Communications in Statistics - Simulation and Computation
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/lssp20

# Brownian Bridge and Other Path-dependent Gaussian Processes Vectorial Simulation

J. Beleza Sousa[a][b], M. L. Esquível[b] & R. M. Gaspar[c]

[a] M2A/ADEETC, Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, Lisboa, Portugal

[b] CMA/DM, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Caparica, Portugal

[c] CEMAPRE, ISEG, Universidade de Lisboa, Lisboa, Portugal

Accepted author version posted online: 23 Jul 2014.

To cite this article: J. Beleza Sousa, M. L. Esquível & R. M. Gaspar (2015) Brownian Bridge and Other Path-dependent Gaussian Processes Vectorial Simulation, Communications in Statistics - Simulation and Computation, 44:10, 2608-2621, DOI: 10.1080/03610918.2014.901352

To link to this article: http://dx.doi.org/10.1080/03610918.2014.901352

*Communications in Statistics—Simulation and Computation®*, 44: 2608–2621, 2015
Copyright © Taylor & Francis Group, LLC
ISSN: 0361-0918 print / 1532-4141 online
DOI: 10.1080/03610918.2014.901352

# Brownian Bridge and Other Path-dependent Gaussian Processes Vectorial Simulation

J. BELEZA SOUSA,[1,2] M. L. ESQUÍVEL,[2] AND R. M. GASPAR[3]

[1]M2A/ADEETC, Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, Lisboa, Portugal
[2]CMA/DM, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Caparica, Portugal
[3]CEMAPRE, ISEG, Universidade de Lisboa, Lisboa, Portugal

*The iterative simulation of the Brownian bridge is well known. In this article, we present a vectorial simulation alternative based on Gaussian processes for machine learning regression that is suitable for interpreted programming languages implementations. We extend the vectorial simulation of path-dependent trajectories to other Gaussian processes, namely, sequences of Brownian bridges, geometric Brownian motion, fractional Brownian motion, and Ornstein–Ulenbeck mean reversion process.*

**Keywords** Brownian bridge; Gaussian processes for machine learning regression; Path-dependent Gaussian processes; Vectorial simulation.

**Mathematics Subject Classification** Primary 60G15; 68U20; Secondary 65C20; 62-04; 90-04.

## 1. Introduction

Interpreted programming languages like Sage, Octave, Mathematica, and Matlab are currently important frameworks in research and development.

In these programming languages, it is crucial to use vectorial algorithms instead of iterative ones in order to achieve the execution speeds of compiled languages. This is because vectorial operations are typically supported by built-in functions which are implemented by optimized machine code.

The iterative simulation of the Brownian bridge is well known (Glasserman, 2003; Qua, 2012). In this article, we present a vectorial simulation alternative based on Gaussian processes for machine learning regression that is suitable for interpreted programming languages implementations.

We extend the vectorial simulation of path-dependent trajectories to other Gaussian processes, namely, sequences of Brownian bridges, geometric Brownian motion, fractional Brownian motion and Ornstein–Ulenbeck mean reversion process, by developing a Gaussian path-dependent trajectories simulation vectorial framework.

Received September 30, 2012; Accepted February 24, 2014

Address correspondence to Prof. J. Beleza Sousa, ISEL Instituto Superior de Engenharia de Lisboa, M2A/ADEETC, Rua Conselheiro Emdio Navarro, 1, Lisboa, 1959-007 Portugal; E-mail: jsousa@deetc.isel.ipl.pt

We illustrate the flexibility of the path-dependent vectorial simulation procedure by creating a two-dimensional Wiener process representation of a Norbert Wiener photograph.

Simulation of Gaussian processes immediately spread with computers availability and became an important tool in many science areas, such as mathematics (Kloeden and Platen, 1992), finance (Glasserman, 2003), engineering (Kasdin, 1995), hydrology (Mandelbrot, 1971), and geology (Alabert, 1987), among many others.

In particular, the simulation of Brownian bridges, geometric Brownian motion, fractional Brownian motion, and Ornstein–Ulenbeck mean reversion process, play an important role in Monte Carlo methods (Moskowitz and Caflisch, 1996), securities pricing (Broadie and Glasserman, 1997), communication networks (Paxson, 1997) and particles motion (Gillespie, 1996), respectively.

A wide range of path-dependent Gaussian trajectories simulation methods exist, spanning from the earlier, based on sampling the unconditional distribution (Hoffman and Ribak, 1991), Cholesky factorization (Davis, 1987) or FFT (Dietrich and Newsam, 1996), to the more recent efforts of implementing the old methods in the emerging parallel architectures (Garland et al., 2008; Volkov and Demmel, 2008).

In this article, we use the Cholesky method. Despite having known limitations (Jean-François, 2000) it allows the extension to the path-dependent case and we show that it is relevant concerning the execution speed of interpreted programming languages implementations.

## 2. Brownian Bridge Iterative Simulation

A Brownian bridge is a standard Brownian motion $W$ conditioned to $W(1) = 0$. The Brownian bridge condition $W(1) = 0$ can be generalized to other time instants greater than zero and to other values besides zero.

The standard Brownian motion $W$, defined in $\mathbb{R}_0^+$, is also called a Wiener process (Björk, 2004) and has the following properties:

1. $W(0) = 0$;
2. $W$ has independent increments, i.e. if $r < s \leq t < u$ then $W(u) - W(t)$ and $W(s) - W(r)$ are independent random variables;
3. For $s < t$ the random variable $W(t) - W(s)$ has the Gaussian distribution $\mathcal{N}(0, \sqrt{t-s})$;
4. $W$ has almost surely continuous trajectories.

