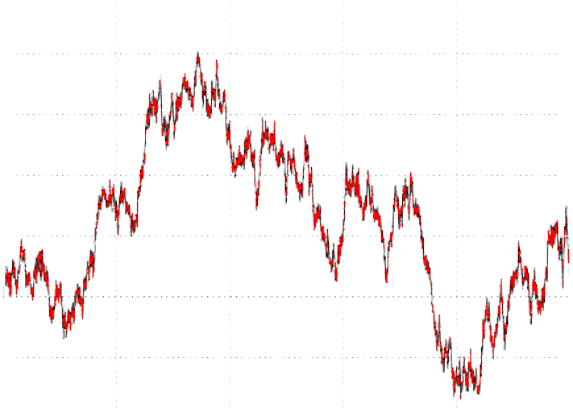

Figura A.1: Imatge .png que dóna wxmaxima amb la rutina dibuixapont() de la simulació d'un moviment brownià de 10000 punts

creapontbrownia(punts,t0,x0,t1,x1)

```
/*procediment que crea un pont brownià en una llista amb el nombre*/
 /*de punts indicats*/
creapontbrownia(punts,t0,x0,t1,x1):=block([aux,aux2,aux3,temps],
aux:create_list(random_normal(0,sqrt(1/punts)),i,1,punts),aux[1]:0,
for i:2 thru punts do(aux[i]:aux[i-1]+aux[i]),
aux2:create_list(0,i,1,punts),
aux3:create_list(x0,i,1,punts),
temps:create_list(i/(punts-1),i,0,punts-1),
for i:1 thru punts do(aux2[i]:aux[i]-temps[i]*aux[punts]),print(aux2[punts]),
for i:1 thru punts do(aux3[i]:(t1-temps[i])*x0+(temps[i]-t0)*x1+aux2[i]),
return(aux3)
);
```

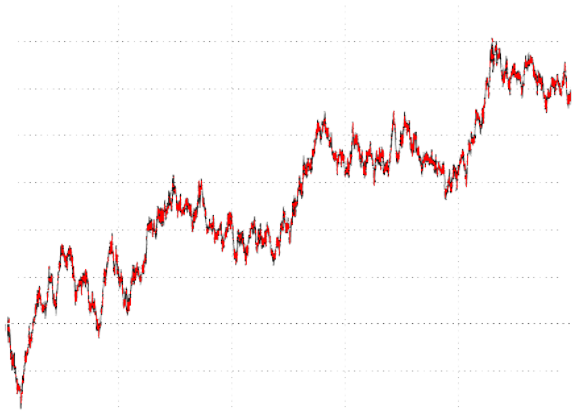

Figura A.2: Imatge .png que dóna maxima amb la rutina dibuixapont() de la simulació d'un pont brownià de $(0,0)(1,1)$ de 10000 punts

A.2 Càlcul de les probabilitat P_i

F(vt,vx,x,j)

Input:vt vector de t_i, vx vector d'imatges x_i de t_i, x la variable que tindrà la funció que retorna, j l'interval del qual se'n voldrà calcular la probabilitat P_j.

Output: Funció productori de l'integrand de les probabilitats P_j.

```
/*Funció productori de l'integrand de les probabilitats P_j*/
F(vt,vx,x,j):=block(
[col],
col:length(vx),
2/(vt[j+1]-vt[j])*(vx[j+1]+vx[j]-2*x)*
exp(-2*(vx[j]-x)*(vx[j+1]-x)/(vt[j+1]-vt[j]))
*product( if i=j then 1 else 1-exp(-2*(vx[i]-x)
*(vx[i+1]-x)/(vt[i+1]-vt[i])),i,1,col-1)
)$
```

P_i(vt,vx,j)

Input:vt vector de t_i, vx vector d'imatges x_i de t_i, j l'interval del qual se'n voldrà calcular la probabilitat P_j.

Output: Probabilitat P_j calculada utilitzant la primitiva de $F(vt, vx, x, j)$.

```
/*Probabilitat de forma analítica en un interval j*/
P_i(vt,vx,j):=block(
[col,m,a],col:length(vx),m:apply(min,vx),
a:2/(vt[j+1]-vt[j])*integrate((vx[j+1]+vx[j]-2*x)
*exp(-2*(vx[j]-x)*(vx[j+1]-x)/(vt[j+1]-vt[j]))
*product( if i=j then 1 else 1-exp(-2*(vx[i]-x)
*(vx[i+1]-x)/(vt[i+1]-vt[i])),i,1,col-1),x,minf,m)
)$
```

P_i_num(vt,vx,j)

Input: vt vector de t_i, vx vector d'imatges x_i de t_i, j l'interval del qual se'n voldrà calcular la probabilitat P_j.

Output: Probabilitat numèrica P_j de l'interval (t_j, t_{j+1}), cridant la rutina F(vt,vx,x,i) i utilitzant `quadpack`.

```
/*Probabilitat numèrica en un interval i, utilitzant QUADPACK*/
P_i_num(vt,vx,j):=block(
[i:i,col:col,maxim:maxim,a:a,res:res,minim:minim],
col:length(vt),minim:apply(min,vx),

res:quad_qagi(F(vt,vx,x,j),x,minf,minim)[1],res
)$
```

Romberg_cvP_i(vt,vx,j)

Input:vt vector de t_i, vx vector d'imatges x_i de t_i, j l'interval del qual se'n voldrà calcular la probabilitat P_j.

Output: Probabilitat numèrica P_j de l'interval (t_j, t_{j+1}), utilitzant el canvi de variable $x = minim(x_i) + 1 - 1/t$ i la rutina romberg de maxima per al càlcul de la integral.

```
load(romberg);rombergtol : 1e-6;
```

```
/*Probabilitat $P_i$ calculada pel métode de Romberg, amb el*/
/* canvi de variable x=m+1-1/t*/
Romberg_cvP_i(vt,vx,j):=block(
[col,m,a],col:length(vx),m:apply(min,vx),
a:2/(vt[j+1]-vt[j])*romberg((vx[j+1]+vx[j]-2
*(m+1-1/x))/x^2*exp(-2*(vx[j]-(m+1-1/x))*
(vx[j+1]-(m+1-1/x))/(vt[j+1]-vt[j]))
*product( if i=j then 1 else 1-exp(-2*(vx[i]-(m+1-1/x))
*(vx[i+1]-(m+1-1/x))/(vt[i+1]-vt[i])),i,1,col-1),x,0.0000001,1)
)$
```

SumRiemman(vt,vx,j,num)

Input:vt vector de t_i, vx vector d'imatges x_i de t_i, j l'interval del qual se'n voldrà calcular la probabilitat P_j, num nombre de divisions a fer de l'interval.

Output: Probabilitat numèrica P_j de l'interval (t_j, t_{j+1}), utilitzant el canvi de variable $x = minim(x_i) + 1 - 1/t$ i sumes de Riemman amb un punt a l'atzar de cada divisió d'interval.

```
/*Probabilitat per sumes de Riemman en un interval
i amb el canvi de variable x=m+1-1/t*/
SumRiemman(vt,vx,j,num):=block(
[col,m,a,b],col:length(vx),m:apply(min,vx),
f:lambda([x],2/(vt[j+1]-vt[j])*((vx[j+1]+vx[j]-2*(m+1-1/x))/x^2
*exp(-2*(vx[j]-(m+1-1/x))*(vx[j+1]-(m+1-1/x))/(vt[j+1]-vt[j]))
*product( if i=j then 1 else 1-exp(-2*(vx[i]-(m+1-1/x))
*(vx[i+1]-(m+1-1/x))/(vt[i+1]-vt[i])),i,1,col-1))),

sum(1/num*(f(0+(i-1)*1/num+random(1.0)*1/num)),i,1,num)
)$
```

SumRiemmanesq(vt,vx,j,num)

Input:vt vector de t_i, vx vector d'imatges x_i de t_i, i l'interval del qual se'n voldrà calcular la probabilitat P_i, num nombre de divisions a fer de l'interval.

Output: Probabilitat numèrica P_j de l'interval (t_j, t_{j+1}), utilitzant el canvi de variable $x = minim(x_i) + 1 - 1/t$ i sumes de Riemman amb el punt de l'esquerra de cada divisió d'interval.

```
/*Probabilitat per sumes de Riemman agafant l'extrem esquerre
d'un interval i amb el canvi de variable x=m+1-1/t*/
SumRiemmanesq(vt,vx,j,num):=block(
[col,m,a,b],col:length(vx),m:apply(min,vx),

f:lambda([x],2/(vt[j+1]-vt[j])*((vx[j+1]+vx[j]-2*(m+1-1/x))/x^2
*exp(-2*(vx[j]-(m+1-1/x))*(vx[j+1]-(m+1-1/x))/(vt[j+1]-vt[j]))
*product( if i=j then 1 else 1-exp(-2*(vx[i]-(m+1-1/x))
*(vx[i+1]-(m+1-1/x))/(vt[i+1]-vt[i])),i,1,col-1))),

sum(1/num*(f((i-1)*1/num)),i,2,num)
)$
```

Psub_isimulada(vt,vx,n)

Input:vt vector de t_i, vx vector d'imatges x_i de t_i, n nombre de mínims simulats.

