Numerical Investigation of a Class of Nonlinear Time-Dependent Delay PDEs Based on Gaussian Process Regression

Wei Gu, Wenbo Zhang and Yaling Han

School of Statistics and Mathematics, Zhongnan University of Economics and Law, Wuhan 430073, China
* Correspondence: z0001605@zuel.edu.cn

Abstract: Probabilistic machine learning and data-driven methods gradually show their high efficiency in solving the forward and inverse problems of partial differential equations (PDEs). This paper will focus on investigating the forward problem of solving time-dependent nonlinear delay PDEs with multi-delays based on multi-prior numerical Gaussian processes (MP-NGPs), which are constructed by us to solve complex PDEs that may involve fractional operators, multi-delays and different types of boundary conditions. We also quantify the uncertainty of the prediction solution by the posterior distribution of the predicted solution. The core of MP-NGPs is to discretize time firstly, then a Gaussian process regression based on multi-priors is considered at each time step to obtain the solution of the next time step, and this procedure is repeated until the last time step. Different types of boundary conditions are studied in this paper, which include Dirichlet, Neumann and mixed boundary conditions. Several numerical tests are provided to show that the methods considered in this paper work well in solving nonlinear time-dependent PDEs with delay, where delay partial differential equations, delay partial integro-differential equations and delay fractional partial differential equations are considered. Furthermore, in order to improve the accuracy of the algorithm, we construct Runge–Kutta methods under the frame of multi-prior numerical Gaussian processes. The results of the numerical experiments prove that the prediction accuracy of the algorithm is obviously improved when the Runge–Kutta methods are employed.

Keywords: probabilistic machine learning; multi-prior numerical Gaussian processes; nonlinear time-dependent delay partial differential equations; fractional partial differential equations; mixed boundary conditions; Runge–Kutta methods

1. Introduction

In the last decade, more and more attention has been paid on the theoretical studies and practical applications concerned with probabilistic machine learning and data-driven methods [1,2], and some studies have been gradually extended to the fields of differential equations. How to predict the solution of (nonlinear) dynamic systems represented by (delay) partial differential equations (PDEs) has been valued [3–8]. However, (deep) neural networks are favored and used more often by researchers [9–16], and the research targets may involve the forward and inverse problem [17].

Machine learning for Gaussian processes (GPs) [18] is not new, and has been wildly employed in many fields, such as geology [19], meteorology [20] and some financial applications [21]. Similar to other machine learning methodologies, Gaussian processes can be utilized in regression problems or classification problems. Moreover, Gaussian processes belong to a set of methods named kernel machines [22]. However, compared with other kernel machine methods, such as support vector machines (SVMs) [23] and relevance vector machines (RVMs) [24], the strict probability deduction of Gaussian processes greatly limits the popularization of this method in industrial circles. Another drawback of Gaussian processes is that the computational cost may be expensive, due to the Cholesky decomposition
of the kernel functions. As a kernel machine with very strong learning ability, Gaussian processes (GPs) have been widely used in the forward and inverse problem of ordinary differential equations (ODEs) and PDEs in recent years.

Markus et al. (2021) [25] constructed multi-output Gaussian process priors with realizations in the solution set of the linear PDEs; the construction was fully algorithmic via Gröbner bases. However, the construction for building GP priors by the parametrization of such systems was done via a pullback. Paterne et al. (2022) [26] efficiently inferred inhomogeneous terms modeled as GPs by using a truncated basis expansion of the GP kernel from the perspective of exact conjugate Bayesian inference. Mamikon et al. (2022) [27] proposed a framework which combined co-Kriging with the linear transformation of GPs together with the use of kernels given by spectral expansions in eigenfunctions of the boundary value problem, to infer the solution of a well-posed boundary value problem (BVP). To estimate parameters of nonlinear dynamic systems, represented by ODEs from sparse and noisy data, Yang et al. (2022) [28] bypassed the need for numerical integration, and proposed a fast and accurate method, named manifold-constrained Gaussian process inference (MAGI), explicitly conditioned on the manifold constraint that derivatives of Gaussian processes must satisfy the considered ODEs, based on a GP model over time series data. Raissi et al. (2018) [29] proposed a method for the forward problem of time-dependent PDEs based on GPR, which was named numerical Gaussian processes (NGPs). The method can solve equations with high accuracy, and the frame of the algorithm is flexible for solving different problems. Generally speaking, the study of Gaussian processes is an important branch of probabilistic numerics [30–35] in numerical analysis.