In addition, the Wiener process is a Gaussian process with mean function $m(t)$ and covariance function $\text{cov}(s, t)$:

$$m(t) = 0, \tag{1}$$

$$\text{cov}(s, t) = \min(s, t). \tag{2}$$

In order to illustrate the iterative simulation of a Brownian bridge trajectory $B$, consider that at some step we have $0 < u < s < t < 1$, $B(u) = \alpha$, $B(t) = \beta$ and we want to simulate the value $B(s)$ (Glasserman, 2003). Given the Wiener process properties, the random vector

$[B(u)B(s)B(t)]^T$ is Gaussian with mean vector and covariance matrix:

$$\begin{bmatrix} B(u) \\ B(s) \\ B(t) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} u & u & u \\ u & s & s \\ u & s & t \end{bmatrix}\right) \tag{3}$$

Therefore, the conditional distribution $B(s)|B(u), B(t)$ is given by

$$B(s)|B(u), B(t) \sim \mathcal{N}\left(\frac{(t-s)\alpha + (s-u)\beta}{t-u}, \frac{(s-u)(t-s)}{t-u}\right) \tag{4}$$

Thus, the value $B(s)$ is simulated by

$$B(s) = \frac{(t-s)\alpha + (s-u)\beta}{t-u} + \sqrt{\frac{(s-u)(t-s)}{t-u}}Z, \tag{5}$$

where $Z \sim \mathcal{N}(0, 1)$ is an increment independent of all $Z$ values previously used in the simulation.

Finally, the iterative simulation of a Brownian bridge trajectory consists of starting with $u = 0$, $t = 1$, $B(0) = 0$, $B(1) = 0$ and iteratively filling a trajectory sample at time $s$ (between $u$ and $t$) with Eq. (5), then moving one of the end points to the simulated sample and repeating the process until all trajectory samples are filled.

Figure 1 shows a sequence of 500 simulated independent Gaussian $\mathcal{N}(0, 1)$ increments (white noise), and the corresponding simulated Wiener process and Brownian bridge trajectories (sampled uniformly 500 times between zero and one). The Brownian bridge trajectory was simulated by the iterative procedure above.

## 3. Gaussian Processes for Machine Learning

The goal of Gaussian processes for machine learning regression is to find the nonlinear unknown mapping $y = f(\mathbf{x})$, from data $(\mathbf{X}, \mathbf{y})$, using Gaussian distributions over functions (Rasmussen and Williams, 2005):

$$\mathcal{GP} \sim \mathcal{N}(m(\mathbf{x}), cov(\mathbf{x_1}, \mathbf{x_2})). \tag{6}$$

The Gaussian process defined by $m(\mathbf{x})$ and $cov(\mathbf{x_1}, \mathbf{x_2})$, in Eq. (6), is the prior process.

The pair $(\mathbf{X}, \mathbf{y})$ is the training set. The matrix $\mathbf{X}$ collects a set of $n$ vectors $\mathbf{x}$, where the value $y = f(\mathbf{x})$ was observed. The corresponding $y$ values are collected in vector $\mathbf{y}$.

The set of vectors $\mathbf{x}^\star$ where the values $y^\star = f(\mathbf{x}^\star)$ were not observed, is collected in matrix $\mathbf{X}^\star$. The matrix $\mathbf{X}^\star$ is the test set.

The regression function is the mean function of the process defined by all the trajectories of the prior process that passes through the training set. The regression confidence is the corresponding covariance function. The regression mean and the regression confidence define the posterior process on data.

As presented in the previous section, the Wiener process is a scalar Gaussian process

$$W \sim \mathcal{N}(m(t), cov(s, t)) \tag{7}$$

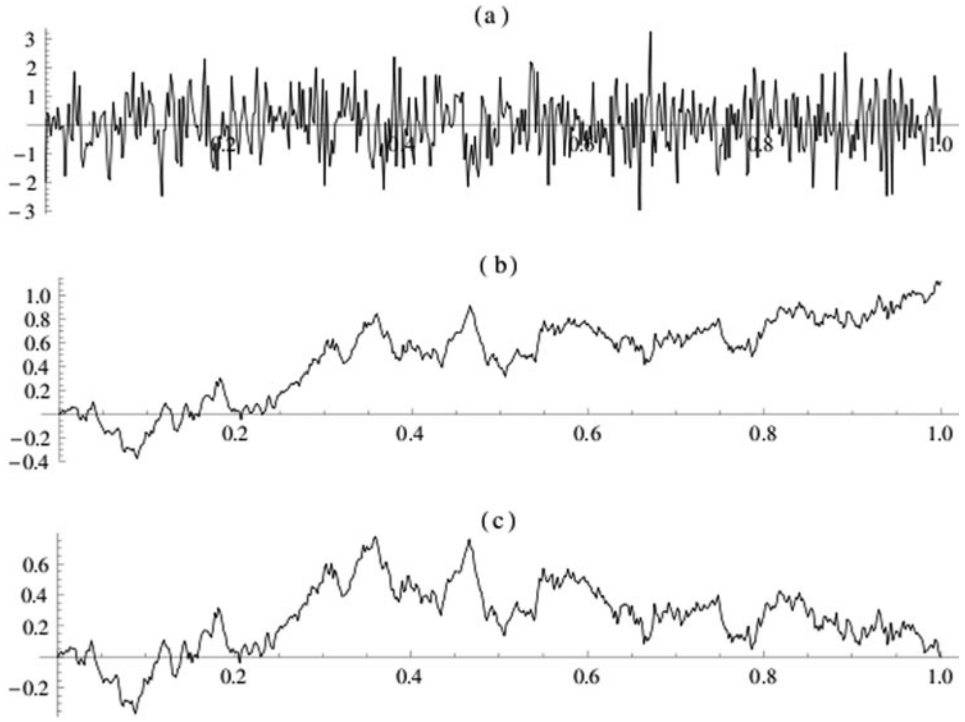

**Figure 1.** Simulated trajectories of: (a) white noise; (b) the corresponding Wiener process; (c) the corresponding Brownian bridge.

where $m(t)$ is given by Eq. (1) and $cov(s, t)$ by Eq. (2).