Output: Probabilitat simulada de totes les P_i, utilitzant la simulació d'un mínim per a cada interval (t_i, t_{i+1}) i contant quin interval l'ha tingut més petit. L'aproximació de cada P_i és la proporció d'aquest recompte.

```
load (distrib)$
Psub_isimulada(vt,vx,n):=block(
[col,i,j,r,L1,pos,posicio],
col:length(vx),
cont:create_list(0,j,1,col-1),
for i:1 thru n do (
L1:create_list((vx[j]+vx[j+1]
-sqrt((vx[j]+vx[j+1])^2-4*(vx[j]*vx[j+1]+(1/2)
*(log(random_continuous_uniform (0,1)))
*(vt[j+1]-vt[j])))*(1/2),j,1,col-1),
posicio: ?position(apply(min,L1),L1),
cont[posicio]:cont[posicio]+1
),
cont/n
```

```
)$
```

donat_d2_trobar_p(vp)

Input: vp vector de probabilitats amb 0.5<p<1.

Output: la imatge d2 del pont brownià que va de $(0.5,0)$ a $(1,d2)$ i que té una probabilitat p de tenir el mínim del pont condicionat als punts $(0,0)$, $(0.5,0)$ i $(1,d2)$, utilitzant el funcional p(d2) de l'exemple (2.3.1).

```
p(d2):=0.5+sqrt(%pi/8)*d2*exp(d2^2/2)*(1-erf(d2/sqrt(2)));


donat_p_trobar_d2(vp):=block([col:length(vp),i],


create_list(find_root(p(d2)=vp[i],d2,0.00001,7),i,1,col)


)$
```

Histogramimatge(vt,vx,n)

Input:vt vector de t_i, vx vector d'imatges x_i de t_i, n nombre de mínims simulats.

Output: Un vector de longitud n, que conté els mínims de cada una de les simulacions d'un mínim per a cada interval (t_i,t_{i+1}).

A partir de les dades d'aquest vector, el paquet `logspline` de `R` construeix l'histograma de la figura 2.2.

```
Histogramimatge(vt,vx,n):=block(
[col,i,j,r,L1,pos],
col:length(vx),
cont:create_list(0,j,1,n),
for i:1 thru n do (
L1:create_list((vx[j]+vx[j+1]
-sqrt((vx[j]+vx[j+1])^2-4*(vx[j]*vx[j+1]+(1/2)
*(log(random_continuous_uniform (0,1)))
*(vt[j+1]-vt[j])))*(1/2),j,1,col-1),
```

```
cont[i]:apply(min,L1)
),
cont
)$
```

Simupuntdelpont(t0,x0,t1,x1,t)

Input: les coordenades t0,x0, t1 i x1 del principi i final del pont, t punt que es troba a l'interval $(t_0, t1)$.

Output:la simulació de la imatge de t seguint una llei de pont brownià $B_t^{(t0,x0)(t1,x1)}$.

```
/*Procediment que crea la simulació d'un punt xi d'un pont brownià*/
/* donat t0,x0,t1,x1,t */
Simupuntdelpont(t0,x0,t1,x1,t):=block([],
random_normal(x0+(t-t0)*(x1-x0)/(t1-t0),sqrt((t-t0)*(t1-t)/(t1-t0)))

)$
```

Vectorpont(vt,x0,x1)

Input: vt vector ordenat de t_i, x0 imatge inicial del pont en vt[1], x1 imatge final del pont vt[length(vt)].

Output: un vector que conté la simulació de les imatges del vector vt seguint una llei de pont $B_t^{(vt[1],x0)(vt[length(vt)],x1)}$.

```
/*Procediment que crea la simulació d'un vector xi d'un pont*/
/* de principi x0 i final x1 donat un vector ti*/
Vectorpont(vt,x0,x1):=block(
[col,k,i,vx],
col:length(vt),
vx:makelist(0,k,1,col),
vx[1]:x0,vx[col]:x1,
for i from 2 thru col-1 do (vx[i]:Simupuntdelpont(vt[i-1],vx[i-1],
vt[col],x1,vt[i]) ),
vx
)$
```

insertL1enL(L,L1,i)

Input: L llista, L1 llista, i posició de L.

Output:La llista L amb la llista L1 inserida a la posició i.

```
insertL1enL(L,L1,i):=block(
L[i]:L1,flatten(L))$
```

filtro(x)

Input:x vector ordenat.

Output: la mínima distància entre els elements de x.

```
filtro(x):=block(
[col,l],
col:length(x),
l:create_list(abs(x[i+1]-x[i]),i,1,col-1),
apply(min,l))$
```

rndEstimadorminpontx(t0,x0,t1,x1,n,nsimus)

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts.

Output: Si nsimus#0, retorna en un vector el mínim de les imatges que ha trobat rastrejant aleatòriament, el mínim simulat del pont condicionat als n punts i la diferència entre aquests dos. Si nsimus=0 retorna el mínim de les imatges que ha trobat rastrejant aleatòriament.

```
rndEstimadorminpontx(t0,x0,t1,x1,n,nsimus):=block(
[minimimatge,minimx,meanhist,minvx,vt,vx,i,vmins],
/*crea un vector de n valors aleatòris entre t0 i t1*/
vt:sort(append([t0,t1],random_continuous_uniform (t0,t1,n))),
/*li simula una trajectòria que passi per vt*/
vx:Vectorpont(vt,x0,x1),
/*troba la mínima imatge dels punts de vt*/
minimimatge:float(apply(min,vx)),
```

```
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return(minimimatge)
)$
```

ondegvt(t,gvt)

Input:t un nombre i gvt un vector ordenat.

Output:la posició que li correspondria a t si pertanyés a gvt.

```
ondegvt(t,gvt):=
block(apply(max,sublist_indices(gvt,lambda([h],h<t))))$
```

eqdEstimadorminpontx(t0,x0,t1,x1,n,nsimus)

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts.

Output: Si nsimus#0, retorna en un vector amb el mínim de les imatges que ha trobat rastrejant de manera equidistant, el mínim simulat del pont condicionat als n punts i la diferència entre aquests dos. Si nsimus=0 retorna el mínim de les imatges que ha trobat rastrejant de manera equidistant.

```
eqdEstimadorminpontx(t0,x0,t1,x1,n,nsimus):=block(
[minimimatge:minimimatge,minvx,meanhist,minimx:minimx,vt,vx,i,vmins,a],
a:n+1,
/*crea un vector de n valors equidistants entre t0 i t1*/
vt:float(makelist(t0+(t1-t0)*(i-1)/a,i,1,a+1)),
/*li simula una trajectòria que passi per vt*/
vx:Vectorpont(vt,x0,x1),
/*troba la mínima imatge dels punts de vt*/
minimimatge:float(apply(min,vx)),
if nsimus # 0 then
```

```
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return(minimimatge)
)$
```

grndEstimadorminpontx(t0,x0,t1,x1,n,nsimus,gvt,gvx)

Input: t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont. Procediment: aquesta rutina és com rndEstimadorminpontx, però tenint en compte els punts que ja han avaluat altres mètodes, i així poder comparar-los amb la mateixa trajectòria.

Output: un vector amb el mínim de les imatges que ha trobat rastrejant de manera equidistant, i els vectors gvt i gvx ampliats amb els nous valors que hagi utilitzat.

```
grndEstimadorminpontx(t0,x0,t1,x1,n,gvt,gvx):=block(
[minimx,minvx:minvx,vt,vx,i,k,vmins],
vt:sort(append([t0,t1],random_continuous_uniform (t0,t1,n))),
vx:makelist(0,i,1,n+2),vx[1]:x0,vx[n+2]:x1,
for i:2 thru n+1 do(
nouarg:vt[i],
/*si el nou t que vol avaluar ja és de gvt, agafa el seu valor*/
/*de gvx*/
if member(nouarg,gvt) then
(novaimtge:gvx[?position(nouarg,gvt)],vx[i]:novaimtge)
else
/*si el nou t que vol avaluar no és de gvt, li asigna un valor*/
/*simulat depenent dels que ja tenim de gvt i de gvx*/
/*control*/
(k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],
gvx[k],gvt[k+1],gvx[k+1],nouarg),
```

```
vx[i]:novaimtge,
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos]
,novaimtge],novapos))
),
/*control*/
return([apply(min,vx),gvt,gvx])
)$
```

myg2(t,t1,x1,t2,x2,p1,t0)

Input: t variable límit superior de la integral de la densitat de l'argument, t1,x1 coordenades de l'inici del pont, t2,x2 coordenades del punt final del pont, p1 probabilitat, t0 limit inferior de la integral de la densitat de l'argument.