On the basis of NGPs, we propose a new method, multi-prior numerical Gaussian processes (MP-NGPs), which is designed to solve complex PDEs. In our known information, most methods based on Gaussian processes do not consider the delay term and high-dimensional problems, and only a few researchers consider fractional derivatives and integro differential derivatives, which obviously cannot meet the requirements of practical application. However, these problems are all considered in MP-NGPs. We also give strict mathematical deduction of MP-NGPs. Several PDEs are solved in the following sections to prove the validity of MP-NGPs.

It is well known that Gaussian processes [18] possess a strict mathematical basis and coincide with the Bayesian method in nature. Unlike traditional methods, including the finite element method [36], spectral method [37], and finite difference method [38], MP-NGPs [29] solve PDEs by means of statistical methods, which is the same as NGPs. In MP-NGPs, the Bayesian method [39] is used to construct an efficient data learning machine, and then infer the solution of the equations.

This paper will mainly exploit the possibility of MP-NGPs to solve the forward problem of a class of time-dependent nonlinear delay partial differential equations with multi-delays (1),

$$\begin{align*}
\frac{\partial u(x,t)}{\partial t} &= g(u,x)\mathcal{L}_x u(x,t) + \sum_{i=1}^{m} h_i(u,x)\mathcal{L}_x^{s_i} u(x,t-s_i) + f(x,t), x \in \Omega, 0 < t < T, \\
u(x,t)|_{x \in \partial \Omega} &= \varphi(x,t), \\
u(x,t) &= \psi(x,t), t \leq 0,
\end{align*}$$

(1)

where $t$ is the temporal variable, $x$ is the $D$ dimensions spatial vector, $s_i (i = 1, 2, \cdots, m)$ are delay terms ($s_m > s_{m-1} > \cdots > s_1 > 0$), and $\Omega \subseteq \mathbb{R}^D$, $\partial \Omega$ is the boundary of $\Omega$. $u(x,t)$ is the solution of (1), $f(x,t)$ is the inhomogeneous term, $g(u,x)$, $h_i(u,x) (i = 1, 2, \cdots, m)$, $\varphi(x,t)$, $\psi(x,t)$ are given functions, and $\mathcal{L}_x, \mathcal{L}_x^{s_i} (i = 1, 2, \cdots, m)$ are linear operators.

The linear operators considered in this paper will contain three different operators; these are partial differential, integro-differential and fractional-order operators. We should point that the fractional-order operators of (1) are considered in the sense of Caputo [40,41]

$$\frac{\partial^\alpha}{\partial x^\alpha} u(x,t) = \frac{1}{\Gamma(n-\alpha)} \int_{-\infty}^{x} \frac{1}{(x-v)^{n+1}} \frac{\partial^n u(v,t)}{\partial v^n} dv,$$

(2)
where \( \alpha > 0 \) is a constant \((n - 1 < \alpha \leq n, n \in \mathbb{N}_+)\), \( \Gamma() \) is the Gamma function.

Apparently, the boundary conditions of the problem (1) are Dirichlet boundary conditions. As for other types of boundary conditions, we will discuss them and give some solutions in Section 4. In order to certify the high efficiency and flexibility of MP-NGPs again, we will consider another type of time-dependent PDEs, wave equations, which still involve delay terms and inhomogeneous terms, in Section 5.

We arrange the outline of this paper as follows: Section 2 introduces MP-NGPs according to the workflow of solving nonlinear delay PDEs. In Section 3, Runge–Kutta methods based on MP-NGPs are constructed. Section 4 introduces the processing of Neumann and mixed boundary conditions in MP-NGPs. Section 5 gives two methods of how to solve wave equations in MP-NGPs. Finally, we present some discussions and conclusions in Sections 6 and 7, respectively.

2. Multi-Priors Numerical Gaussian Processes

In this section, we introduce MP-NGPs based on the workflow of solving nonlinear delay PDEs with multi-delays.

