mathematics

MDPI

Article

Quadrature Integration Techniques for Random Hyperbolic PDE Problems

Rafael Company [1,*], Vera N. Egorova [2] and Lucas Jódar [1]

[1] Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain; ljodar@imm.upv.es
[2] Departamento de Matemática Aplicada y Ciencias de la Computación, Universidad de Cantabria, Avenida de los Castros s/n, 39005 Santander, Spain; vera.egorova@unican.es
[*] Correspondence: rcompany@imm.upv.es

Abstract: In this paper, we consider random hyperbolic partial differential equation (PDE) problems following the mean square approach and Laplace transform technique. Randomness requires not only the computation of the approximating stochastic processes, but also its statistical moments. Hence, appropriate numerical methods should allow for the efficient computation of the expectation and variance. Here, we analyse different numerical methods around the inverse Laplace transform and its evaluation by using several integration techniques, including midpoint quadrature rule, Gauss–Laguerre quadrature and its extensions, and the Talbot algorithm. Simulations, numerical convergence, and computational process time with experiments are shown.

Keywords: random hyperbolic model; random laplace transform; numerical integration; monte carlo method; numerical simulation; talbot algorithm

check for updates

Citation: Company, R.; Egorova, V.N.; Jódar, L. Quadrature Integration Techniques for Random Hyperbolic PDE Problems. *Mathematics* **2021**, *9*, 160. https://doi.org/10.3390/math9020160

Received: 7 October 2020
Accepted: 12 January 2021
Published: 14 January 2021

1. Introduction

Random hyperbolic partial differential equations (PDEs) are mathematical models that describe wave phenomena with applications in various fields: fluid mechanics [1,2], electromagnetic radiation [3], geosciences [4], and many others. The theory of hyperbolic problems has been well developed based on the assumption that parameters of the model, such as coefficients or initial values are exactly known, which is not available in the real world, where error measurement and the unavailability of the measurement occur. It causes the increasing interest for the random models, which can estimate the impact of the uncertainty to the predicted solution.

The solution is found numerically due to the complexity of random models. Following the mean square approach [5], we can extend existing numerical methods for deterministic problems to the random case by applying the Monte Carlo method [6,7] in order to approximate the statistical moments of the solution. Nevertheless, iterative numerical methods require the storage of the preliminary results and huge number of repetitions, which leads to the the necessity of enormous computational resources and makes them not appropriated to deal with random models. Thus, it becomes urgent to search for an accurate and fast numerical algorithm. Integral transform is a good alternative, as it allows us to construct the solution at one fixed point, not necessarily in the whole domain as it occurs in the case of the finite difference methods, as it is shown in the literature [8].

Integral transform methods convert the original random PDE to an ordinary differential equation (ODE), which can be solved analytically, in some cases, or numerically. Once obtained the solution of the random ODE, the inverse transform is applied in order o restore the solution of the original problem. This inverse transform can be done by the definition, i.e., integrating over the infinite domain, or by using some numerical techniques [9]. There are several widely used methods: Fourier Series, Stehfest approach [10], and Talbot inverse algorithm [11]. Because the inverse Laplace transform is ill-posed problem, the regularization property of the numerical algorithm is necessary. In this sense, the Talbot inverse becomes the best option, since it guarantees the regularization property, while other numerical inversion schemes fail in dealing with noisy data [12].

In this work, we construct a numerical solution for random hyperbolic PDE models, not only by constructing the approximating stochastic process solution, but also while computing its expectation and variance. Thinking of practical applications, we deal with random models where the uncertainty is described by stochastic processes (s.p.'s) having a finite degree of randomness ([5], p. 37); this means that the involved s.p.'s take the form

$$g(x) = G(x, V_1, V_2, \dots, V_m), \tag{1}$$

where V_i, $1 \leq i \leq m$, are mutually independent random variables (r.v.'s).

We propose an analytic-numerical approach that is based on random integral transform technique combined with various numerical integration methods, such as midpoint rule, Gauss-quadratures, and Talbot inverse [11]. The Monte Carlo method is used for the evaluations of the integrands involving the solution of random ordinary differential problems and also for the computation of the expectation and variance of the approximating stochastic process solution. The oscillatory nature of the appearing integrands deserves careful attention, because not all of the quadrature rules are advisable [13–15].

The proposed analytical–numerical approach for solving random hyperbolic PDE problems considered in this paper includes known state of the art of numerical integration methods, which are compared between themselves in terms of accuracy and computational time: the midpoint quadrature rule, the Talbot algorithm for Laplace inverse, the Gauss–Laguerre quadrature, the Exponential-Fitting Gauss-Laguerre quadrature, and the adaptative quadrature. This comparison is provided to highlight the advantages and drawbacks of each method. Moreover, this complex approach is compared with standard finite-difference methods for solving the random hyperbolic PDE problem. In all cases, the Monte Carlo simulations are used in order to calculate the statistical moments of the random solution process.

The rest of the paper is organized, as follows. In Section 2, the random hyperbolic PDE problem is formulated and the random Laplace transform method is briefly described. Section 3 proposes numerical integration methods for Laplace inverse, while Section 4 gives an algorithm for Monte Carlo simulations. All of the proposed methods are compared by the series of numerical tests in Section 5. Section 6 discusses the results.

All of the numerical tests have been carried out by MatLAB, version R2020a, for Windows 10 Home (64-bit), Intel(R) Core(TM) i5-8265U CPU, 1.60 GHz.

2. Preliminaries and Integral Transform for Random Hyperbolic PDE

This section begins by recalling previous results and definitions [8,16]. Let us consider a complete probability space $(\Omega, \mathcal{F}, \mathbb{R})$ and the set L_p with the p-norm of a real-values random variable $Y \in L_p(\Omega)$, as defined by

$$\|Y\|_p = (\mathbb{E}[|Y|^p])^{1/p}, \quad p \geq 1, \tag{2}$$

where the expectation $\mathbb{E}[|Y|^p] < \infty$, and $L_p(\Omega)$ is a Banach space [17]. By using definition (2), the integrability, continuity, and differentiability of a function $Y(t) \in L_p(\Omega)$ can be defined straightforwardly.

Note that, if $p = 2$, then it is a mean square (m.s.) case. Let $\mathcal{C}$ be the class of all m.s. locally integrated two-stochastic processes (s.p.'s) $h(t)$ defined in $\mathbb{R}$ such that $h(t) = 0$, for all negative arguments and the two-norm satisfies

$$\exists c \geq 0,\, M > 0 : \quad \|h(t)\|_2 \leq M \exp(ct), \quad \forall t \geq 0. \tag{3}$$

Subsequently, for $h(t) \in \mathcal{C}$, the m.s. integral

$$H(s) = \mathcal{L}[h(t)](s) = \int_0^\infty h(t)\exp(-st)dt, \tag{4}$$

where s is a complex number with real part $Re(s) > c_0 \geq 0$, and it is called the random Laplace transform of 2-s.p. $h(t)$. The constant c_0 is chosen, such that $Re(s) > c_0$ specifies the region where $H(s)$ is analytic and it has some form of singularity on the line $Re(s) = c_0$ [9]. If $H(s)$ is known, then the random inverse transform for $t > 0$ is defined, as follows

$$h(t) = \frac{1}{2\pi i}\int_{\alpha - i\infty}^{\alpha + i\infty} H(s)\exp(st)ds, \tag{5}$$

where i stands for the imaginary unit and $\alpha > c_0$ [16].