Output: una llista auxiliar que es fa servir com a funció $f(t) = -p1 + \int_{t0}^{t} f(s)ds$ on $f(s)$ és la densitat de l'argument.

```
myg2(t,t1,x1,t2,x2,p1,t0) := block([val,qlist, numer],
numer:true,
qlist:
quad_qags((x2-x1)*sqrt((2*(t2-t1))*(t2-s)/(%pi*(s-t1)))
*exp(-(x2-x1)^2*(s-t1)/((2*(t2-t1))*(t2-s)))/(t2-t1)^2
+(t2-t1-(x2-x1)^2)*erfc((x2-x1)*sqrt((s-t1)/((2*(t2-t1))
*(t2-s))))/(t2-t1)^2, s ,t0,t)-p1,
val : qlist[1],
val
)$
```

eqp(t0,x0,t1,x1,n)

Input:t0,x0 coordenades de l'inici del pont, t1,x1 coordenades del punt final del pont, n nombre de punts.

Procediment: utilitza la rutina find_root per trobar els zeros de myg2(t,t0,x0,t1,x1,1/(n+1),t0) a l'interval [t0+0.000000001,1]. (El maxima no podia treballar amb la singularitat a t0).

Output: n punts tals que els n-1 intervals que formen tenen la mateixa probabilitat de contenir el mínim del pont $B_t^{(t0,x0)(t1,x1)}$.

```
eqp(t0,x0,t1,x1,n) := block([vect,p1],
vect:create_list(0,i,1,n),
p1:1/(n+1),
vect[1]:find_root(myg2(t,t0,x0,t1,x1,1/(n+1),t0)
,t,t0+0.000000001,1),

for i from 2 thru n do(

vect[i]:find_root(myg2(t,t0,x0,t1,x1,p1,vect[i-1])
,t,vect[i-1],1)

),
vect
)$
```

geqpEstimadorminpontx(t0,x0,t1,x1,n,nsimus,gvt,gvx)

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont.

Procediment: aquesta rutina és com rndEstimadorminpontx, però tenint en compte els punts que ja han avaluat altres mètodes, d'aquesta manera per simular x_i es tenen en compte els x_{i-1} i x_{i+1} que ja es tenen i així es poden comparar els mètodes amb la mateixa trajectòria.

Output: un vector amb el mínim de les imatges que ha trobat rastrejant de manera equiprobable, i els vectors gvt i gvx ampliats amb els nous valors que hagi utilitzat.

```
geqpEstimadorminpontx(t0,x0,t1,x1,n,nsimus,gvt,gvx):=block(
[minimx,meanhist,minvx,vt,vx,i,k,vmins],
vt:sort(append([t0,t1],eqp(t0, x0, t1, x1, n))),
vx:makelist(0,i,1,n+2),vx[1]:x0,vx[n+2]:x1,
for i:2 thru n+1 do(


nouarg:vt[i],
```

```
if member(nouarg,gvt) then
(novaimtge:gvx[?position(nouarg,gvt)],vx[i]:novaimtge)
else
/*control*/
(k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont
(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
vx[i]:novaimtge,
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],
novaimtge],novapos))
),
/*control*/

return([apply(min,vx),gvt,gvx])
)$
```

eqd_rnd_eqp_enllacats(t0,x0,t1,x1,n,nsimus,printgvx)

Compara els tres mètodes eqd, rnd i eqp. Input:t0,x0 coordenades del punt inicial del pont, t1,x1: coordenades del punt final del pont, n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, printgvx: variable boolean, si val #0 es printen també les ordenades dels punts que s'han obtingut de la trajectòria.

Output: Si nsimus=0, una matriu amb el valor més baix que ha rastrejat cadascun dels tres mètodes. Si nsimus#0, la matriu anterior a més d'una altra matriu amb els errors de cada mètode respecte del mínim simulat de la trajectòria amb tots els punts que s'han utilitzat.

```
eqd_rnd_eqp_enllacats(t0,x0,t1,x1,n,nsimus,printgvx):=block(
[minimimatge,minvx,meanhist,minimx,vt,vx,i,vmins,a:n+1,
generalvt,generalvx,fetrnd,resdernd,feteqp,resdeeqp,finaloutput],

/*comença procés equidistant*/
vt:float(makelist(t0+(t1-t0)*(i-1)/a,i,1,a+1)),
```

```
vx:Vectorpont(vt,x0,x1),
minimimatge:float(apply(min,vx)),
generalvt:vt,generalvx:vx,
/*acaba procés equidistant*/

/*comença procés random*/
fetrnd:grndEstimadorminpontx(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdernd:fetrnd[1],generalvt:fetrnd[2],generalvx:fetrnd[3],
/*acaba procés random*/

/* comença procés eqp*/

feteqp:geqpEstimadorminpontx(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdeeqp:feteqp[1],generalvt:feteqp[2],generalvx:feteqp[3],
/*acaba procés eqp*/

if printgvx#0 then print(generalvx),
finaloutput:matrix(
["equidistant",minimimatge],
["random",resdernd],
["eqp",resdeeqp],

["dim de gvt",length(generalvt)-2]
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(generalvt,generalvx,nsimus)),
return([finaloutput,[abs(meanhist-minimimatge),abs(meanhist-resdernd),
abs(meanhist-resdeeqp)]]))
else
return(finaloutput)
)$
```

st_tots_vs_tots(t0,x0,t1,x1,n,nsimus,mcops)

Input:t0,x0 coordenades del punt inicial del pont, t1,x1: coordenades del punt final del pont, n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, mcops: nombre de vegades que es comparen els mètodes llançant la rutina anterior enllacats().

Output: una matriu amb la mitjana, desviació típica i intervals de confiança dels errors de cada mètode respecte del mínim simulat de la trajectòria amb tots els punts que s'han utilitzat.

```
st_tots_vs_tots(t0,x0,t1,x1,n,nsimus,mcops):=block(
[i,res,aux,finaloutput,meqd,seqd,mrnd,srnd,meqp,seqp,B],
reseqd:makelist(0,i,1,mcops),resrnd:makelist(0,i,1,mcops),


/*llancem enllacats mcops vegades i ho deixa en una matriu*/
B:apply(matrix,create_list(enllacats(t0,x0,t1,x1,n,
nsimus,0)[2],i,1,mcops)),

/*la primera columna és eqd, la segona rnd i la tercera eqp*/
meqd:mean(col(B,1))[1],
seqd:std(col(B,1))[1],
mrnd:mean(col(B,2))[1],
srnd:std(col(B,2))[1],
meqp:mean(col(B,3))[1],
seqp:std(col(B,3))[1],

/*ordenem tota la informació en una matriu*/
finaloutput:matrix(
["mitjana (eqd-minhist)",meqd],
["desvest (eqd-minhist)",seqd],
["I.C. 95% (eqd-minhist) ",float(meqd-1.96*seqd/sqrt(mcops))],
["I.C. 95% (eqd-minhist) ",float(meqd+1.96*seqd/sqrt(mcops))],

["mitjana (rnd-minhist)",mrnd],
```

```
["desvest (rnd-minhist)",srnd],
["I.C. 95% (rnd-minhist) ",float(mrnd-1.96*srnd/sqrt(mcops))],
["I.C. 95% (rnd-minhist) ",float(mrnd+1.96*srnd/sqrt(mcops))],



["mitjana (eqp-minhist)",meqp],
["desvest (eqp-minhist)",seqp],
["I.C. 95% (eqp-minhist) ",float(meqp-1.96*seqp/sqrt(mcops))],
["I.C. 95% (eqp-minhist) ",float(meqp+1.96*seqp/sqrt(mcops))]

),

return(finaloutput)

)$
```

A.3 Rutines utilitzades a la Secció 2.6 Simulating the law of the location of the minimum

argx2(t0,x0,t1,x1,y1)

Necessita les definicions de: fs(t0,x0,t1,x1,y,s) densitat de l'argument condicionada a un mínim y. y(t0,x0,t1,x1) simulació del mínim del pont (t0,x0)(t1,x1) Input: t0,x0 coordenades de l'inici del pont, t1,x1 coordenades del punt final del pont, y1 mínim de tot el pont.