2.1. Prior

Use the backward Euler scheme in the first equation of (1) to discretize time, take the time step size as \( \Delta t \), and then obtain

\[
u^{n-1} = u^n - \Delta t \cdot g(u^n, x) \mathcal{L}_x u^n - \Delta t \cdot \sum_{i=1}^m h_i(u^n, x) \mathcal{L}_x^{\tau_i} u^{n-\tau_i} - \Delta t \cdot f^n, \tag{3}
\]

where \( n = 1, 2, \ldots, N \), \( N = T / \Delta t \), \( u^n = u(x, t^n) \), \( u^{n-\tau_i} = u(x, t^n - s_i) \), \( \tau_i = s_i / \Delta t \), \( f^n = f(x, t^n) \). We approximate the nonlinear terms \( g(u^n, x) \mathcal{L}_x u^n \) and \( h_i(u^n, x) \mathcal{L}_x^{\tau_i} u^{n-\tau_i} \) \((i = 1, 2, \cdots, m)\) with \( g(\mu^{n-1}, x) \mathcal{L}_x u^n \) and \( h_i(\mu^{n-1}, x) \mathcal{L}_x^{\tau_i} u^{n-\tau_i} \) \((i = 1, 2, \cdots, m)\), respectively, where \( \mu^{n-1} \) denotes the predicted solution (posterior mean) of the previous time step. Then introduce notations that \( \mathcal{L}_x^0 u^n = u^n - \Delta t \cdot g(\mu^{n-1}, x) \mathcal{L}_x u^n \), \( \mathcal{L}_x^{\tau_i} u^{n-\tau_i} = -\Delta t \cdot h_i(\mu^{n-1}, x) \mathcal{L}_x^{\tau_i} u^{n-\tau_i} \) \((i = 1, 2, \cdots, m)\) and \( \mathcal{L}_x^{m+1} u^n = -\Delta t \cdot f^n \), so we can obtain the simplified equation

\[
u^{n-1} = \mathcal{L}_x^0 u^n + \sum_{i=1}^m \mathcal{L}_x^{\tau_i} u^{n-\tau_i} + \mathcal{L}_x^{m+1} f^n. \tag{4}
\]

Equation (4) captures the whole structure of solutions at each time step. We can link the solutions at different time steps by setting reasonable prior assumptions. Assume the following prior hypotheses [29]

\[
u^n(x) \sim \mathcal{GP}(0, k_{n,m}(x, x'; \theta_0)), \quad u^{n-\tau_i}(x) \sim \mathcal{GP}(0, k_{n-\tau_i,m}^{n-\tau_i}(x, x'; \theta_i)), \quad i = 1, 2, \cdots, m, \quad f^n(x) \sim \mathcal{GP}(0, k_{f,f}(x, x'; \theta_{m+1})), \tag{5}
\]

to be \( m + 2 \) mutually independent Gaussian processes. Considering the calculation feasibility, we set a prior assumption on \( u^n \) rather than \( u^{n-1} \). Assume covariance functions to be of the following exponential form [18],

\[
k_{n,m}(x, x'; \theta_0) = \sigma_0^2 \exp\left(-\frac{1}{2} \sum_{d=1}^D w_{0,d}(x_d - x'_d)^2\right),
\]

\[
k_{n-\tau_i,m}^{n-\tau_i}(x, x'; \theta_i) = \sigma_i^2 \exp\left(-\frac{1}{2} \sum_{d=1}^D w_{i,d}(x_d - x'_d)^2\right), \quad i = 1, 2, \cdots, m,
\]

\[
k_{f,f}(x, x'; \theta_{m+1}) = \sigma_{m+1}^2 \exp\left(-\frac{1}{2} \sum_{d=1}^D w_{m+1,d}(x_d - x'_d)^2\right), \tag{6}
\]
where \( \{ \theta_j \}_{j=0, 1, \ldots, m+1} \) denotes the hyper-parameters. In Equations (6), \( \theta_j = \left( \frac{\sigma_j^2}{\sigma_n^2} (w_{j,n})^2 \right)_{n=1}^N \), \( j = 0, 1, \ldots, m+1 \). In short, any information of PDEs, such as delay terms, can be encoded into MP-NGPs through assuming multi-priors, as long as the structure constructed by these priors is feasible.