For the purposes of present study, we recall some of the important properties of the random Laplace transform (4): if s.p. $h(t)$ is twice m.s. differentiable and $h'(t)$, $h''(t)$ belong to $\mathcal{C}$, then

$$\mathcal{L}[h'(t)](s) = sH(s) - h(0+), \qquad \mathcal{L}[h''(t)](s) = s^2H(s) - sh(0+) - h'(0+). \tag{6}$$

In this paper, we consider a one-dimensional random hyperbolic PDE modelling the s.p. of the vibrating string motion $u(x,t)$, depending on the spatial variable x and time t,

$$u_{tt}(x,t)(\xi) = a(x)(\xi)u_{xx}(x,t)(\xi) + b(x)(\xi)u_x(x,t)(\xi) + c(\xi)u(x,t)(\xi), \quad x \in [0,\, L],\ t > 0,\ \xi \in \Omega, \tag{7}$$

$$u(x,0)(\xi) = f_0(x)(\xi), \quad u_t(x,0)(\xi) = f_1(x)(\xi), \tag{8}$$

$$u(0,t)(\xi) = g_0(t)(\xi), \quad u(L,t)(\xi) = g_1(t)(\xi), \tag{9}$$

where $a(x)(\xi) > 0$, $b(x)(\xi)$ are m.s.-continuous stochastic processes with a finite degree of randomness and absolutely integrable with respect to the spatial variable in $\mathbb{R}$; $c(\xi)$ is a random variable (r.v.). The s.p.'s $f_0(x)(\xi)$, $f_1(x)(\xi)$, $g_0(t)(\xi)$, and $g_1(t)(\xi)$ are functions depending on a finite number of r.v. that represent random initial and boundary conditions with a finite degree of randomness.

The random hyperbolic partial differential equation (PDE) (7) is solved using an analytic-numerical method that is based on Laplace transform combined with an appropriate numerical integration technique. In this paper, we consider various quadratures for inverse Laplace transform.

Following the ideas of [8,18], let us define the random Laplace transform with respect to the temporal variable, as

$$U(x,s)(\xi) = \mathcal{L}[u(x,t)(\xi)]. \tag{10}$$

Because $u(x,t)(\xi)$ is a twice m.s. differentiable s.p., one gets

$$\mathcal{L}[u_{tt}(x,t)(\xi)] = s^2U(x,s)(\xi) - su(x,0)(\xi) - u_t(x,0)(\xi) = s^2U(x,s)(\xi) - sf_0(x)(\xi) - f_1(x)(\xi). \tag{11}$$

Subsequently, (7) is transformed to the following random non-homogeneous ordinary differential equation (ODE) with respect to the spatial variable

$$a(x)(\xi)U_{xx}(x,s)(\xi) + b(x)(\xi)U_x(x,s)(\xi) + (c - s^2)U(x,s)(\xi) = -[sf_0(x)(\xi) + f_1(x)(\xi)], \tag{12}$$

for $x \in [0,\, L]$, $\xi \in \Omega$.

Assuming $a(x)(\xi) > 0$ for each event $\xi \in \Omega$, one gets

$$U_{xx}(x,s)(\xi) + \frac{b(x)(\xi)}{a(x)(\xi)}U_x(x,s)(\xi) + \frac{c - s^2}{a(x)(\xi)}U(x,s)(\xi) = -\frac{sf_0(x)(\xi) + f_1(x)(\xi)}{a(x)(\xi)}. \tag{13}$$

Equation (13) is a linear second order ODE with respect to the spatial variable, which can be analytically solved in some cases, or numerically in other cases. Because the boundary conditions (9) for the PDE are functions on t, the boundary conditions for (13) are the corresponding Laplace transforms of (9):

$$U(0,s)(\xi) = \mathcal{L}[g_0(t)(\xi)], \quad U(L,s)(\xi) = \mathcal{L}[g_1(t)(\xi)]. \tag{14}$$

Once obtaining the solution $U(x,s)(\xi)$, a real-valued $u(x,t)(\xi)$ is restored by while using random inverse Laplace transform that is given by (5). Taking advantage of the relationship between the inverse Laplace transform and Fourier cosine integrals, see [9], the following formula is used

$$u(x,t)(\xi) = \frac{2e^{\alpha t}}{\pi} \int_0^\infty Re[U(x,\alpha + iw)(\xi)] \cos(wt) dw, \quad \xi \in \Omega, \tag{15}$$

where $Re[\cdot]$ stands for the real part of a complex number. Note that the integrand appearing in (15) has an oscillatory kernel that deserves special care for the numerical integration.

3. Numerical Integration Methods

This section describes briefly acknowledged integration methods for the integrals of the type (15).

THe numerical solution of Equation (7) is constructed in the domain $\Delta = [0;\, L] \times [0;\, T]$ for each fixed event ξ. Let us introduce a uniform grid $\{x_j, t^n\}$, such that

$$x_j = jh,\ h = \frac{L}{N_x},\ j = 0, \dots, N_x; \quad t^n = nk,\ k = \frac{T}{N_t},\ n = 0, \dots, N_t. \tag{16}$$

At each node (x_j, t^n), the numerical solution is defined by $u_j^n(\xi)$ for each realization of ξ and it is obtained by approximating the integral (15). Hence, at every fixed (x_j, t^n), the following function is defined

$$f_{j,n}(w) = f_{j,n}(w, \xi) = Re\big[U(x_j, \alpha + iw)(\xi)\big] \cos(wt^n), \tag{17}$$

where $U(x_j, \alpha + iw)(\xi)$ is the numerical solution of ODE (13) at the point x_j for fixed value of $s = \alpha + iw$. Now, we briefly describe all of the considered methods for numerical integration.

3.1. Midpoint Quadrature Rule

The midpoint quadrature rule is a method of approximation of integral (15) based on the Riemann sums, the simplest case of Newton–Cotes open formulas, for truncated domain $[0, R]$. In the general case, the midpoint quadrature rule is written, as follows

$$\int_0^\infty f(w) dw \approx \int_0^R f(w) dw = \sum_{k=0}^{N} f(w_{k+1/2}) h_{MP} + O(h^2), \tag{18}$$

where $w_{k+1/2} = \left(k + \frac{1}{2}\right) h_{MP}$, $h_{MP} = \frac{R}{N}$, $k = 0, \dots, N-1$.

It is well known that the main advantage of this method is its simplicity of implementation and the consideration of all the information regarding the integrand, which makes it applicable for a wider class of integrand functions [14]. However, the high accuracy of the quadrature requires large enough value of N, leading to the increasing computational cost. In the case of improper integral (in the infinite domain), the method can also be sensitive to the choice of R.

3.2. *Gauss-Laguerre Quadrature*

The novelty of Gauss quadratures is to choose nodes where the integrand is evaluated in order to minimize the error of approximation. It is a good alternative to Newton–Cotes formulas, especially when the evaluation of function itself requires a lot of computational resources, because good accuracy can be reached with a small number, four or five, of nodes if the integrand is well conditioned. This is not the case when the integrand is of oscillatory type [19].