Output:l'argument del mínim condicionat al mínim y1 mitjançant el mètode d'Acceptance-Rejection.

```
fs(t0,x0,t1,x1,y,s):=exp((x0-x1)^2/(2*t1-2*t0))
*sqrt(2*t1-2*t0)/sqrt(%pi*(s-t0)^3*(t1-s)^3)
*(y-x0)*(y-x1)*exp(-(y-x0)^2/(2*s-2*t0)-(y-x1)^2/(2*t1-2*s))
*(2*(x1+x0-2*y)/(t1-t0)*exp(-2*(x1-y)*(x0-y)/(t1-t0)))^(-1);

y(t0,x0,t1,x1):=(x0+x1
-sqrt((x0+x1)^2-4*(x0*x1+(1/2)*
```

```
(log(random_continuous_uniform (0,1)))
*(t1-t0))))*(1/2);


argx2(t0,x0,t1,x1,y1):=
block(
[j:j,aux:aux,i,col,vx,v,c,u1,u2],
v:map(rhs,float(realroots(-3/2*(-2*x+t1+t0)*(x-t0)*(t1-x)
+(t1-x)^2*(y1-x0)^2/2-(x-t0)^2*(y1-x1)^2/2,1e-20))),
col:length(v),
vx:create_list(0,i,1,col),
for i from 1 thru col do vx[i]:fs(t0,x0,t1,x1,y1,v[i]),
c:ev(apply(max,vx),numer),
u1:random_continuous_uniform (t0,t1,1)[1],
u2:random_continuous_uniform (0,1,1)[1],
cont:1,
while u2>ev(fs(t0,x0,t1,x1,y1,u1)/c,numer) do
(u1:random_continuous_uniform (t0,t1,1)[1],
u2:random_continuous_uniform (0,1,1)[1], cont:cont+1),
return(u1));
```

simuargmin(vt,vx,n)

Input: vt vector d'abscisses dels punts als que està condicionat el pont, vx vector d'ordenades dels punts als que està condicionat el pont, n nombre de simulacions.

Output:llista amb n simulacions de l'argument del mínim del pont condicionat als punts (t,x) que donen vt i vx.

```
simuargmin(vt,vx,n):=
block(
[j:j,aux:aux,i,k,col,v,c,argminlist,ymin],
col:length(vt),
argminlist:create_list(0,i,1,n),
for k from 1 thru n do (
```

```
vmins:create_list((vx[i]+vx[i+1]
-sqrt((vx[i]+vx[i+1])^2-4*(vx[i]*vx[i+1]+(1/2)
*(log(random_continuous_uniform (0,1)))
*(vt[i+1]-vt[i])))*(1/2),i,1,col-1),
ymin:apply(min,vmins),
i:?position(ymin,vmins),
argminlist[k]:argx2(vt[i],vx[i],vt[i+1],vx[i+1],ymin)
),
argminlist)$
```

Apèndix B

Rutines del Capítol 3: Algorismes adaptatius d'un pas per cercar el mínim del pont brownià

B.1 Rutines utilitzades a la Secció 3.3 Funcions de pèrdues involucrant t i $\theta(X)$

tayint(t1,x1,t2,x2)

Input:t1,x1 punt inicial del pont t2,x2 punt final del pont.

Output: l'esperança de la localització de l'argument del mínim.

```
tayint(t1,x1,t2,x2):=
if x2-x1<=0 then
quad_qags(-s*(x2-x1)*sqrt((2*(t2-t1))*(s-t1)/(%pi*(t2-s)))
*exp(-(x2-x1)^2*(t2-s)/((2*(t2-t1))*(s-t1)))/(t2-t1)^2
+s*(t2-t1-(x2-x1)^2)*erfc(-(x2-x1)*sqrt((t2-s)/((2*(t2-t1))
*(s-t1))))/(t2-t1)^2, s ,t1,t2)[1]
else
quad_qags(s*(x2-x1)*sqrt((2*(t2-t1))*(t2-s)/(%pi*(s-t1)))
*exp(-(x2-x1)^2*(s-t1)/((2*(t2-t1))*(t2-s)))/(t2-t1)^2
+s*(t2-t1-(x2-x1)^2)*erfc((x2-x1)*sqrt((s-t1)/((2*(t2-t1))
*(t2-s))))/(t2-t1)^2, s ,t1,t2)[1]$
```

myg3(t,t1,x1,t2,x2)

Input: t variable límit superior de la integral de la densitat de l'argument del mínim fdensmin(t1,x1,t2,x2,s), t1,x1 punt inicial del pont, t2,x2 punt final del pont.

Output: una llista auxiliar que es fa servir com a funció $f(t) = -0.5 + \int_0^t fdensmin(s)ds$ on fdensmin(s) és la densitat de l'argument.

```
fdensmin(t1,x1,t2,x2,s):=
if x2-x1<=0 then
-(x2-x1)*sqrt((2*(t2-t1))*(s-t1)/(%pi*(t2-s)))*exp(-(x2-x1)^2*(t2-s)
/((2*(t2-t1))*(s-t1)))/(t2-t1)^2+(t2-t1-(x2-x1)^2)
*erfc(-(x2-x1)*sqrt((t2-s)/((2*(t2-t1))*(s-t1))))/(t2-t1)^2
else
(x2-x1)*sqrt((2*(t2-t1))*(t2-s)/(%pi*(s-t1)))*exp(-(x2-x1)^2*(s-t1)
/((2*(t2-t1))*(t2-s)))/(t2-t1)^2+(t2-t1-(x2-x1)^2)
*erfc((x2-x1)*sqrt((s-t1)/((2*(t2-t1))*(t2-s))))/(t2-t1)^2$

myg3(t,t1,x1,t2,x2) := block([val,qlist, numer],
numer:true,
qlist: quad_qags(fdensmin(t1,x1,t2,x2,s), s ,0,t)-0.5,
val : qlist[1],
val
)$
```

medianaarg(t1,x1,t2,x2)

Input: t1,x1 punt inicial del pont, t2,x2 punt final del pont. Procediment: utilitza la rutina find_root per trobar els zeros de l'anterior myg3() a l'interval [t1+10^17,t2]. (El maxima no podia treballar amb la singularitat a t1).

Output: la mediana de l'argument del mínim.

```
medianaarg(t1,x1,t2,x2):=find_root(myg3(t,0,x1,t2-t1,x2)
,t,0.00000000000000001,t2-t1)+t1$
```

tdefdeperdues4(t0,x0,t1,x1)

Input: t0,x0 punt inicial del pont, t1,x1 punt final del pont.

Output: 0<t<1 solució de l'equació 4t^3-12t^2I_1+12tI_2-8I_3=0, la qual resulta de la funció de pèrdues (t-s)^4.

```
tdefdeperdues4(t0,x0,t1,x1):=(block[a,b,c],
a:quad_qags(s*fdensmin(t0,x0,t1,x1,s),s,t0,t1)[1],
b:quad_qags(s^2*fdensmin(t0,x0,t1,x1,s),s,t0,t1)[1],
c:quad_qags(s^3*fdensmin(t0,x0,t1,x1,s),s,t0,t1)[1],
float(realroots(4*t^3-12*a*t^2+12*t*b-4*c,0.00000001))
)$
```

B.2 Rutines utilitzades a la Secció 3.4 Mètodes de selecció d'interval

filtro(x)

Input:x vector ordenat.

Output: la mínima distància entre els elements de x.

```
filtro(x):=block(
[col,l],
col:length(x),
l:create_list(abs(x[i+1]-x[i]),i,1,col-1),
apply(min,l))$
```

espmin(t0,x0,t1,x1)

Input:t0,x0 coordenades del principi del pont, t1,x1 coordenades del final del pont.

Output: calcula l'esperança de la imatge del mínim del pont.

```
espmin(t0,x0,t1,x1):=
block([minxs,t],
minxs:apply(min,[x0,x1]),t:t1-t0,

float(minxs-0.25*sqrt(2*%pi*t)*exp((x1-x0)^2/(2*t))
*(1+erf((2*minxs-x0-x1)/sqrt(2*t))))

)$
```

vintvminiesp(vt,vx)

Input: vt,vx vectors amb les abscisses i ordenades dels punts per on passa el pont.