By Mercer’s theorem [42], given a positive definite covariance function \( k(x, x') \) we can decompose it into \( k(x, x') = \sum \lambda_i \varphi_i(x) \varphi_i(x') \), where \( \lambda_i \) and \( \varphi_i \) satisfying \( \int k(x, x') \varphi_i(x) dx = \lambda_i \varphi_i(x) \). Treat \( \{ \sqrt{\lambda_i} \varphi_i \}_{i=1}^\infty \) as a set of orthogonal basis and construct a reproducing kernel Hilbert space (RKHS) [43], which is the so-called kernel trick [43]. The covariance function in Equation (6) does not have a finite decomposition. Different covariance functions, such as rational quadratic covariance functions and Matérn covariance functions, can be appropriately selected under different prior information [18].

2.2. Training

By the properties of GPs [18], we can obtain \( y \sim \mathcal{GP}(0, K) \), where

\[
\begin{align*}
y & = \begin{bmatrix}
u^n_0 \\
u^n_{n-1} \\
u^n_{n-2} \\
\vdots \\
u^n_1 \\
u^n_0 
\end{bmatrix}, \\
K & = \begin{bmatrix}
k_{0,0} & k_{0,1} & \cdots & k_{0,n-2} & k_{0,n-1} \\
k_{1,0} & k_{1,1} & \cdots & k_{1,n-2} & k_{1,n-1} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
k_{n-2,0} & k_{n-2,1} & \cdots & k_{n-2,n-2} & k_{n-2,n-1} \\
k_{n-1,0} & k_{n-1,1} & \cdots & k_{n-1,n-2} & k_{n-1,n-1}
\end{bmatrix},
\end{align*}
\]

\[
k_{i,j} = \int k_{i,j}(x^n_i, x^n_j) + \sigma^2 \mathbb{I} + \sigma^2 \mathbb{I}, \\
k_{i,i} = k_{i,0}(x^n_i, x^n_i) + \sigma^2 \mathbb{I}, \\
k_{0,1} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i,i-1} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i-1,i} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i,i} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i,i} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i,i} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i,i} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i,i} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I}, \\
k_{i,i} = \int k_{i,0}(x^n_i, x^n_j) + \sigma^2 \mathbb{I},
\]

It is worth noting that \( K \) is a symmetry matrix, and elements besides the upper triangular part of \( K \) are not written in this paper.

Consider additive noise in the observed data, and introduce notations that \( u^n_i = u(x^n_i, t^n_i) + \varepsilon_{u_i}, f^n_i = \int f(x^n_i, t^n_i) + \varepsilon_{f_i}, u^n_{n-1} = u(x^n_{n-1}, t^n_{n-1}) + \varepsilon_{u_{n-1}}, \) and \( u^n_{n-1} = u(x^n_{n-1}, t^n_{n-1}) + \varepsilon_{u_{n-1}} \) for \( i = 1, 2, \ldots, m \), where \( \varepsilon_{u_i} \sim \mathcal{N}(0, \sigma^2 \varepsilon_{u_i}), \varepsilon_{f_i} \sim \mathcal{N}(0, \sigma^2 \varepsilon_{f_i}), \varepsilon_{u_{n-1}} \sim \mathcal{N}(0, \sigma^2 \varepsilon_{u_{n-1}}), \) and \( \varepsilon_{u_{n-1}} \sim \mathcal{N}(0, \sigma^2 \varepsilon_{u_{n-1}}) \) for \( i = 1, 2, \ldots, m \). Assume \( \varepsilon_{u_i}, \varepsilon_{f_i}, \varepsilon_{u_{n-1}}, \) and \( \varepsilon_{u_{n-1}} \) to be \( m + 3 \) mutually independent additive noise. Notice that \( k_{i,j}^{n_{i,j}} \) is equal to \( k_{i,j}^{n_{i,j}} \). That is, the boundary points can be seen as the known points uses for the next time step, as Dirichlet boundary conditions are considered. For different types of boundary conditions, the corresponding treatment will be introduced in Section 4.