In this case, the mapping $f$ is the scalar mapping $y = f(t)$, where $y$ is the value of $W$ at time $t$. This reduces the training set to the pair of vectors ($\mathbf{t}$, $\mathbf{y}$), and the test set to vector $\mathbf{t}^\star$.

Since the process is Gaussian (Rasmussen and Williams, 2005)

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^\star \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m} \\ \mathbf{m}^\star \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_\star \\ \mathbf{K}_\star^T & \mathbf{K}_{\star\star} \end{bmatrix}\right) \tag{8}$$

and

$$p(\mathbf{y}^\star|\mathbf{t}^\star, \mathbf{t}, \mathbf{y}) \sim \mathcal{N}\left(\mathbf{m}^\star + \mathbf{K}_\star^T\mathbf{K}^{-1}(\mathbf{y} - \mathbf{m}), \mathbf{K}_{\star\star} - \mathbf{K}_\star^T\mathbf{K}^{-1}\mathbf{K}_\star\right) \tag{9}$$

where $\mathbf{m}$ and $\mathbf{m}^\star$ are mean vectors of training and test sets, $\mathbf{K}$ is the training set covariance matrix, $\mathbf{K}_\star$ the training-test covariance matrix and $\mathbf{K}_{\star\star}$ the test set covariance matrix.

The conditional distribution

$$p(\mathbf{y}^\star|\mathbf{t}^\star, \mathbf{t}, \mathbf{y}) \tag{10}$$

corresponds to the posterior process on the data

$$\mathcal{GP}_\mathcal{D} \sim \mathcal{N}(m_\mathcal{D}(t), cov_\mathcal{D}(s, t)) \tag{11}$$

where

$$m_{\mathcal{D}}(t) = m(t) + \mathbf{K}_{\mathbf{t},t}^T \mathbf{K}^{-1}(\mathbf{y} - \mathbf{m}) \tag{12}$$

and

$$cov_{\mathcal{D}}(s, t) = cov(s, t) - \mathbf{K}_{\mathbf{t},s}^T \mathbf{K}^{-1} \mathbf{K}_{\mathbf{t},t} \tag{13}$$

where $\mathbf{K}_{\mathbf{t},t}$ is a covariance vector between every training instant and $t$.

Eq. (12) is the regression function, while Eq. (13) is the regression confidence. Eqs. (12) and (13) are the central equations of Gaussian processes for machine learning regression.

Figure 2 shows an example of Gaussian processes for machine learning regression using the Wiener process as the prior process and the set of 500 time instants, uniformly distributed between zero and one, as the test set.

## 4. Browning Bridge Vectorial Simulation

The vectorial simulation of Brownian bridge trajectories is as achieved by joining Sections 2 and 3 .

Considering:

1. the Wiener process $W$ with mean and covariance functions given by Eqs. (1) and (2);
2. the training set with the single pair $(t, y) = (1, 0)$ corresponding to the Brownian bridge condition $W(1) = 0$;
3. the test vector $\mathbf{t}^\star = [t_1, t_2, \ldots, t_n]^T$ where $t_1, t_2, \ldots, t_n$ are the time instants where to sample the Brownian bridge trajectory.

The Brownian bridge process $B$ is Gaussian and the random vector $\mathbf{B} = B(\mathbf{t}^\star)$ is also Gaussian

$$\mathbf{B} \sim \mathcal{N}(\mathbf{m}_{\mathcal{D}}, \mathbf{cov}_{\mathcal{D}}) \tag{14}$$

where $\mathbf{m}_{\mathcal{D}}$ is $\mathbf{B}$ mean vector and $\mathbf{cov}_{\mathcal{D}}$ is $\mathbf{B}$ covariance matrix. The element $i$ of vector $\mathbf{m}_{\mathcal{D}}$ is given by

$$\mathbf{m}_{\mathcal{D}i} = m_{\mathcal{D}}(t_i) \tag{15}$$

and the element $i, j$ of matrix $\mathbf{cov}_{\mathcal{D}}$ is given by

$$\mathbf{cov}_{\mathcal{D}i,j} = \mathrm{cov}_{\mathcal{D}}(t_i, t_j). \tag{16}$$

Functions $m_{\mathcal{D}}(t_i)$ and $\mathrm{cov}_{\mathcal{D}}(t_i, t_j)$ are those of Eqs. (1) and (2).