Output: l'interval que té l'esperança del mínim més petita.

```
vintvminiesp(vt,vx):=block(
[i:i,col:col,minim:minim,a:a,res:res],
col:length(vt),
a:1,
minim:espmin(vt[1],vx[1],vt[2],vx[2]),
for i:2 step 1 thru col-1 do (res:espmin(vt[i],vx[i],vt[i+1],vx[i+1]),
if res<minim then (minim:res,a:i)),
a
)$
```

gp_arg_minimpontesp(t0,x0,t1,x1,n,nsimus,gvt,gvx)

esp/esp

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont. Procediment: va seleccionant l'interval que té l'esperança del mínim més petita, i en aquest interval calcula l'esperança de l'argument. I així successivament.

Output: si nsimus=0 un vector amb el mínim de les imatges que ha trobat rastrejant en els intervals que tenen la esperança del mínim més petita, i els vectors gvt i gvx ampliats amb els nous valors que hagi utilitzat. si nsimus #0, el vector anterior i l'esperança de l'error respecte el mínim simulat dels punts que ha utilitzat només aquest procés.

```
gp_arg_minimpontesp(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt:vt,vx:vx,j,minvx,meanhist,aux,i,novaimtge,nouarg,novapos,k],
nouarg:tayint(t0,x0,t1,x1),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then (novaimtge:gvx[?position(nouarg,gvt)])
```

```
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(
for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (
j:vintvminiesp(vt,vx),
nouarg:tayint(vt[j],vx[j],vt[j+1],vx[j+1]),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/

vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
```

```
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return([apply(min,vx),gvt,gvx])
)$
```

mediana(t0, x0, t1, x1)

Input:t0,x0 punt inicial del pont t1,x1 punt final del pont.

Output: mediana del mínim del pont.

```
mediana(t0, x0, t1, x1):= 0.5*(x0+x1-sqrt((x0+x1)^2-4*x0*x1-2*log(.5)*(t1-t0)))$
```

vintvminmed(vt,vx)

Input: vt,vx vectors amb les abscisses i ordenades dels punts per on passa el pont.

Output: l'interval que té la mediana del mínim més petita.

```
vintvminmed(vt,vx):=block(
[i:i,col:col,a:a,res:res,minim:minim],
col:length(vt),
a:1,
minim:mediana(v[1],vx[1],v[2],vx[2]),
for i:2 step 1 thru col-1 do (res:mediana(v[i],vx[i],v[i+1],vx[i+1]),
if res<minim then (minim:res,a:i)),
a
)$
```

gp_arg_minimpontmed(t0,x0,t1,x1,n,nsimus,gvt,gvx)

med/esp

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont. Procediment: va seleccionant l'interval que té la mediana del mínim més petita, i en aquest interval calcula l'esperança de l'argument. I així successivament.

Output: si nsimus=0 un vector amb el mínim de les imatges que ha trobat rastrejant en els intervals que tenen la esperança del mínim més petita, i els vectors gvt i gvx ampliats amb els nous valors que hagi

utilitzat. si nsimus #0, el vector anterior i l'esperança de l'error respecte el mínim simulat dels punts que ha utilitzat només aquest procés.

```
gp_arg_minimpontmed(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt:vt,vx:vx,j:j,minvx:minvx,meanhist:meanhist,aux:aux,i,novaimtge:novaimtge,
     nouarg:nouarg,novapos:novapos,k:k],
nouarg:tayint(t0,x0,t1,x1),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(
for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (
j:vintvminmed(vt,vx),
nouarg:tayint(vt[j],vx[j],vt[j+1],vx[j+1]),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
```

```
/*control*/


vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return([apply(min,vx),gvt,gvx])
)$
```

vintvmaxpsubi(vt,vx)

Input: vt,vx vectors amb les abscisses i ordenades dels punts per on passa el pont.

Output: l'interval que té la màxima P_i.

```
vintvmaxpsubi(vt,vx):=block(
[i:i,col:col,maxim:maxim,a:a,res:res,minim:minim],
col:length(vt),minim:apply(min,vx),
a:1,
maxim:quad_qagi(F(vt,vx,x,1),x,minf,minim)[1],
for i:2 step 1 thru col-1 do (res:quad_qagi(F(vt,vx,x,i),x,minf,minim)[1],
if res>maxim then (maxim:res,a:i)),
a
)$
```

gp_arg_minimpontpsi(t0,x0,t1,x1,n,nsimus,gvt,gvx)

P_i/esp

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat

als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont. Procediment: va seleccionant l'interval que té la màxima probabilitat de contenir l'argument del mínim, i en aquest interval calcula l'esperança de l'argument. I així successivament.

Output: si nsimus=0 un vector amb el mínim de les imatges que ha trobat rastrejant en els intervals que tenen la esperança del mínim més petita, i els vectors gvt i gvx ampliats amb els nous valors que hagi utilitzat. si nsimus #0, el vector anterior i l'esperança de l'error respecte el mínim simulat dels punts que ha utilitzat només aquest procés.

```
gp_arg_minimpontpsi(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt,vx,j,meanhist,minvx,aux,i,novaimtge,nouarg,novapos,k],
nouarg:tayint(t0,x0,t1,x1),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(
for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (
j:vintvmaxpsubi(vt,vx),
nouarg:tayint(vt[j],vx[j],vt[j+1],vx[j+1]),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
```

```
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/

vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)])
)
else
return([apply(min,vx),gvt,gvx])
)$
```

gpminimpontesp(t0,x0,t1,x1,n,nsimus,gvt,gvx)

esp/med

El mateix que gp_arg_minimpontesp(), però agafant la mediana de l'argument en lloc de l'esperança.

```
gpminimpontesp(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt,vx,j,minvx,meanhist,aux,i,novaimtge,nouarg,novapos,k,vmesp],
nouarg:medianaarg(t0,x0,t1,x1),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then (novaimtge:gvx[?position(nouarg,gvt)])
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
```

```
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then

(vmesp:[espmin(t0,x0,nouarg,novaimtge),mediana(nouarg,novaimtge,t1,x1)],
for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (

j:?position(apply(min,vmesp),vmesp),

nouarg:medianaarg(vt[j],vx[j],vt[j+1],vx[j+1]),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/

vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos),

vmesp:insertL1enL(vmesp,[espmin(vt[novapos],vx[novapos],nouarg,novaimtge),
espmin(nouarg,novaimtge,vt[novapos+2],vx[novapos+2])],novapos)
))
```

```
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return([apply(min,vx),gvt,gvx])
)$
```

gpminimpontmed(t0,x0,t1,x1,n,nsimus,gvt,gvx)

med/med

El mateix que gp_arg_minimpontmed(), però agafant la mediana de l'argument en lloc de l'esperança.

```
gpminimpontmed(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt,vx,j,minvx,meanhist,aux,i,novaimtge,nouarg,novapos,k,vmmed],
nouarg:medianaarg(t0,x0,t1,x1),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(vmmed:[mediana(t0,x0,nouarg,novaimtge),mediana(nouarg,novaimtge,t1,x1)],

for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (
```

```
j:?position(apply(min,vmmed),vmmed),

nouarg:medianaarg(vt[j],vx[j],vt[j+1],vx[j+1]),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/

vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos),
/*aprofitar els càlculs anteriors*/
vmmed:insertL1enL(vmmed,[mediana(vt[novapos],vx[novapos],
nouarg,novaimtge),
mediana(nouarg,novaimtge,vt[novapos+2],vx[novapos+2])],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return([apply(min,vx),gvt,gvx])
)$
```

gpminimpontpsi(t0,x0,t1,x1,n,nsimus,gvt,gvx)

P_i/med

El mateix que gp_arg_minimpontpsi(), però agafant la mediana de l'argument en lloc de l'esperança.

```
gpminimpontpsi(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt,vx,j,meanhist,minvx,aux,i,novaimtge,nouarg,novapos,k],
nouarg:medianaarg(t0,x0,t1,x1),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(
for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (
j:vintvmaxpsubi(vt,vx),
nouarg:medianaarg(vt[j],vx[j],vt[j+1],vx[j+1]),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
```

```
vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)])
)
else
return([apply(min,vx),gvt,gvx])
)$
```

enllacats2(t0,x0,t1,x1,n,nsimus,printgvx)

Compara els 6 mètodes esp/esp, med/esp, P_i/esp, esp/med med/med i P_i/med. Input:t0,x0 coordenades del punt inicial del pont, t1,x1: coordenades del punt final del pont, n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, printgvx: variable boolean, si val #0 es printen també les ordenades dels punts que s'han obtingut de la trajectòria.