The improper integral is approximated by Gauss–Laguerre (GL) quadrature of N_{GL} nodes by the following sum, see [20],

$$\int_0^\infty f(w)dw \approx \sum_{k=1}^{N_{GL}} \gamma_k f(w_k) e^{w_k}, \tag{19}$$

where w_k is the k-th root of Laguerre polynomial $L_{N_{GL}}(w)$, γ_k is the weight of the quadrature given by

$$\gamma_k = \frac{w_k}{(N_{GL}+1)^2 \left[L_{N_{GL}+1}(w_k)\right]^2}, \quad k = 1, \dots, N_{GL}. \tag{20}$$

3.3. *Exponentially-Fitted Gauss-Laguerre Quadrature*

Exponential fitting is an approach that is used in numerical differentiation, interpolation, and integration for improving the accuracy of the methods. Because integrand in (15) is oscillating, Exponentially-fitted Gauss–Laguerre quadrature (EF-GL), as proposed in [21], could be a good option. For EF-GL, nodes and weights depend on integrand and cannot be defined a priori. The computation of these N_{GL} pairs of nodes and weights is based on the solution of a nonlinear system of N_{GL} equations, which leads to additional computational cost. In [21], the numerical algorithm is described in details. Further, in Section 5, we compare the accuracy and computational time of GL and EF-GL quadrature rules.

3.4. *Talbot Inverse*

The method of Talbot for the Laplace inversion problem [11] is based on numerical contour integration. Instead of formula (15), the Bromwich integral is used

$$u_j^n(\xi) = \frac{1}{2\pi i} \int_{\alpha - i\infty}^{\alpha + i\infty} e^{st^n} U(x_j, s) ds. \tag{21}$$

The contour deformation is used in order to obtain the Hankel contour and exploit the exponential factor, which makes the integral suitable for further application of a Newton–Cotes formula [22]. The Talbot inversion quadrature for N_{TI} nodes is written, as follows

$$u_j^n(\xi) = \frac{2}{5t^n} \sum_{k=0}^{N_{TI}-1} Re\left[\gamma_k U(x_j, \frac{w_k}{t^n})\right], \tag{22}$$

where w_k are the nodes and γ_k are the weights defined by

$$w_0 = \frac{2N_{TI}}{5}, \quad w_k = \frac{2\pi k}{5}\left(\cot\left(\frac{k\pi}{N_{TI}} + i\right)\right), \tag{23}$$

$$\gamma_0 = 0.5\exp(w_0), \quad \gamma_k = exp(w_k) \cdot \left[1 + \frac{k\pi}{N_{TI}}\left(1 + \cot^2\left(\frac{k\pi}{N_{TI}}\right) - i\cot\left(\frac{k\pi}{N_{TI}}\right)\right)\right]. \tag{24}$$

Here, the number of nodes N_{TI} should be chosen in accordance with desired accuracy: for n significant digits $N_{TI} = \lceil 1.7n \rceil$. It shows the flexibility of the method and the high degree of accuracy with fast convergence. Moreover, as stated in [12], the main advantage of the Talbot algorithm is the regularization property, which means the ability to handle noisy data. It is important for the inverse Laplace transform problem due to its ill-posedness and it becomes even more urgent in the random case dealing with perturbed initial conditions or parameters of the problem.

Summarizing, a numerical solution is constructed following the steps of Algorithm 1 for all of the described methods.

Algorithm 1: Numerical solution for deterministic string vibrating problem

```
Initialization: set the mesh $\{x_j, t^n\}$ by (16);
Set initial conditions $u(x_j, 0) = f_0(x_j)$, $j = 0, \dots, N_x$;
Set $\alpha > c_0$;
Set number of nodes of the quadrature $N$;
Set $n = 0$;
while $t^n < T$ do
    Increment $n$;
    for $j = 0, \dots, N_x$ do
        Compute nodes and weigths $\{w_k, \gamma_k\}$ of the chosen quadrature
        - Midpoint rule: uniform grid with $N$ nodes ;
        - GL quadrature: nodes $w_k$ are the roots of the Laguerre polynomial of $N$-th order, $k = 1, \dots, N$;
        - EF-GL quadrature [21]: nodes $w_k$ and weights $\gamma_k$ are found by solving nonlinear system of $2N$ equations;
        - Talbot inverse: nodes $w_k$ and weights $\gamma_k$, $k = 0, \dots, N-1$ are defined by (23)–(24);
        Get the approximated value $u_j^n$:
        - Midpoint rule: integral in (15) is approximated by (18);
        - GL and EF-GL quadratures: integral in (15) is approximated by (19)–(20);
        - Talbot inverse: formula (22) ;
    end
end
```

4. Monte Carlo Method for Random Hyperbolic PDE

The coefficients of the random m.s. Equation (7) and corresponding initial and boundary conditions (9) are stochastic processes (s.p's) that are defined in a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$, i.e., s.p.'s $a(x)$, $b(x)$, $f_0(x)$, $f_1(x)$, $g_0(x)$ and $g_1(x)$ are described as continuous s.p.'s with with one-degree of randomness.

The solution of the random m.s. problem is approximated by using the the Monte Carlo approach [6,7], when the expectation $\mathbb{E}[u(x,t)]$ is approximated by the average of a sufficiently large number of realizations $\xi \in \Omega$ of the corresponding deterministic realized transformed random ordinary differential problem. The Algorithm 2 describes the steps of the numerical solution.

Algorithm 2: Numerical solution for random hyperbolic PDE problem

```
Initialization: set the mesh $\{x_j, t^n\}$ by (16);
Set number of the MC realizations $N_{MC}$;
Generate $N_{MC}$ random variables for s.p.'s $a(x)$, $b(x)$, $f_0(x)$, $f_1(x)$, $g_0(x)$;
Choose the method of numerical integration for $m = 1, \dots N_{MC}$ do
    Define s.p.'s $a(x)$, $b(x)$, $f_0(x)$, $f_1(x)$, $g_0(x)$ for fixed realization;
    Run Algorithm 1 to obtain the numerical solution $u_m$ of the deterministic problem;
    Increment $m$;
end
Compute $\mathbb{E}[u] = \sum_{m=1}^{N_{MC}} \frac{u_m}{N_{MC}}$;
Compute $\mathbb{E}[u^2] = \sum_{m=1}^{N_{MC}} \frac{u_m^2}{N_{MC}}$;
Compute $\sqrt{\mathrm{Var}[u]} = \sqrt{\mathbb{E}[u^2] - (\mathbb{E}[u])^2}$
```

5. Numerical Results

This section deals with the comparison of the above-described methods of numerical integration and Laplace inversion for several test problems.

5.1. Deterministic PDE Problem with Constant Coefficients

We start with simple one dimensional deterministic problem with a known analytical solution in order to check the viability of the proposed numerical integration techniques. The deterministic example corresponds to one fixed event $\xi \in \Omega$. Instead of the bounded spatial domain $[0; L]$, the whole real axis $\mathbb{R}$ is considered. Thus, no boundary conditions are needed. We also assume that $a > 0$, b, and c are constants, i.e., the following wave equation is considered

$$u_{tt}(x,t) = a^2 u_{xx}(x,t) + b u_x(x,t) + c u(x,t), \quad x \in \mathbb{R}, t > 0, \tag{25}$$

subject to initial conditions $u(x,0) = f_0(x)$, $u_t(x,0) = f_1(x)$.