Therefore, a Brownian bridge trajectory can be simulated as any other Gaussian vector (Glasserman, 2003), using:

$$\mathbf{B} = \mathbf{m}_{\mathcal{D}} + \mathbf{C}\mathbf{Z} \tag{17}$$

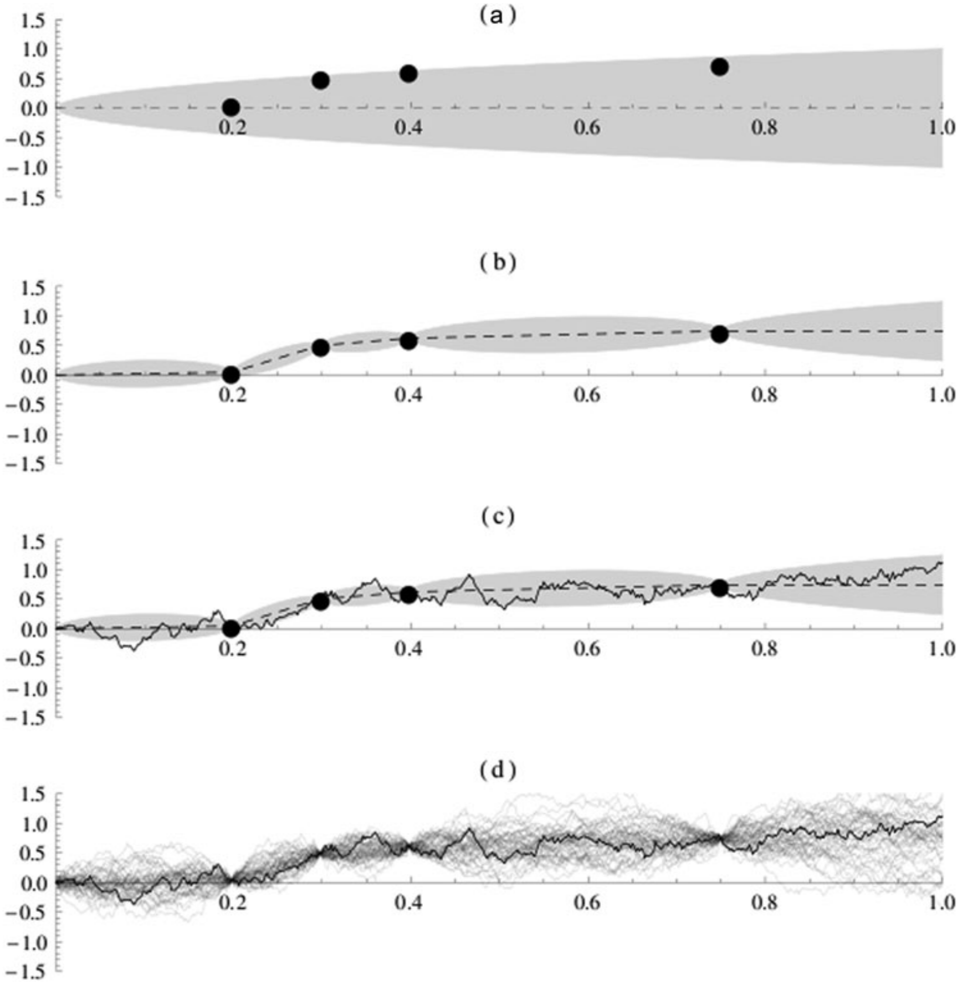

**Figure 2.** Gaussian processes for machine learning regression with the Wiener process as prior: (a) prior process mean (dashed), prior process two standard deviations band (gray) and the training set (circles); (b) regression function (dashed) and two standard deviations regression confidence band (gray); (c) training set simulated trajectory; (d) simulated Wiener process trajectories passing through the training set.

where $\mathbf{C}$ is the Cholesky decomposition of $\mathbf{cov}_{\mathcal{D}}$ and $\mathbf{Z}$ is a sample of the Gaussian random vector $\mathcal{N}(0, I)$.

Figure 3 shows some Brownian bridge trajectories simulated with the vectorial Eq. (17). The solid black one was simulated with the Gaussian increments of Fig. 1. Since Eq. (17) is just a vectorial alternative to the iterative procedure of Section 2, the solid black trajectory is, as it would be expected, equal to the Brownian bridge trajectory of Fig. 1.

## 5. Execution Time Comparison

In order to compare the execution times of the iterative Brownian bridge simulation procedure of Section 2 and the vectorial procedure of Section 4 under an interpreted language

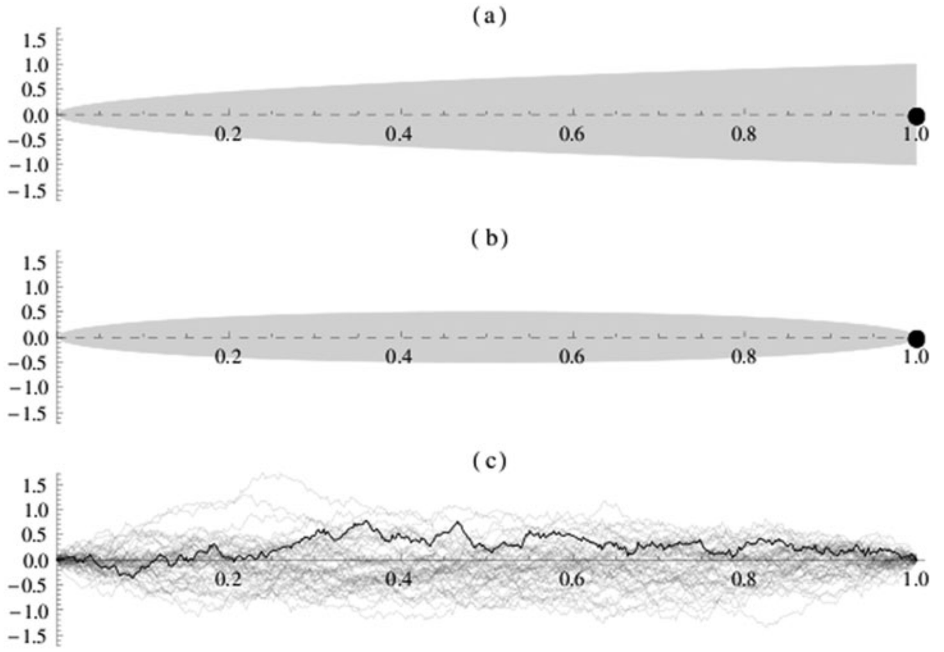

**Figure 3.** Brownian bridge trajectories simulated with the vectorial Eq. (17): (a) prior process mean (dashed), prior process two standard deviations band (gray) and the training set (circle); (b) regression function (dashed) and two standard deviation regression confidence band (gray); (c) Brownian bridge simulated trajectories.