Output: Si nsimus=0, una matriu amb el valor més baix que ha rastrejat cadascun dels sis mètodes. Si nsimus#0, la matriu anterior a més d'una altra matriu amb els errors de cada mètode respecte del mínim simulat de la trajectòria amb tots els punts que s'han utilitzat.

```
enllacats2(t0,x0,t1,x1,n,nsimus,printgvx):=block(
[minimimatge,minvx,meanhist,minimx,vt,vx,i,vmins,a,
generalvt,generalvx,fetminiesp,
fetrnd,resdernd,resdeminiesp,fetminimed,
resdeminimed,resdepsi,fetpsi,

feteqp,resdeeqp,finaloutput,fetesparg,resdeesparg,fetmedarg,
resdemedarg,fetpsiarg,resdepsiarg],

/*comença miniespd'arg*/
```

```
fetesparg:gp_arg_minimpontesp(t0,x0,t1,x1,n,0,[t0,t1],[x0,x1]),
resdeesparg:fetesparg[1],generalvt:fetesparg[2], generalvx:fetesparg[3],
/*comença miniespd'arg*/

/*comença minimedd'arg*/
fetmedarg:gp_arg_minimpontmed(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdemedarg:fetmedarg[1],generalvt:fetmedarg[2], generalvx:fetmedarg[3],
/*comença minimedd'arg*/


/*comença max psi d'arg*/
fetpsiarg:gp_arg_minimpontpsi(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdepsiarg:fetpsiarg[1],generalvt:fetpsiarg[2], generalvx:fetpsiarg[3],
/*comença max psi d'arg*/

/*comença procés miniesp*/

fetminiesp:gpminimpontesp(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdeminiesp:fetminiesp[1],generalvt:fetminiesp[2],generalvx:fetminiesp[3],
/*acaba procés miniesp*/

/*comença procés minimed*/
fetminimed:gpminimpontmed(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdeminimed:fetminimed[1],generalvt:fetminimed[2],generalvx:fetminimed[3],
/*acaba procés minimed*/
/*comença procés psi*/
fetpsi:gpminimpontpsi(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdepsi:fetpsi[1],generalvt:fetpsi[2],generalvx:fetpsi[3],
/*acaba procés psi*/
if printgvx#0 then print(generalvx),
finaloutput:matrix(
["equidistant",minimimatge],
```

```
["random",resdernd],
["eqp",resdeeqp],
["minima esp",resdeminiesp],
["minima med",resdeminimed],["max psi",resdepsi],
["min de la trajectoria",float(apply(min,generalvx))],
["dim de gvt",length(generalvt)-2]
),
if nsimus # 0 then
(
minvx:apply(min,generalvx),meanhist:
mean(Histogramimatge(generalvt,generalvx,nsimus)),
return([finaloutput,[abs(meanhist-resdeesparg),
abs(meanhist-resdemedarg),abs(meanhist-resdepsiarg),
abs(meanhist-resdeminiesp),abs(meanhist-resdeminimed),
abs(meanhist-resdepsi)]]))
else
return(finaloutput)
)$
```

st_tots_vs_tots2

Input:t0,x0 coordenades del punt inicial del pont, t1,x1: coordenades del punt final del pont, n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, mcops: nombre de vegades que es comparen els mètodes llançant la rutina anterior enllacats2().

Output: una matriu amb la mitjana, desviació típica i intervals de confiança dels errors de cada mètode respecte del mínim simulat de la trajectòria amb tots els punts que s'han utilitzat.

```
st_tots_vs_tots2(t0,x0,t1,x1,n,nsimus,mcops):=block(
[i,res,aux,finaloutput,mit1,desvest1,mit2,desvest2,B],
reseqd:makelist(0,i,1,mcops),resrnd:makelist(0,i,1,mcops),

B:apply(matrix,create_list(enllacats2(t0,x0,t1,x1,
n,nsimus,0)[2],i,1,mcops)),
```

```
meqd:mean(col(B,1))[1],
seqd:std(col(B,1))[1],
mrnd:mean(col(B,2))[1],
srnd:std(col(B,2))[1],
meqp:mean(col(B,3))[1],
seqp:std(col(B,3))[1],
mesp:mean(col(B,4))[1],
sesp:std(col(B,4))[1],
mmed:mean(col(B,5))[1],
smed:std(col(B,5))[1],
mpsi:mean(col(B,6))[1],
spsi:std(col(B,6))[1],

finaloutput:matrix(
["mitjana (miniespamb_esparg-minhist)",meqd],
["desvest (miniespamb_esparg-minhist)",seqd],
["I.C. 95% (miniespamb_esparg-minhist) ",
float(meqd-1.96*seqd/sqrt(mcops))],
["I.C. 95% (miniespamb_esparg-minhist) ",
float(meqd+1.96*seqd/sqrt(mcops))],

["mitjana (minimedamb_esparg-minhist)",mrnd],
["desvest (minimedamb_esparg-minhist)",srnd],
["I.C. 95% (minimedamb_esparg-minhist) ",
float(mrnd-1.96*srnd/sqrt(mcops))],
["I.C. 95% (minimedamb_esparg-minhist) ",
float(mrnd+1.96*srnd/sqrt(mcops))],



["mitjana (maxpsiamb_esparg-minhist)",meqp],
["desvest (maxpsiamb_esparg-minhist)",seqp],
["I.C. 95% (maxpsiamb_esparg-minhist) ",
```

```
float(meqp-1.96*seqp/sqrt(mcops))],
["I.C. 95% (maxpsiamb_esparg-minhist) ",
float(meqp+1.96*seqp/sqrt(mcops))],


["mitjana (miniespamb_medarg-minhist)",mesp],
["desvest (miniespamb_medarg-minhist)",sesp],
["I.C. 95% (miniespamb_medarg-minhist) ",
float(mesp-1.96*sesp/sqrt(mcops))],
["I.C. 95% (miniespamb_medarg-minhist) ",
float(mesp+1.96*sesp/sqrt(mcops))],


["mitjana (minimedamb_medarg-minhist)",mmed],
["desvest (minimedamb_medarg-minhist)",smed],
["I.C. 95% (minimedamb_medarg-minhist) ",
float(mmed-1.96*smed/sqrt(mcops))],
["I.C. 95% (minimedamb_medarg-minhist) ",
float(mmed+1.96*smed/sqrt(mcops))],


["mitjana (maxpsiamb_medarg-minhist)",mpsi],
["desvest (maxpsiamb_medarg-minhist)",spsi],
["I.C. 95% (maxpsiamb_medarg-minhist) ",
float(mpsi-1.96*spsi/sqrt(mcops))],
["I.C. 95% (maxpsiamb_medarg-minhist) ",
float(mpsi+1.96*spsi/sqrt(mcops))]

),

return(finaloutput)
```

```
)$
```

B.3 Rutines utilitzades a la Secció 3.5 Mètodes d'escollir el millor candidat

escollirdecandidats(vt,vx,vc)

Input: vt,vx vectors amb les abscisses i ordenades dels punts per on passa el pont, vc vector satisfent length(vc)=length(tv)-1 i vt[i]<vc[i]<vt[i+1].

Output: el vc[i] que tingui l'esperança de la imatge més petita.

```
escollirdecandidats(vt,vx,vc):=block([i,auxlist],
auxlist:create_list((vx[i+1]-vx[i])/(vt[i+1]-vt[i])
*(vc[i]-vt[i])+vx[i],i,1,length(vt)-1),
return(vc[?position(apply(min,auxlist),auxlist)])
)$
```

c_gpminimpontesp(t0,x0,t1,x1,n,nsimus,gvt,gvx)

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont. Procediment: selecciona l'esperança de l'argument del mínim de cadascun dels intervals en què esta dividida la trajectòria, i després es queda amb el que té l'esperança de la imatge més petita. I així successivament fins a tenir n punts.

Output: si nsimus=0 un vector amb el mínim de les imatges que ha trobat rastrejant en els intervals que tenen la esperança del mínim més petita, i els vectors gvt i gvx ampliats amb els nous valors que hagi utilitzat. si nsimus #0, el vector anterior i l'esperança de l'error respecte el mínim simulat dels punts que ha utilitzat només aquest procés.

```
c_gpminimpontesp(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt,vx,j,l,minvx,meanhist,aux,i,novaimtge,nouarg,novapos,k],
nouarg:medianaarg(t0,x0,t1,x1),
```

```
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then (novaimtge:gvx[?position(nouarg,gvt)])
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],
gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(
for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (

vc:create_list(tayint(vt[l],vx[l],vt[l+1],vx[l+1]),l,1,
length(vt)-1),
nouarg:escollirdecandidats(vt,vx,vc),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/

vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
```

```
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return([apply(min,vx),gvt,gvx])
)$
```

c_gpminimpontmed(t0,x0,t1,x1,n,nsimus,gvt,gvx)

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont. Procediment: selecciona la mediana de l'argument del mínim de cadascun dels intervals en què esta dividida la trajectòria, i després es queda amb el que té l'esperança de la imatge més petita. I així successivament fins a tenir n punts.