This problem admits an analytical solution that can be written in terms of Bessel function of the first kind, see [23], p. 574, Equation 6.1.5, as follows

- for $c - \frac{1}{4}a^{-2}b^2 = \sigma^2 > 0$:

$$\begin{aligned} u(x,t) &= \frac{1}{2} f(x+at)\exp\left(\frac{bt}{2a}\right) + \frac{1}{2} f(x-at)\exp\left(-\frac{bt}{2a}\right) \\ &+ \frac{\sigma t}{2a}\exp\left(-\frac{bx}{2a^2}\right)\int_{x-at}^{x+at}\exp\left(\frac{b\xi}{2a^2}\right)\frac{I_1\left(\sigma\sqrt{t^2-(x-\xi)^2/a^2}\right)}{\sqrt{t^2-(x-\xi)^2/a^2}} f(\xi)d\xi \\ &+ \frac{1}{2a}\exp\left(-\frac{bx}{2a^2}\right)\int_{x-at}^{x+at}\exp\left(\frac{b\xi}{2a^2}\right) I_0\left(\sigma\sqrt{t^2-(x-\xi)^2/a^2}\right) g(\xi)d\xi, \end{aligned} \tag{26}$$

  where $I_0(z)$ and $I_1(z)$ are the modified Bessel function of the first kind;
- for $c - \frac{1}{4}a^{-2}b^2 = -\sigma^2 < 0$:

$$\begin{aligned} u(x,t) &= \frac{1}{2} f(x+at)\exp\left(\frac{bt}{2a}\right) + \frac{1}{2} f(x-at)\exp\left(-\frac{bt}{2a}\right) \\ &- \frac{\sigma t}{2a}\exp\left(-\frac{bx}{2a^2}\right)\int_{x-at}^{x+at}\exp\left(\frac{b\xi}{2a^2}\right)\frac{J_1\left(\sigma\sqrt{t^2-(x-\xi)^2/a^2}\right)}{\sqrt{t^2-(x-\xi)^2/a^2}} f(\xi)d\xi \\ &+ \frac{1}{2a}\exp\left(-\frac{bx}{2a^2}\right)\int_{x-at}^{x+at}\exp\left(\frac{b\xi}{2a^2}\right) J_0\left(\sigma\sqrt{t^2-(x-\xi)^2/a^2}\right) g(\xi)d\xi, \end{aligned} \tag{27}$$

  where $J_0(z)$ and $J_1(z)$ are Bessel function of the first kind.

In order to test the proposed numerical integration methods we apply Laplace transform, as described in Section 2, and obtain a deterministic version of Equation (13):

$$U_{xx}(x,s) + \frac{b}{a^2}U_x(x,s) + \frac{c-s^2}{a^2}U(x,s)(\xi) = -\frac{sf_0(x)+f_1(x)}{a^2}. \tag{28}$$

Applying the non-unitary Fourier transform with angular frequency

$$\hat{U}(w,s) = \mathcal{F}[U(x,s)] = \int_{-\infty}^{\infty} U(x,s)\exp(-ixw)dx, \tag{29}$$

Equation (13) takes the following form

$$-w^2\hat{U}(w,s) + iw\frac{b}{a^2}\hat{U}(w,s) + \frac{c-s^2}{a^2}\hat{U}(w,s) = -\mathcal{F}\left[\frac{sf_0(x)+f_1(x)}{a^2}\right]. \tag{30}$$

Algebraic Equation (30) is solved directly

$$\hat{U}(w,s) = \frac{-\mathcal{F}\left[\frac{sf_0(x)+f_1(x)}{a^2}\right]}{-w^2 + iw\frac{b}{a^2} + \frac{c-s^2}{a^2}}. \tag{31}$$

Hence, the solution $U(x,s)$ of (28) can be obtained by applying inverse Fourier transform to (31).

In the next Example 1, we consider a particular case of Equation (25) with constant coefficients and trigonometric initial conditions.

Example 1. *Let us consider deterministic problem* (25) *with coefficients* $a = 2$, $b = 1$, $c = 3$, *and initial conditions* $f_0(x) = \cos(x)$ *and* $f_1(x) = \sin(x)$.

Numerical solution is constructed in the truncated domain $[0, L] \times [0, T]$, $L = 5$, $T = 1$, for discrete uniformly distributed nodes (16), $N_x = 5$, $N_t = 10$, by applying the described in previous section methods, see Algorithm 1. We set $\alpha = 1$.

Applying the inverse Fourier transform to (31), one obtains

$$U(x,s) = \frac{1}{2\pi}\left[\frac{\pi e^{-xi}(s+i)}{a^2 + bi + (s^2 - c)} + \frac{\pi e^{xi}(s-i)}{a^2 - bi + (s^2 - c)}\right], \quad s = \alpha + iw, \tag{32}$$

where i is the imaginary unit. Once the solution of ODE (28) is obtained, formula (15) is used to restore the solution of the PDE while using various numerical integration techniques.

Note that Equation (25) admits the analytical solution, as described above. Because the function $u(x,t)$ is close to zero, we compute the relative error of the discrete numerical solution at the mesh nodes in order to estimate the accuracy of the methods

$$\mathrm{RelErr}(j,n) = \frac{|u_{\mathrm{ref}}(x_j, t^n) - U_{\mathrm{num}}(x_j, t^n)|}{|u_{\mathrm{ref}}(x_j, t^n)|}, \tag{33}$$

where U_{num} is the matrix of numerical solution $U_{\mathrm{num}} = \{u_j^n\}$, $j = 0, \ldots, N_x$, $n = 0, \ldots, N_t$, as computed by Algorithm 1; $u_{\mathrm{ref}}(x_j, t^n)$ is the reference value at the point (x_j, t^n). In this example, as the exact solution is known, the reference value is equal to this exact solution. For other cases where the exact solution is not available, a reference value is obtained using accurate finite difference method (FDM) for solving the original PDE (7). The total computational time for the proposed methods are presented in Table 1, together with the maximum of $\mathrm{RelErr}(j,n)$.

The adaptative quadrature (MatLAB function `integral` [24]) has the same order of accuracy as the midpoint rule, but it requires greater computational resources. Thus, it will not be considered in further more complicated examples.

For the Talbot algorithm $M = 17$ is chosen to guarantee the accuracy up to 10 significant digits [22]. Even in that case, the method performs much faster than standard numerical integration methods for (15). Thus, the Talbot inverse method is found to be the most effective method for the deterministic case with constant coefficients.

Table 1. Comparison of various numerical methods for problem (25) with $a = 2$, $b = 1$ and $c = 3$ (Example 1).

Method	**Error**	**CPU-Time, s**
Midpoint rule ($R = 10^4, h_{MQ} = 0.1$)	4.4700×10^{-7}	0.59
Gauss-Laguerre ($N_{GL} = 25$)	3.5551×10^{-1}	0.05
EF-GL (five nodes)	7.9303	3.97
Talbot inverse ($M = 17$)	7.3457×10^{-11}	0.02
Adaptative quadrature	4.8559×10^{-6}	7.53

The relative errors for Midpoint rule and Talbot inverse methods are plotted in Figure 1. Because no boundary conditions are posed for the problem, the largest values of the relative errors are situated at the boundary $x = L$.