framework, we implemented both using the Mathematica 8 language (Wolfram Research, 2011) and tested the two alternatives on two different stages:

**Stage 1** Inspired by the order of magnitude of typical setups found in financial markets, such as 250 daily prices per year, and stock indices with up to 500 stocks, we defined the reference task of generating 1000 Brownian bridges sampled uniformly 1000 times. This would correspond to simulate four years, of daily prices, of an index as bigger as twice the S&P500.

**Stage 2** In order to evaluate the performance sensitivity to the task specification, we varied both the number of samples per trajectory and the number of trajectories.

Table 1 describes the execution system, the reference task, and the execution times obtained for both alternatives on Stage 1.

As it would be expected, the Brownian bridge vectorial simulation with the Mathematica 8 language is faster than the iterative alternative. In the particular case of the reference task, approximately 10 times faster. As mentioned before, this is because vectorial operations are supported by built-in functions which are implemented by optimized machine code.

Tables 2 and 3 describe the execution times obtained for both alternatives on Stage 2.

Table 2 shows, in a clear way, the main limitation of the vectorial simulation procedure, which is memory space. Simulation of all trajectories at a time, using Eq. (17), requires memory space for the number-of-samples-by-number-of-samples square matrix **C** and for the number-of-samples-by-number-of-trajectories rectangular matrix **Z**. As the number of

**Table 1**
Iterative and vectorial execution times comparison for the reference task

| | |
|---|---|
| (a) System | |
| CPU | Intel Core2 CPU 6300 1.86GHz |
| Memory | 4GB |
| OS | Linux × 86 (32bit) |
| Language | Mathematica 8.0.4.0 |
| (b) Task | |
| Simulation | Brownian bridge trajectories |
| Number of trajectories | 1000 |
| Number of samples per trajectory | 1000 (uniformly) |
| (c) Execution Times (in seconds) | |
| Iterative | 32.31 |
| Vectorial | 3.49 |

trajectories and the number of samples per trajectory grow, the memory space becomes a severe limitation of the vectorial simulation procedure.

Table 3 shows that, for a small number of trajectories (up to 100) with a reasonable number of samples (1000), the vectorial simulation procedure is useless, due to the overhead execution time for computing matrix **C**.

## 6. Extensions

It is clear by the Brownian bridge vectorial simulation construction that the simulation procedure can be naturally extended in the following three ways:

1. considering a condition different from $W(1) = 0$ (either in the time instant and its value);
2. considering more than one condition (sequences of bridges);
3. considering other Gaussian processes besides the Wiener process (considering mean and covariance functions different from the Wiener process ones).

**Table 2**
1000 trajectories execution time sensitivity to the number of samples

| | Execution time (s) | | |
|---|---|---|---|
| Number of samples | Iterative | Vectorial | Improvement (times faster) |
| 10 | 0.26 | 0.003 | 86.67 |
| 100 | 3.39 | 0.04 | 84.75 |
| 1000 | 32.42 | 3.93 | 8.25 |
| 10000 | 305.43 | Out of Memory | — |
| 100000 | 2989.15 | Out of Memory | — |

**Table 3**
1000 samples per trajectory execution time sensitivity to the number of trajectories

| | Execution time (s) | | |
|---|---|---|---|
| Number of trajectories | Iterative | Vectorial | Improvement (times faster) |
| 10 | 0.29 | 3.32 | 11.45 (slower) |
| 100 | 3.40 | 3.36 | 1.01 |
| 1000 | 32.24 | 3.91 | 8.24 |
| 10000 | 300.28 | 8.48 | 35.41 |
| 100000 | 2981.13 | Out of Memory | — |

The first two ways were already illustrated by Fig. 2(d), where there were a total of four conditions, different from the Brownian bridge condition.

Regarding the third way Figs. 4–6 illustrate the same example of Fig. 2, but now for geometric Brownian motion, fractional Brownian motion and Ornstein–Ulenbeck mean reversion process. We chose these processes for their importance in modeling stock prices