Output: si nsimus=0 un vector amb el mínim de les imatges que ha trobat rastrejant en els intervals que tenen la esperança del mínim més petita, i els vectors gvt i gvx ampliats amb els nous valors que hagi utilitzat. si nsimus #0, el vector anterior i l'esperança de l'error respecte el mínim simulat dels punts que ha utilitzat només aquest procés.

```
c_gpminimpontmed(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt,vx,j,vc,minvx,meanhist,aux,i,l,novaimtge,nouarg,novapos,k],
nouarg:medianaarg(t0,x0,t1,x1),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
```

```
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(
for i:1 thru n-1 do(
if filtro(vt)>0.00001 then (
vc:create_list(medianaarg(vt[l],vx[l],vt[l+1],vx[l+1]),l,1,length(vt)-1),
nouarg:escollirdecandidats(vt,vx,vc),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1]
,gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/

vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
```

```
return([apply(min,vx),gvt,gvx])
)$
```

c_gpminimpontbis(t0,x0,t1,x1,n,nsimus,gvt,gvx)

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, gvt: abscisses dels punts per on passa el pont gvx: ordenades dels punts per on passa el pont. Procediment: selecciona el punt mig de cadascun dels intervals en què esta dividida la trajectòria, i després es queda amb el que té l'esperança de la imatge més petita. I així successivament fins a tenir n punts.

Output: si nsimus=0 un vector amb el mínim de les imatges que ha trobat rastrejant en els intervals que tenen la esperança del mínim més petita, i els vectors gvt i gvx ampliats amb els nous valors que hagi utilitzat. si nsimus #0, el vector anterior i l'esperança de l'error respecte el mínim simulat dels punts que ha utilitzat només aquest procés.

```
c_gpminimpontbis(t0,x0,t1,x1,n,nsimus,gvt,gvx):=
block(
[vt,vx,j,vc,minvx,meanhist,aux,i,l,novaimtge,nouarg,novapos,k],
nouarg:float((t1+t0)/2),
vt:[t0,nouarg,t1],
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/
vx:[x0,novaimtge,x1],
if n>1 then
(
for i:1 thru n-1 do(
```

```
if filtro(vt)>0.00001 then (
vc:create_list((vt[l]+vt[l+1])/2,l,1,length(vt)-1),
nouarg:escollirdecandidats(vt,vx,vc),
/*control*/
if member(nouarg,gvt) then novaimtge:gvx[?position(nouarg,gvt)]
else  (k:ondegvt(nouarg,gvt),
novaimtge:Simupuntdelpont(gvt[k],gvx[k],gvt[k+1],gvx[k+1],nouarg),
gvt:sort(append(gvt,[nouarg])),
novapos:?position(nouarg,gvt)-1,
gvx:insertL1enL(gvx,[gvx[novapos],novaimtge],novapos)
),
/*control*/

vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
))
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,abs(minvx-meanhist)]))
else
return([apply(min,vx),gvt,gvx])
)$
```

enllacats3(t0,x0,t1,x1,n,nsimus,printgvx)

Compara els sis mètodes eqd,rnd,eqp vs c_esp, c_med, i c_bis. Input:t0,x0 coordenades del punt inicial del pont, t1,x1: coordenades del punt final del pont, n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, printgvx: variable boolean, si val #0 es printen també les ordenades dels punts que s'han obtingut de la trajectòria.

Output: Si nsimus=0, una matriu amb el valor més baix que ha rastrejat cadascun dels sis mètodes. Si nsimus#0, la matriu anterior a més d'una altra matriu amb els errors de cada mètode respecte del mínim

simulat de la trajectòria amb tots els punts que s'han utilitzat.

```
enllacats3(t0,x0,t1,x1,n,nsimus,printgvx):=block(
[minimimatge,minvx,meanhist,minimx,vt,vx,i,vmins,a,
generalvt,generalvx,fetminiesp,
fetrnd,resdernd,resdeminiesp,fetminimed,
resdeminimed,resdebis,fetbis,

feteqp,resdeeqp,finaloutput:finaloutput],

a:n+1,
vt:float(makelist(t0+(t1-t0)*(i-1)/a,i,1,a+1)),
vx:Vectorpont(vt,x0,x1),
minimimatge:float(apply(min,vx)),
generalvt:vt,generalvx:vx,
/*comença procés random*/
fetrnd:grndEstimadorminpontx(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdernd:fetrnd[1],generalvt:fetrnd[2],generalvx:fetrnd[3],
/*acaba procés random*/

/* comença procés eqp*/

feteqp:geqpEstimadorminpontx(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdeeqp:feteqp[1],generalvt:feteqp[2],generalvx:feteqp[3],
/*acaba procés eqp*/



/*comença procés c_miniesp*/

fetminiesp:c_gpminimpontesp(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdeminiesp:fetminiesp[1],generalvt:fetminiesp[2],
generalvx:fetminiesp[3],
```

```
/*acaba procés c_miniesp*/

/*comença procés c_minimed*/
fetminimed:gpminimpontmed(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdeminimed:fetminimed[1],generalvt:fetminimed[2],
generalvx:fetminimed[3],
/*acaba procés c_minimed*/

/*comença procés bis*/
fetbis:c_gpminimpontbis(t0,x0,t1,x1,n,0,generalvt,generalvx),
resdebis:fetbis[1],generalvt:fetbis[2],generalvx:fetbsi[3],
/*acaba procés bis*/
if printgvx#0 then print(generalvx),
finaloutput:matrix(
["equidistant",minimimatge],
["random",resdernd],
["eqp",resdeeqp],
["minima c_esp",resdeminiesp],
["minima c_med",resdeminimed],["minima c_bis",resdebis],
["min de la trajectoria",float(apply(min,generalvx))],
["dim de gvt",length(generalvt)-2]
),
if nsimus # 0 then
(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(generalvt,
generalvx,nsimus)),
return([finaloutput,[abs(meanhist-minimimatge),
abs(meanhist-resdernd),abs(meanhist-resdeeqp),
abs(meanhist-resdeminiesp),abs(meanhist-resdeminimed),
abs(meanhist-resdebis)]]))
else
return(finaloutput)
```

```
)$
```

st_tots_vs_tots3(t0,x0,t1,x1,n,nsimus,mcops)

Compara eqd,rnd,eqp vs c_esp, c_med, i c_bis. Input: t0,x0 coordenades del punt inicial del pont, t1,x1: coordenades del punt final del pont, n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts, mcops: nombre de vegades que es comparen els mètodes llançant la rutina anterior enllacats3().

Output: una matriu amb la mitjana, desviació típica i intervals de confiança dels errors de cada mètode respecte del mínim simulat de la trajectòria amb tots els punts que s'han utilitzat.

```
st_tots_vs_tots(t0,x0,t1,x1,n,nsimus,mcops):=block(
[i,res,aux,finaloutput,mit1,desvest1,mit2,desvest2,B,
meqd,seqd,mrnd,srnd,meqp,seqp,mesp,sesp,mmed,smed,mbis,sbis],
reseqd:makelist(0,i,1,mcops),resrnd:makelist(0,i,1,mcops),

B:apply(matrix,create_list(enllacats3(t0,x0,t1,x1,n,nsimus,0)[2],i,1,mcops)),
meqd:mean(col(B,1))[1],
seqd:std(col(B,1))[1],
mrnd:mean(col(B,2))[1],
srnd:std(col(B,2))[1],
meqp:mean(col(B,3))[1],
seqp:std(col(B,3))[1],
mesp:mean(col(B,4))[1],
sesp:std(col(B,4))[1],
mmed:mean(col(B,5))[1],
smed:std(col(B,5))[1],
mbis:mean(col(B,6))[1],
sbis:std(col(B,6))[1],

finaloutput:matrix(
["mitjana (eqd-minhist)",meqd],
["desvest (eqd-minhist)",seqd],
["I.C. 95% (eqd-minhist) ",float(meqd-1.96*seqd/sqrt(mcops))],
```

```
["I.C. 95% (eqd-minhist) ",float(meqd+1.96*seqd/sqrt(mcops))],
["mitjana (rnd-minhist)",mrnd],
["desvest (rnd-minhist)",srnd],
["I.C. 95% (rnd-minhist) ",float(mrnd-1.96*srnd/sqrt(mcops))],
["I.C. 95% (rnd-minhist) ",float(mrnd+1.96*srnd/sqrt(mcops))],
["mitjana (eqp-minhist)",meqp],
["desvest (eqp-minhist)",seqp],
["I.C. 95% (eqp-minhist) ",float(meqp-1.96*seqp/sqrt(mcops))],
["I.C. 95% (eqp-minhist) ",float(meqp+1.96*seqp/sqrt(mcops))],
["mitjana (min X_esp-minhist)",mesp],
["desvest (min X_esp-minhist)",sesp],
["I.C. 95% (min X_esp-minhist) ",float(mesp-1.96*sesp/sqrt(mcops))],
["I.C. 95% (min X_esp-minhist) ",float(mesp+1.96*sesp/sqrt(mcops))],
["mitjana (min X_med-minhist)",mmed],
["desvest (min X_med-minhist)",smed],
["I.C. 95% (min X_med-minhist) ",float(mmed-1.96*smed/sqrt(mcops))],
["I.C. 95% (min X_med-minhist) ",float(mmed+1.96*smed/sqrt(mcops))],
["mitjana (min X_bis-minhist)",mpsi],
["desvest (min X_bis-minhist)",spsi],
["I.C. 95% (min X_bis-minhist) ",float(mbis-1.96*sbis/sqrt(mcops))],
["I.C. 95% (min X_bis-minhist) ",float(mbis+1.96*sbis/sqrt(mcops))]
),
return(finaloutput)

)$
```

puntprop(t0,x0,t1,x1,p)

Input:t1,x1 punt inicial del pont, t2,x2 punt final del pont, p proporció que pertany a (0,1).