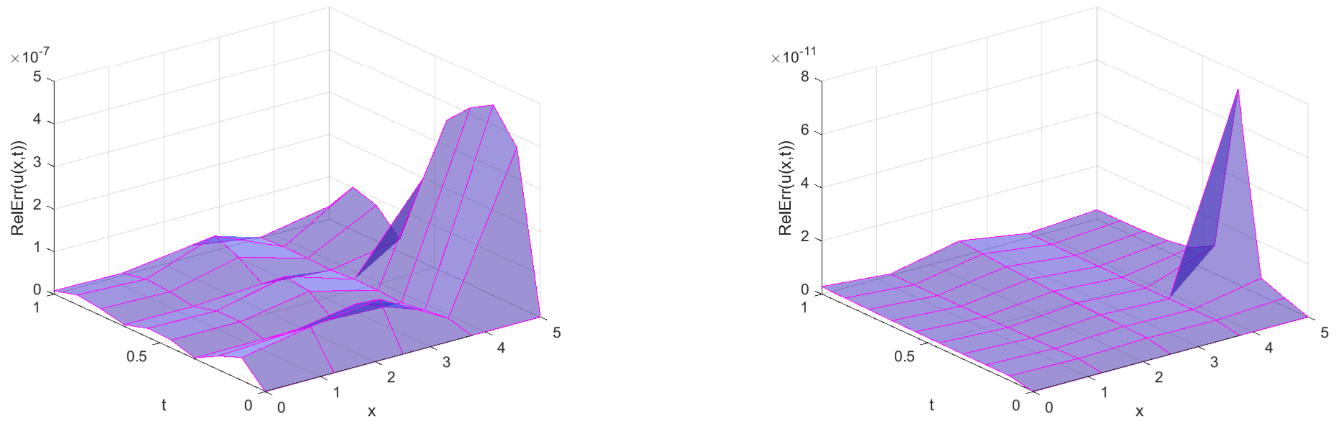

Figure 1. Distribution of the relative error among space and time for midpoint rule (**left**) and Talbot inverse (**right**) methods in Example 1.

Table 2 presents a comparison of GL and EF-GL quadratures in terms of the maximum relative error and the CPU-time varying the number of nodes N_{GL}. It is important to notice that the CPU-time may vary from simulation to simulation, thus only the order should be taken into account. In the case of GL quadrature, we find out that the computational time is similar with increasing number of nodes, while the CPU-time for EF-GL method is increasing exponentially. The convergence of the GL quadrature is shown, while taking the results shown in Table 1 into account: the error reduces significantly with an increasing number of nodes. The potential improvement of the GL method by the exponential fitting expectedly has higher computational cost, due to the solution of the nonlinear system at each point of the computational domain. However, the accuracy of the EF-Gl quadrature for this example with oscillating integrand has not been improved when comparing with the standard GL rule. Thus, it will not be considered in further more complicated examples.

Table 2. Gauss–Laguerre (GL) and Exponentially-fitted Gauss–Laguerre quadrature (EF-GL) methods results, depending on number of nodes of the quadrature for Example 1.

N_{GL}	**3**	**5**	**8**	**15**
GL Error	8.2739	7.4679	2.5601	1.4560
GL CPU-time, s	0.02	0.02	0.05	0.05
EF-GL Error	9.4937	7.9303	7.5438	3.9366×10^3
EF-GL CPU-time, s	0.44	1.84	15.84	430.89

The accuracy of the midpoint rule depends on the truncation R and step size h_{MP}. A bigger domain, as well as smaller step size, lead to an increased computational time. Figure 2 presents the plots of errors and the CPU-time for fixed step size $h_{MP} = 10^{-1}$ with respect to increasing domain. The accuracy in dependence on the step size h_{MP} is also studied. In Table 3, the maximum relative error is reported for various h_{MP} and fixed $R = 10^4$. The maximum relative error is decreasing with step size until 4.4699×10^{-7} ($h_{MP} = 1/16$); further fragmentation of the step size does not reduce the error for $R = 10^4$.

Table 3. The maximum relative error and computational time of the midpoint quadrature rule with respect to step size h_{MP} for Example 1.

h_{MP}	Error	CPU-Time, s
1/1	2.8832×10^{-1}	0.06
1/2	3.1862×10^{-2}	0.06
1/4	7.0942×10^{-5}	0.19
1/8	4.4700×10^{-7}	0.22
1/16	4.4699×10^{-7}	0.53
1/32	4.4699×10^{-7}	1.03
1/64	4.4699×10^{-7}	1.47

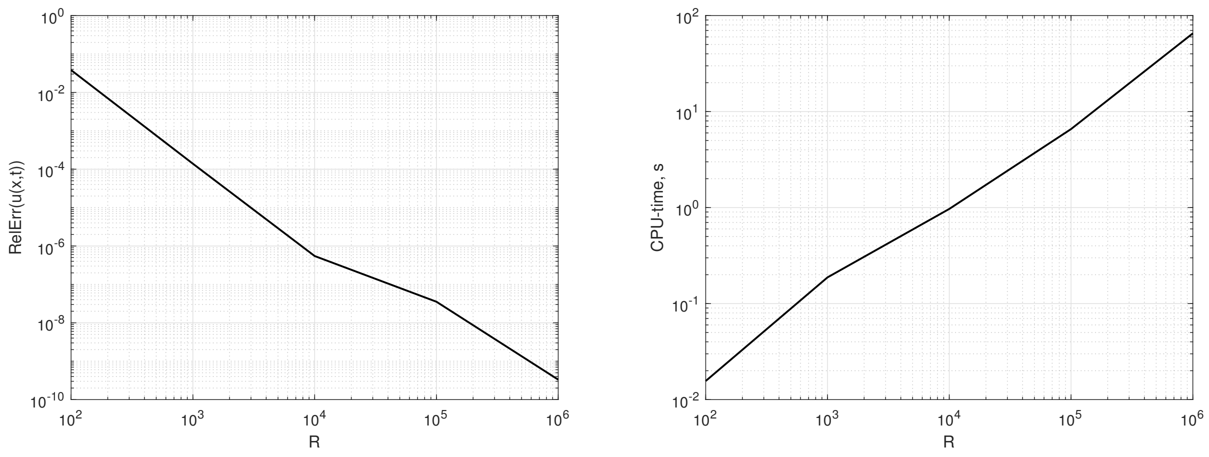

Figure 2. Error and total computational time of the midpoint rule with respect to the domain size R with fixed $h = 10^{-1}$ for Example 1.

5.2. *Deterministic PDE with Non-Constant Coefficients*

In the case of non-constant coefficients in (13), the analytical solution is not always available; thus, FDM is applied to construct a reference numerical solution. Note that the function $U(x,s)$ that is used in expression (17) means the value of the numerical solution of the ODE (13) at the fixed point x for fixed parameter s.

Equation (13) is discretized by the central differences on the same mesh $\{x_j\}$, $j = 0, \ldots, N_x$, as follows

$$\frac{U_{j+1} - 2U_j + U_{j-1}}{h^2} + \frac{b(x_j)}{a(x_j)} \frac{U_{j+1} - U_{j-1}}{2h} + \frac{c - s^2}{a(x_j)} U_j = -\frac{s f_0(x_j) + f_1(x_j)}{a(x_j)}, \quad j = 1, \ldots, N_x - 1, \tag{34}$$

where U_j stands for the approximated value of $U(x,s)$ at the node x_j. The values at the boundaries are found from the boundary conditions by applying the Laplace transform

$$U_0 = \mathcal{L}[g_0(t)], \quad U_{N_x} = \mathcal{L}[g_1(t)]. \tag{35}$$

Hence, the integrand (17) has to be evaluated at each fixed node of the computational grid in order to approximate integral (15), which provokes a significant augment of the CPU-time. In the next example, we increase the complexity by regarding a variable coefficients deterministic problem.