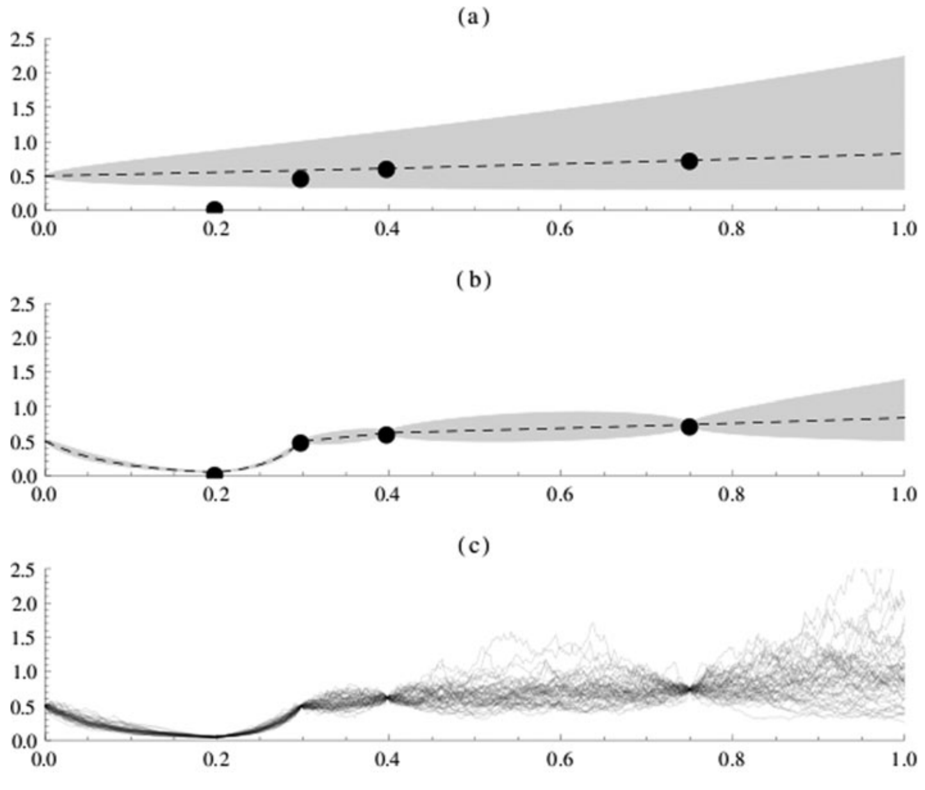

**Figure 4.** Gaussian processes for machine learning regression with geometric Brownian motion as prior: (a) prior process mean (dashed), prior process two standard deviations band (gray) and the training set (circle); (b) regression function (dashed) and two standard deviation regression confidence band (gray); (c) path-dependent simulated trajectories (passing through the training set).

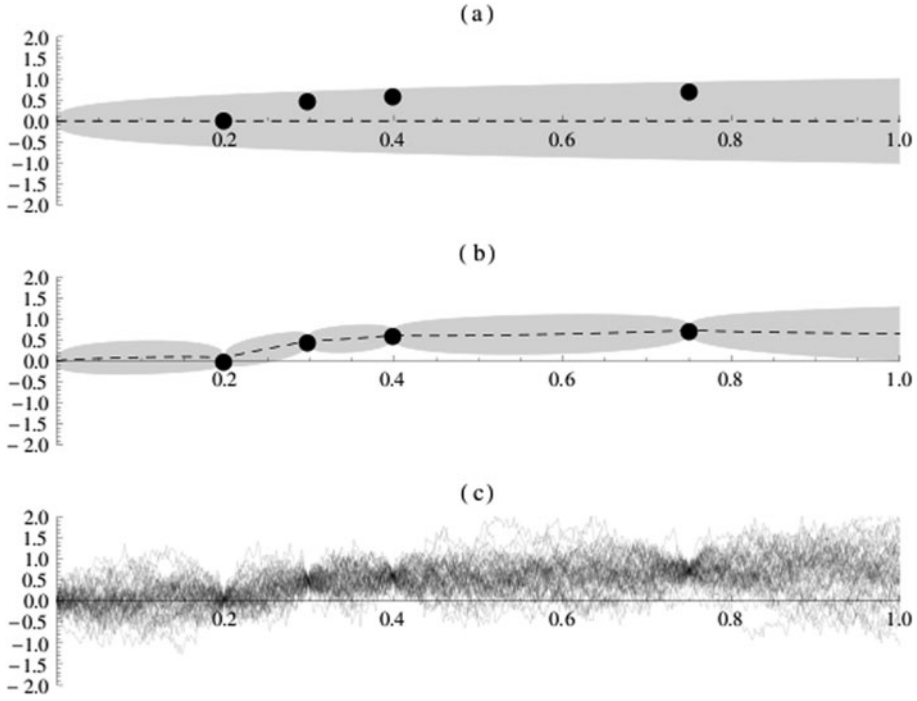

**Figure 5.** Gaussian processes for machine learning regression with the Ornstein–Ulenbeck mean reversion process as prior: (a) prior process mean (dashed), prior process two standard deviations band (gray) and the training set (circles); (b) regression function (dashed) and two standard deviation regression confidence band (gray); (c) path-dependent simulated trajectories (passing through the training set).

and interest rates. The simulation procedure is the same as the one in the example of Fig. 2, except that the appropriate mean and covariance functions are used.

In the geometric Brownian motion case, the simulation was done for the underlying log normal process, which is a Gaussian process with mean and covariance functions given by

$$m(t) = \left(\mu - \frac{\sigma^2}{2}\right) t \tag{18}$$

and

$$\mathrm{cov}(s, t) = \sigma^2 \min(s, t) \tag{19}$$

where $\mu$ is the drift and $\sigma$ is the volatility. The geometric Brownian motion trajectories were obtained by taking the exponential of the log normal ones and multiplying by the process initial value $x_0$. The values used in the simulation examples of Fig. 4 were: $x_0 = 0.5$; $\mu = 1.0$ and $\sigma = 1.0$.