By (5), the hyper-parameters \( \{ \theta_j \}_{j=0, 1, \ldots, m+1} \) are trained by minimizing the following log marginal likelihood function:

\[
-\log p(y | x^n_0, x^n_{n-1}, \cdots, x^n_{n-1}, \varepsilon_{u_i}, \varepsilon_{f_i}, \varepsilon_{u_{n-1}}, \theta_0, \cdots, \theta_{m+1}) = \frac{1}{2} \log |K| + \frac{1}{2} y^T K^{-1} y + \frac{N}{2} \log 2\pi,
\]

where \( N \) is the length of \( y \). A Quasi-Newton optimizer \( L - BFGS - B \) [44] is applied for training in this paper. It should be pointed that other optimizer algorithm, such as the
whale optimization algorithm [45] (WOA) and the ant colony optimization algorithm [46], can also be used for training.

The training procedure is one of the most important things for exploiting the algorithm, which can be seen as the “regression nature” of MP-NGPs. However, to numerically solve the forward problem of PDEs, the method of MP-NGPs subtly turns algebraic problems into optimization problems, which are more familiar with machine learning methods.

2.3. Posterior

We can easily obtain the following posterior distribution of the prediction solution \( u^n(x^n) \) based on the conditional distribution of Gaussian processes,

\[
\begin{bmatrix}
  u^n_b \\
  u^{n-\tau_1}_b \\
  \vdots \\
  u^{n-\tau_m}_b \\
  f^n
\end{bmatrix} \sim N \left( \begin{bmatrix}
  q^n \\
  q^{n-\tau_1} \\
  \vdots \\
  q^{n-\tau_m} \\
  f^n
\end{bmatrix}, K_{u,u}^{n,n}(x^n_k, x^n_k) - q^n K^{-1} q^n \right),
\]

where \( q^n := [k_{u,b}^{n,n}(x^n_k, x^n_b), k_{u,u}^{n,n-\tau_1}(x^n_k, x^n_k), \ldots, k_{u,u}^{n,n-m}(x^n_k, x^n_k), k_{u,f}^{n,n}(x^n_k, x^n_f), k_{u,u}^{n,n-1}(x^n_k, x^n_k)] \).

2.4. Propagating Uncertainty

Notice that \( \{ x^n_b \} (n > 1) \) and \( \{ x^n_u \} (n > \tau_i, i = 1, \ldots, m) \) are generated data, we cannot use the outcome of the conditional distribution (8) in the next time step. We should propagate the uncertainty firstly, marginalize \( u^{n-1} (u^{n-\tau_i}, n > \tau_i) \) out by exploiting (when \( \tau_i < n - \eta \leq \tau_{i+1} \))

\[
\begin{bmatrix}
  u_1^n \\
  u_2^n \\
  \vdots \\
  u_{p+1}^n \\
  u_1^{n-\tau_1} \\
  u_2^{n-\tau_1} \\
  \vdots \\
  u_{p+1}^{n-\tau_1} \\
  u_1^{n-\tau_m} \\
  u_2^{n-\tau_m} \\
  \vdots \\
  u_{p+1}^{n-\tau_m} \\
  f_1 \\
  f_2 \\
  \vdots \\
  f_{p+1} \\
  f^n \\
  u_0^n \\
  u_1^{n-\tau_1} \\
  u_2^{n-\tau_1} \\
  \vdots \\
  u_{p+1}^{n-\tau_1} \\
  u_1^{n-\tau_m} \\
  u_2^{n-\tau_m} \\
  \vdots \\
  u_{p+1}^{n-\tau_m} \\
  f_1 \\
  f_2 \\
  \vdots \\
  f_{p+1} \\
  u_0^n \\
  u_1^{n-\tau_1} \\
  u_2^{n-\tau_1} \\
  \vdots \\
  u_{p+1}^{n-\tau_1} \\
  u_1^{n-\tau_m} \\
  u_2^{n-\tau_m} \\
  \vdots \\
  u_{p+1}^{n-\tau_m} \\
  f_1 \\
  f_2 \\
  \vdots \\
  f_{p+1} \\
  u_0^n
\end{bmatrix} \sim N(\mu^n, \Sigma^n),
\]