Because the analytical solution for the deterministic PDE problem in general form (7) is not available, a numerical method has to be employed to obtain the reference numerical solution. We consider an explicitly centred in time and space finite difference scheme for the mesh function $u_j^n \approx u(x_j, t^n)$:

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{(\Delta t)^2} = a(x_j)\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} + b(x_j)\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + cu_j^n, \tag{36}$$

where $j = 1, \dots, N_x$, $n = 2, \dots, N_t$. The initial conditions (8) are used in order to obtain the solution at the first time levels t^0 and t^1. The derivative in (8) is approximated by the forward difference. Because the considered scheme is conditionally stable, the step sizes Δt and Δx are chosen to guarantee the stability. In order to obtain a good approximation, which could be considered as the reference solution, the mesh should be chosen appropriately fine.

Example 2. *Let us consider a deterministic vibrating string problem* (7) *on rectangle* $[0, L] \times [0, T]$, $L = 0.5$, $T = 0.2$. *We set non-constant coefficients* $a(x) = 9x + 1$, $b = -e^x$, $c = -5$, *initial conditions* $f_0(x) = x(x - L)$ *and* $f_1(x) = 0$, *and boundary conditions* $g_0(t) = g_1(t) = 0$.

The numerical solution is constructed by the Algorithm 1, choosing $N_x = 10$, $N_t = 5$. For the midpoint rule, $N = 100$ and $R = 100$ are used. Table 4 presents the comparison of the methods in terms of maximum relative error and computational time. The reference solution is the numerical solution that is computed by the FDM (36) in refined mesh ($N_x = 100$, N_t = 16,000), which preserves the stability of the scheme. Because an explicit method is used and no iterative procedures are needed for solving nonlinear system at each time-level, the total computational time is comparably small: 0.15 s. Figure 3 plots the reference solution.

Table 4. A comparison of various methods of numerical integration for Example 2.

Method	**Error**	**CPU-Time, s**
Midpoint quadrature	4.5628×10^{-2}	116.38
Talbot inverse	7.1305×10^{-2}	51.00
Gauss-Laguerre (9 nodes)	7.4450×10^{-1}	5.00
Gauss-Laguerre (25 nodes)	7.9014×10^{-2}	17.48

Figure 4 plots the solution at the moment $t = T$. The midpoint rule and Talbot inverse method perform more accurately than GL quadrature of nine nodes, but they require more computational time due to larger number of calls of integrand (17). However, taking 25 nodes in the GL quadrature, the accuracy has been improved significantly.

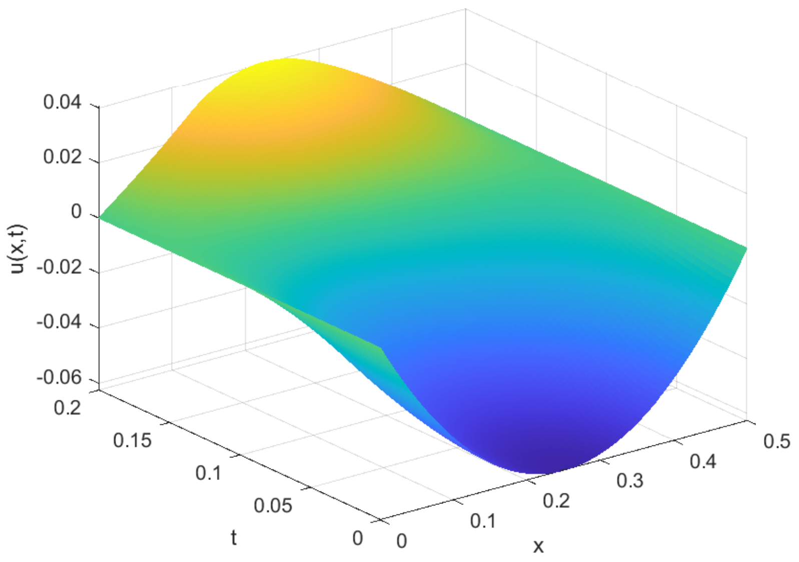

Figure 3. Reference solution for Example 2 computed by the finite difference method (FDM) (36).

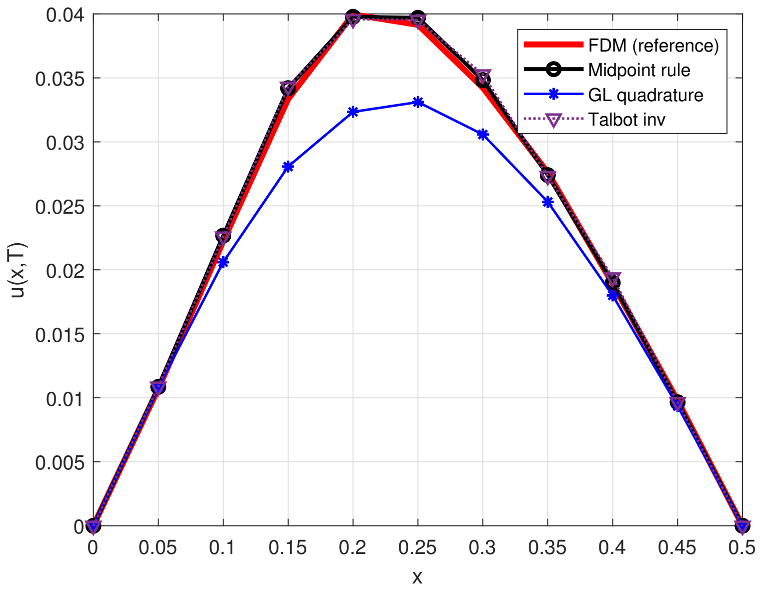

Figure 4. Numerical solution for Example 2 at the moment $t = T$ obtained by considered methods.

5.3. *Random PDE with Constant Coefficients*

In this subsection, we deal with random models with constant coefficient random variables. It is remarkable that, in this case, we need not only the computation of the approximation s.p. solution, but also the computation of its statistical moments.

Example 3. *We consider a random version of problem* (25)*, with $a \sim \mathcal{N}(2, 0.25)$, $b, c \sim Beta(2, 5)$. In order to approximate the mean and variance of the solutions, the Monte Carlo method with N_{MC} simulations is used.*

Expectation and variance of the exact solution for the random hyperbolic PDE (25) are plotted in Figure 5. As in previous examples, we compare the proposed methods of integration and Laplace inverse in terms of maximum relative error and computational time. Table 5 presents the results for various N_{MC}. The CPU-time refers to the total computational time for all N_{MC} simulations. Note that, for 1000 simulations, the exact solution (26)–(27) requires 28.41 s to perform the simulations. Thus, Midpoint rule ($R = 100$, $h = 0.1$), Talbot inverse and GL quadrature require less computational time than calculation by the exact formula. As expected, the computational time is increasing with the number of simulations linearly, but errors preserve the order in most cases.