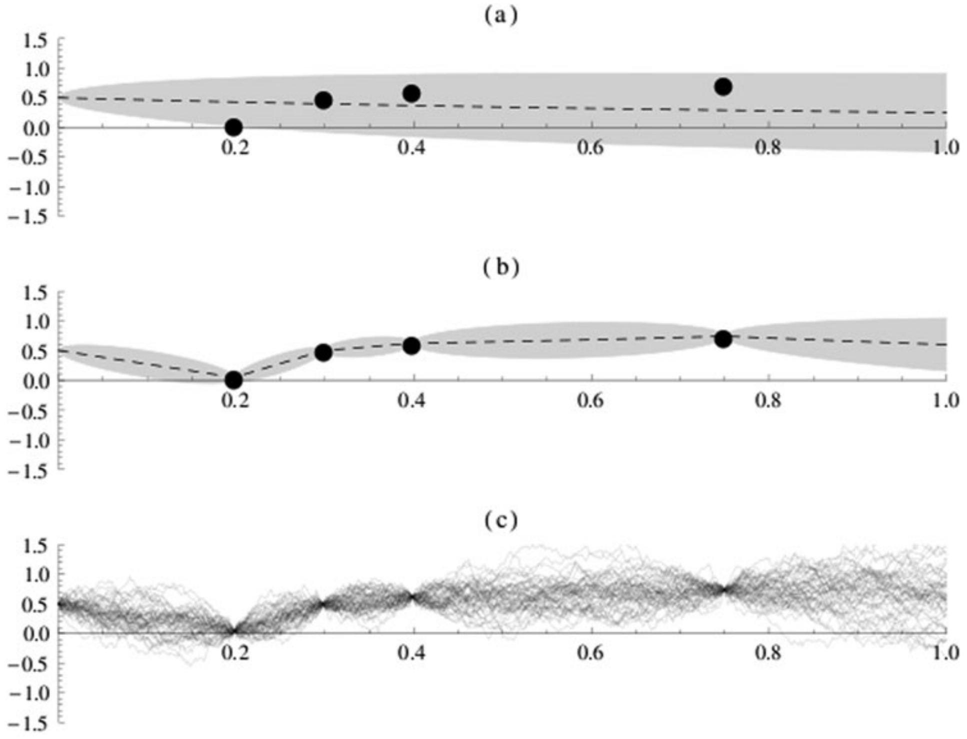

**Figure 6.** Gaussian processes for machine learning regression with fractional Brownian motion as prior: (a) prior process mean (dashed), prior process two standard deviations band (gray) and the training set (circles); (b) regression function (dashed) and two standard deviation regression confidence band (gray); (c) path-dependent simulated trajectories (passing through the training set).

In the fractional Brownian motion case, the mean and covariance functions are given by

$$m(t) = 0 \tag{20}$$

and

$$\operatorname{cov}(s,t) = \frac{1}{2}\left(|s|^{2H} + |t|^{2H} - |s-t|^{2H}\right) \tag{21}$$

where $H$ is the Hurst index. The value used in the simulation examples of Fig. 5 was: $H = 0.3$.

In the Ornstein–Ulenbeck mean reversion process case, the mean and covariance functions are given by

$$m(t) = x_0 e^{-kt} + \theta(1 - e^{-kt}) \tag{22}$$

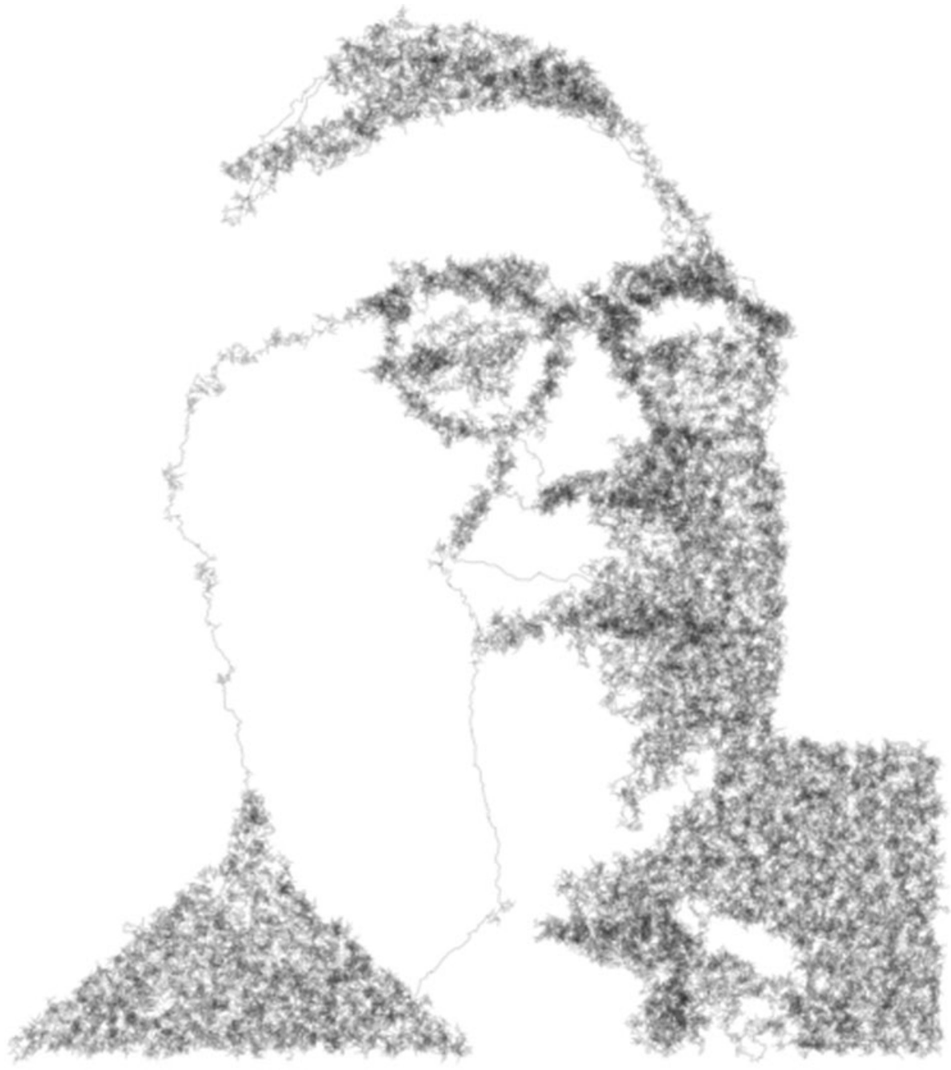

**Figure 7.** 2D Wiener process single path representation of a Norbert Wiener photo.

and

$$\mathrm{cov}(s,t) = \frac{\sigma^2}{2k} e^{-k(s+t)} \left(e^{2k\min(s,t)} - 1\right) \tag{23}$$