Output: el punt t que es troba a proporció p del t0 o t1 que tingui la imatge més petita.

```
puntprop(t0,x0,t1,x1,p):=block(

if x0< x1 then return(bfloat((t1-t0)*p+t0))
```

```
else

return (bfloat(-(t1-t0)*p+t1))

)$
```

positionbfloat(vt,valor)

Input: vt vector, valor element de vt.

Output: la posició que ocupa valor a vt. Aquesta funció s'ha definit perquè la ?position de wxmaxima no funciona per a bfloats.

```
positionbfloat(vt,valor):=block([i,pos,col],
col:length(vt),
for i:1 thru col do (if vt[i]=valor
then  (pos:i,i:col)),
return(pos)
)$
```

proporcio(t0,x0,t1,x1,p,n,nsimus)

Input:t0,x0 coordenades del punt inicial del pont t1,x1: coordenades del punt final del pont n: nombre de punts que es volen avaluar nsimus: nombre de simulacions de la imatge del mínim del pont condicionat als n punts. Output:un vector amb el mínim valor que ha trobat, el mínim simulat i la diferència entre aquests dos.

```
proporcio(t0,x0,t1,x1,p,n,nsimus):=
block(
[vt,vx,j,vc,minvx,meanhist,aux:aux,i,l,novaimtge,nouarg,novapos,k],
nouarg:puntprop(t0,x0,t1,x1,p),
vt:[t0,nouarg,bfloat(t1)],
novaimtge:Simupuntdelpont(t0,x0,t1,x1,nouarg),

vx:[x0,novaimtge,x1],
if n>1 then
```

```
(
for i:1 thru n-1 do(

vc:create_list(puntprop(vt[l],vx[l],vt[l+1],vx[l+1],p),l,1,length(vt)-1),
nouarg:escollirdecandidats(vt,vx,vc),


k:ondegvt(nouarg,vt),
novaimtge:Simupuntdelpont(vt[k],vx[k],vt[k+1],vx[k+1],nouarg)
,


vt:sort(append(vt,[nouarg])),
novapos:?position(nouarg,vt)-1,
vx:insertL1enL(vx,[vx[novapos],novaimtge],novapos)
)
),

(
minvx:apply(min,vx),meanhist:mean(Histogramimatge(vt,vx,nsimus)),
return([minvx,meanhist,float(abs(minvx-meanhist))])
)
)$
```

B.4 Rutina utilitzada a la Secció 3.6 Funcions de pèrdues amb X_t i $m(X)$

```
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_integration.h>
#define MIN(a,b) (((a)<(b))?(a):(b))

double H (double, void * );
```

```
double F (double, void * );
double G_0t (double, void * );
double G_t1 (double, void * );

double integrand (double x, double y){
//return (MIN(x,0)-y); // E[(min(B_t,B_0)-m(B))]
return (MIN(x,0)-y)*(MIN(x,0)-y); // E[(min(B_t,B_0)-m(B))^2]
//return (x-y)*(x-y); // E[(B_t-m(B))^2]
// return 1; // integra la pròpia densitat conjunta
}

double G_0t (double s, void * params){  /* s \in [0,t] */
double t, x1, y, x;
double g;

struct paramsZ{ double t; double x1; double y; double x;}
* alphaZ = (struct paramsZ *) params;
t = (*alphaZ).t;
x1 = (*alphaZ).x1;
y = (*alphaZ).y;
x = (*alphaZ).x;

g = integrand(x,y)*(x-y)*(-y)*exp(-(x-y)*(x-y)/(2*t-2*s)-y*y/(2*s))
/ pow((t-s)*s,3./2);
return g;
}

double G_t1 (double s, void * params) {   /* s \in [t,1] */
double t, x1, y, x;
double g;

struct paramsZ{ double t; double x1; double y; double x;}
```

```
* alphaZ = (struct paramsZ *) params;
t = (*alphaZ).t;
x1 = (*alphaZ).x1;
y = (*alphaZ).y;
x = (*alphaZ).x;

g = integrand(x,y)*(x-y)*(x1-y)*exp(-(x1-y)*(x1-y)/(2-2*s)
-(x-y)*(x-y)/(2*s-2*t)) / pow((1-s)*(s-t),3./2);
return g;
}

double F (double x, void * params) {
double f1, f2, f;
double result, error;
double t, x1, y;
gsl_integration_workspace * w = gsl_integration_workspace_alloc (1000);

struct paramsF{ double t; double x1; double y;}
* alphaF = ( struct paramsF *) params;
struct { double t; double x1; double y; double x;} alphaG;
alphaG.t = (*alphaF).t;
alphaG.x1 = (*alphaF).x1;
alphaG.y = (*alphaF).y;
alphaG.x = x;
t = alphaG.t;
x1 = alphaG.x1;
y = alphaG.y;
//printf ("   x = %g;\n", x);

gsl_function func;
func.function = &G_0t;
func.params = &alphaG;
```

```
//f1: integral for s from 0 to t:
gsl_integration_qag (&func, 0, t, 1e-3, 1e-3, 1000, 2,
w, &result, &error);
//printf ("   Integral de G_0t(s) en [0,t] = %g;", result);

f1 = result /sqrt(1-t)*(exp(-(x-x1)*(x-x1)/(2*(1-t)))
-exp(-(x+x1-2*y)*(x+x1-2*y)/(2*(1-t))));
func.function = &G_t1;
//f2: integral for s from t to 1:
gsl_integration_qag (&func, t, 1, 1e-3, 1e-3, 1000, 2,
w, &result, &error);
//printf ("  Integral de G_t1(s) en [t,1] = %g \n", result);
f2 = result /sqrt(t)*(exp(-x*x/(2*t))-exp(-(x-2*y)*(x-2*y)/(2*t)));

f = f1 + f2;
gsl_integration_workspace_free (w);
return f;
}

double H (double y, void * params) {
gsl_integration_workspace * w = gsl_integration_workspace_alloc (1000);
double result, error;

struct paramsH{ double t; double x1;} * alphaH = ( struct paramsH *) params;
struct { double t; double x1; double y;} alphaF;
alphaF.t = (*alphaH).t;
alphaF.x1 = (*alphaH).x1;
alphaF.y = y;
//printf ("y = %g;\n", y);
gsl_function func;
func.function = &F;
```

```
func.params = &alphaF;
gsl_integration_qagiu (&func, y+.000, 1e-3, 1e-3, 1000, //2,
w, &result, &error);

//printf ("   Integral de F(x) en [y,infty]= %g \n", result);
gsl_integration_workspace_free (w);
return result;
}

int
main (void)
{
gsl_integration_workspace * w
= gsl_integration_workspace_alloc (1000);

double result, error;
float t;
struct { double t; double x1;} alpha;
for (t = 0.05; t<0.96; t=t+0.05){
alpha.t = t;
alpha.x1 = 1;

gsl_function func;
func.function = &H;
func.params = &alpha;
gsl_integration_qagil (&func, 0-.000, 1e-3, 1e-3, 1000, //2,
w, &result, &error);

result = 1/M_PI * exp(alpha.x1*alpha.x1/2) * result;
printf ("\nRESULTAT:\nJ( %g, %g)= Integral de H(y) en [-infty,0] = %17.15f\n",
 alpha.t, alpha.x1, result);
printf ("ERROR ESTIMAT: %g", error);
```

```
}
gsl_integration_workspace_free (w);
return 0;
}
```