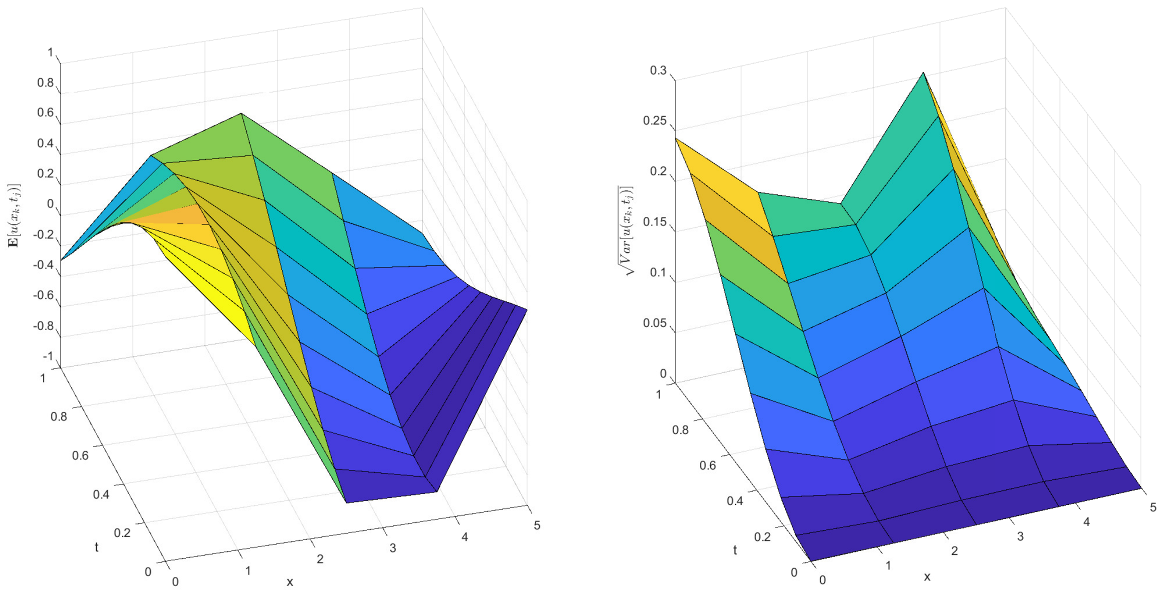

Figure 5. Expectation and variance of the exact solution for the random hyperbolic partial differential equation (PDE) (25) with $a \sim \mathcal{N}(2, 0.25)$, $b, c \sim \text{Beta}(2, 5)$, performed using the Monte-Carlo method with $N_{MC} = 10^3$ simulations.

Table 5. Comparison of various methods of numerical integration for the random hyperbolic PDE (25) with $a \sim \mathcal{N}(2, 0.25)$, $b, c \sim \text{Beta}(2, 5)$.

Method	**Error of Mean**	**Error of Variance**	**CPU-Time, s**
	$N_{MC} = 500$		
Midpoint rule	5.6048×10^{-2}	2.3034×10^{-2}	4.88
Talbot inverse	5.0744×10^{-2}	2.3032×10^{-2}	6.83
GL quadrature (3 nodes)	1.9009×10^{-1}	2.1436×10^{-1}	2.92
	$N_{MC} = 1000$		
Midpoint rule	4.1520×10^{-2}	2.5102×10^{-2}	7.52
Talbot inverse	4.0345×10^{-2}	2.5102×10^{-2}	12.67
GL quadrature	1.9086×10^{-1}	2.1503×10^{-1}	5.64
	$N_{MC} = 2000$		
Midpoint rule	3.7210×10^{-2}	1.2823×10^{-2}	16.88
Talbot inverse	3.1905×10^{-2}	1.2823×10^{-2}	24.86
GL quadrature	1.9001×10^{-1}	2.1443×10^{-1}	11.30
	$N_{MC} = 4000$		
Midpoint rule	4.6382×10^{-2}	9.8022×10^{-3}	32.89
Talbot inverse	4.1078×10^{-2}	9.7971×10^{-3}	51.09
GL quadrature	1.8993×10^{-1}	2.1581×10^{-1}	24.31

5.4. Random PDE with Non-Constant Coefficients

To complete the study, a random variable coefficient problem is considered.

Example 4. *The vibration of the string in* $[0, L]$ *is described by Equation* (7), *subject to the initial conditions* $f_0(x) = x(x - L)$ *and* $f_1(x) = 0$; *and boundary conditions* $g_0(t) = g_1(t) = 0$. *We set up the parameters:*

$$L = 0.5, \quad T = 0.2, \quad a(x) = \varphi x + 1, \ \varphi \sim \mathcal{N}(9, 0.5), \quad b(x) = -e^x, \quad c \sim Beta(2, 5). \tag{37}$$

Unlike the deterministic Example 2 with non-constant coefficients where FDM provides a reference analytical solution, reference values are not available here due to the computational complexity that arises in the evaluation of the statistical moments of the approximate stochastic process when time step advances [18]. A survival reference FDM solution is taking the Monte Carlo method for an appropriate set of realizations. In this case, the number of realizations is $N_{MC} = 10^3$ and CPU-time is 16,212 s.

Figure 6 plots the numerical solution. The zero-variance at the boundaries is caused by the boundary conditions. Similar plots are obtained for the considered methods. Thus, we compare them in terms of the maximum relative error, see Table 6. As it is expected from the previous examples, the most accurate solution is obtained by the midpoint rule and Talbot inverse, although this advantage pays the price of additional computational cost.

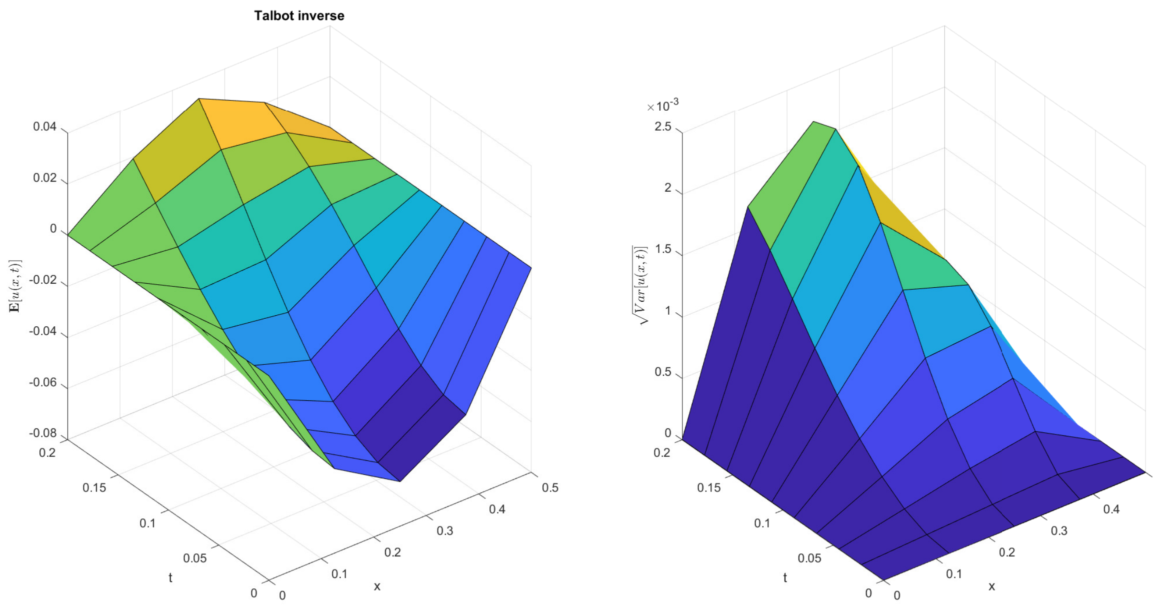

Figure 6. Expectation and variance of the numerical solution for the random hyperbolic PDE (25) with $\varphi \sim \mathcal{N}(9, 0.5)$, $b(x) = -e^x$, $c \sim \text{Beta}(2, 5)$, performed by using the Monte-Carlo method with $N_{MC} = 10^3$ simulations.