where $x_0$ is the process initial value, $k$ is the mean reversion velocity, $\theta$ is the mean reversion level, and $\sigma$ is the volatility. The values used in the simulation examples of Fig. 6 were: $x_0 = 0.5$; $k = 2.0$; $\theta = 0.1$ and $\sigma = 0.5$.

## 7. Illustration

In this section, we illustrate the great flexibility of the path dependent vectorial simulation procedure by constructing a 2D Wiener process single path representation of a Norbert Wiener photo. The steps taken to construct the representation are the following:

1. Choose a white background photo.
2. Obtain a binarized with dithering version of the photo.
3. Obtain a possible sequence of nearest black pixels: starting at a random black pixel find its nearest black pixel neighbor; repeat the procedure from the found neighbor, not considering the pixels already processed, until reaching the last unprocessed pixel.
4. Consider the black pixels coordinates, $x$ and $y$, as the conditioning constraints.
5. For the sequence of the $x$ coordinate constraints, simulate a Wiener process trajectory by sampling uniformly 50 times each successive pair of constraints.
6. Repeat the previous step for the $y$ coordinate (using increments independent from those used for the $x$ coordinate).
7. Plot the $y$ coordinate trajectory as a function of the $x$ coordinate trajectory.

Figure 7 shows the resulting image.

## 8. Conclusions

The contribution of the present article is twofold:

1. It presents a vectorial alternative to the iterative simulation of Brownian bridge trajectories which is based on Gaussian processes for machine learning regression and is relevant regarding the execution speed of interpreted programming languages implementations. The main limitation of the presented alternative is memory space.
2. It extends in a natural way the vectorial simulation of path dependent trajectories to other Gaussian processes such as sequences of Brownian bridges, geometric Brownian motion, fractional Brownian motion, and Ornstein–Ulenbeck mean reversion process.

## Acknowledgments

The authors would like to thank Alexey A. Muravlev, from Steklov Mathematical Institute of RAS, Moscow, Russia, the idea of extending the simulation procedure to path-dependent fractional Brownian motion.

## Funding

J. Beleza Sousa was partially supported by the ISEL/IPL scholarship SFRH/PROTEC/ 49576/2009. R. M. Gaspar's research was partially supported by the SANAF project, financed by the Portuguese Science Foundation with project reference number UTA CMU/MAT/0006/2009.

## References

Alabert, F. (1987). The practice of fast conditional simulations through the lu decomposition of the covariance matrix. *Mathematical Geology* 19(5):369–386.

Björk, T. (2004). *Arbitrage Theory in Continuous Time.* 2nd ed. Oxford:Oxford University Press.

Broadie, M., Glasserman, P. (1997). Pricing american-style securities using simulation. *Journal of Economic Dynamics and Control* 21(8):1323–1352.

Davis, M. W. (1987). Production of conditional simulations via the lu triangular decomposition of the covariance matrix. *Mathematical Geology* 19(2):91–98.

Dietrich, C. R., Newsam, G. N. (1996). A fast and exact method for multidimensional gaussian stochastic simulations: Extension to realizations conditioned on direct and indirect measurements. *Water Resources Research* 32(6):1643–1652.

Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V. (2008). Parallel computing experiences with cuda. *Micro, IEEE* 28(4):13–27.

Gillespie, D. T. (1996). Exact numerical simulation of the Ornstein–Uhlenbeck process and its integral. *Physical Review E* 54(2):2084.

Glasserman, P. (2003). *Monte Carlo Methods in Financial Engineering (Stochastic Modelling and Applied Probability) (v. 53)*. Berlin: Springer.

Hoffman, Y., Ribak, E. (1991). Constrained realizations of gaussian fields-a simple algorithm. *The Astrophysical Journal* 380:L5–L8.

Jean-François, C. (2000). Simulation and identification of the fractional Brownian motion: a bibliographical and comparative study. *Journal of Statistical Software* 5:1–53.

Kasdin, N. J. (1995). Discrete simulation of colored noise and stochastic processes and $1/f^{\alpha}$ power law noise generation. *Proceedings of the IEEE* 83(5): 802–827.

Kloeden, P. E., Platen, E. (1992). *Numerical Solution of Stochastic Differential Equations*. Berlin: Springer.

Mandelbrot, B. B. (1971). A fast fractional gaussian noise generator. *Water Resources Research* 7(3):543–553.

Moskowitz, B., Caflisch, R. E. (1996). Smoothness and dimension reduction in quasi-Monte Carlo methods. *Mathematical and Computer Modelling* 23(8):37–54.

Paxson, V. (1997). Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *ACM SIGCOMM Computer Communication Review* 27(5):5–18.

Quantliba free/open-source library for quantitative finance, 2012. Available at: http://quantlib.org/".

Rasmussen, C. E., Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press.

Volkov, V., Demmel, J. (2008). Lu, qr and cholesky factorizations using vector capabilities of gpus. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-49, May* pp. 2008–49.

Wolfram Research. Inc. (2011). Mathematica edition: Version 8.0.4.0.