Table 6. Comparison of various methods of numerical integration for Example 4, $N_{MC} = 1000$.

Method	**Error of Mean**	**Error of Variance**	**CPU-Time, s**
Midpoint rule	3.8216×10^{-2}	2.2638×10^{-2}	65,385.00
Talbot inverse	3.4976×10^{-2}	2.1375×10^{-2}	28,965.54
GL quadrature (9 nodes)	1.8671×10^{-1}	2.5603×10^{-2}	7192.14

6. Conclusions

The solution of a random hyperbolic PDE problem is a challenging task that is demanded in many practical applications. Computing an expression of the approximating stochastic process makes the computation of its statistical moments available. In this paper, we propose a combination of the random Laplace transform with the numerical integration techniques for its inverse, and the Monte Carlo method for the evaluation of numerical solution of the transformed problem at a particular required point.

The Monte Carlo simulations require a fast and efficient basis numerical algorithm for solving deterministic hyperbolic PDE problem, for every fixed realization. FDM could not be an option due to the high computational cost and memory requirements. In order to avoid the numerical differentiation of the PDE, Laplace transform is applied, which results in ODE equation. In some cases, as it has been shown in present paper, the analytical solution of ODE is known; thus, we use numerical integration methods for inverse Laplace transform. If the solution of ODE is not available, then numerical techniques for boundary value problem have to be employed.

Several numerical integration methods have been considered: midpoint rule and GL-quadrature for improper integrals. However, due to the oscillatory behaviour of the integrand function GL quadrature with a small number of nodes shows comparatively poor results, while the midpoint rule is comparable with Talbot's Laplace inverse for random hyperbolic PDEs. The proposed complex analytic-numerical approach is compared with the classical explicit FDM scheme for the original random PDE problem.

Author Contributions: R.C., V.N.E. and L.J. contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the Spanish Ministerio de Economía, Industria y Competitividad (MINECO), the Agencia Estatal de Investigación (AEI) and Fondo Europeo de Desarrollo Regional (FEDER UE) grant MTM2017-89664-P.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We state that data are available to the readers.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FDM	finite difference method
GL	Gauss-Laguerre
m.s.	mean square
ODE	ordinary differential equation
PDE	partial differential equation
r.v.	random variable
s.p.	stochastic process

References

1. Pettersson, M.P.; Iaccarino, G.; Nordström, J. *Polynomial Chaos Methods for Hyperbolic Partial Differential Equations. Numerical Techniques for Fluid Dynamics Problems in the Presence of Uncertainties*; Springer International Publishing: Cham, Switzerland, 2015; p. 214.
2. Yeh, K.C.; Liu, C.H. Wave Propagation in Random Media. In *Theory of Ionospheric Waves*; Academic Press: Cambridge, MA, USA, 1972; Volume 17, pp. 308–366. [CrossRef]
3. Gibson, W.C. *The Method of Moments in Electromagnetics*; Taylor & Francis Group: Abingdon, UK, 2008.
4. Vergara, R.C. Development of Geostatistical Models Using Stochastic Partial Differential Equations. Ph.D. Thesis, Université Paris Sciences et Lettres, Paris, France, 2018.
5. Soong, T. *Random Differential Equations in Science and Engineering*, 1st ed.; Academic Press: Cambridge, MA, USA, 1973; Volume 103.
6. Kroese, D.P.; Taimre, T.; Botev, Z. *Handbook of Monte Carlo Methods*; John Wiley & Sons: New York, NY, USA, 2011.
7. Asmussen, S.; Glynn, P. *Stochastic Simulation: Algorithms and Analysis*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007; Volume 57.
8. Casabán, M.C.; Company, R.; Jódar, L. Non-gaussian quadrature integral transform solution of parabolic models with a finite degree of randomness. *Mathematics* **2020**, *8*, 1112. [CrossRef]
9. Davies, B.; Martin, B. Numerical inversion of the laplace transform: A survey and comparison of methods. *J. Comput. Phys.* **1979**, *33*, 1–32. [CrossRef]

10. Stehfest, H. Algorithm 368: Numerical Inversion of Laplace Transforms [D5]. *Commun. ACM* **1970**, *13*, 47–49. [CrossRef]
11. Talbot, A. The Accurate Numerical Inversion of Laplace Transforms. *IMA J. Appl. Math.* **1979**, *23*, 97–120. [CrossRef]
12. Defreitas, C.L.; Kane, S.J. The noise handling properties of the Talbot algorithm for numerically inverting the Laplace transform. *J. Algorithms Comput. Technol.* **2018**, *13*, 1748301818797069. [CrossRef]
13. Iserles, A. On the numerical quadrature of highly-oscillating integrals I: Fourier transforms. *IMA J. Numer. Anal.* **2004**, *24*, 365–391. [CrossRef]
14. Davis, P. J. *Methods of Numerical Integration*; Dover Publications: Mineola, NY, USA, 2007; p. 612.
15. Casabán, M.C.; Company, R.; Egorova, V.N.; Jódar, L. Integral transform solution of random coupled parabolic partial differential models. *Math. Methods Appl. Sci.* **2020**, *43*, 8223–8236. [CrossRef]
16. Casabán, M.C.; Cortés, J.C.; Jódar, L. A random Laplace transform method for solving random mixed parabolic differential problems. *Appl. Math. Comput.* **2015**, *259*, 654–667. [CrossRef]
17. Arnold, L. *Stochastic Differential Equations Theory and Applications*; John Wiley: Hoboken, NJ, USA, 1974.
18. Casabán, M.C.; Company, R.; Jódar, L. Numerical Integral Transform Methods for Random Hyperbolic Models with a Finite Degree of Randomness. *Mathematics* **2019**, *7*, 853. [CrossRef]
19. Iserles, A.; Nørsett, S.P. On Quadrature Methods for Highly Oscillatory Integrals and Their Implementation. *BIT Numer. Math.* **2004**, *44*, 755–772. [CrossRef]
20. Shao, T.S.; Frank, T.C.; Chen, R.M. Tables of zeros and Gaussian weights of certain associated Laguerre polynomials and the related generalized Hermite polynomials. *Math. Comput.* **1964**, *18*, 598–616. [CrossRef]
21. Conte, D.; Ixaru, L.G.; Paternoster, B.; Santomauro, G. Exponentially-fitted Gauss-Laguerre quadrature rule for integrals over an unbounded interval. *J. Comput. Appl. Math.* **2014**, *255*, 725–736. [CrossRef]
22. Abate, J.; Whitt, W. A Unified Framework for Numerically Inverting Laplace Transforms. *INFORMS J. Comput.* **2006**, *18*, 408–421. [CrossRef]
23. Polyanin, A.D.; Nazaikinskii, V.E. *Handbook of Linear Partial Differential Equations for Engineers and Scientists*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2016; p. 1643.
24. Shampine, L. Vectorized adaptive quadrature in MATLAB. *J. Comput. Appl. Math.* **2008**, *211*, 131–140. [CrossRef]