and then have (when \( \tau_p < n \leq \tau_{p+1} \))

\[
\begin{bmatrix}
  u_1^n \\
  u_2^n \\
  \vdots \\
  u_{p+1}^n \\
  u_1^{n-\tau_p} \\
  u_2^{n-\tau_p} \\
  \vdots \\
  u_{p+1}^{n-\tau_p} \\
  u_1^{n-\tau_m} \\
  u_2^{n-\tau_m} \\
  \vdots \\
  u_{p+1}^{n-\tau_m} \\
  f_1 \\
  f_2 \\
  \vdots \\
  f_{p+1} \\
  u_0^n \\
  u_1^{n-\tau_p} \\
  u_2^{n-\tau_p} \\
  \vdots \\
  u_{p+1}^{n-\tau_p} \\
  u_1^{n-\tau_m} \\
  u_2^{n-\tau_m} \\
  \vdots \\
  u_{p+1}^{n-\tau_m} \\
  f_1 \\
  f_2 \\
  \vdots \\
  f_{p+1} \\
  u_0^n
\end{bmatrix} \sim N(\mu^n, \Sigma^n),
\]
1. Firstly, we obtain the initial data where

\[ \mu^n(x^n_0) = q^T K^{-1} \]

\[ u^n_{\theta} \]

\[ \mu^n_{-\gamma_1} \]

\[ \mu^n_{-\gamma_p} \]

\[ u^n_{-\gamma_{p+1}} \]

\[ \ldots \]

\[ u^n_{-\gamma_m} \]

\[ \mu^n_{-1} \]

\[ \Sigma^n(x^n, x'^n) = k^n(x^n, x'^n) - q^T K^{-1} q + q^T K^{-1} S K^{-1} q, \]

and \( S = diag(0, \Sigma^n_{-\gamma_1}, \ldots, \Sigma^n_{-\gamma_p}, \ldots, 0, \ldots, \Sigma^n_{-1, n-1}) \).

Then from (10), we can obtain both the form of \( u^n \sim N(\mu^n, \Sigma^n) \) and the artificial data \( \{x^n_i, u^n\} \) of the next time step, where \( \mu^n = \mu^n(x^n_0), \Sigma^n = \Sigma^n(x^n_i, x^n_i) \).

2.5. Basic Workflow

Summing up the above introduction, we have the following algorithm workflow:

1. Firstly, we obtain the initial data \( \{x^n_0, u^0\} \), delay term data \( \{x^{1-\gamma_i}, u^{1-\gamma_i}\} \) \( i = 1, \ldots, m \) and the inhomogeneous term data \( \{x^1, f^1\} \) by randomly selecting points on spatial domain \( \Omega \), and obtain the boundary data \( \{x^1_0, u^1_0\} \) by randomly selecting points on \( \partial \Omega \). Then we train the hyper-parameters \( \{\theta| j = 0, 1, \ldots, m + 1\} \) in (7) by using the obtained data.

2. From the obtained hyper-parameters of step 1, (8) is used to predict the solution for the first time step and obtain artificially generated data \( \{x^n_1, u^n_1\} \), where points \( x^n_1 \) are randomly sampled from \( \Omega \), and \( u^n_1 \) is the corresponding random vector. We obtain the delay term data \( \{x^{2-\gamma_i}, u^{2-\gamma_i}\} \) \( i = 1, \ldots, m \) and the inhomogeneous term data \( \{x^2, f^2\} \) by randomly selecting points on \( \Omega \), then obtain the boundary data \( \{x^n_2, u^n_2\} \) by randomly selecting points on \( \partial \Omega \).

3. From \( \{x^n_0, u^n\}, \{x^{2-\gamma_i}, u^{2-\gamma_i}\} \) \( i = 1, \ldots, m \), \( \{x^2, f^2\} \) and \( \{x^n_2, u^n_2\} \), we train the hyper-parameters \( \{\theta| j = 0, 1, \ldots, m + 1\} \) for the second time step.

4. From the obtained hyper-parameters of step 3, (10) is used to predict the solution for the second time step and obtain artificially generated data \( \{x^n_3, u^n_3\} \), where points \( x^n_3 \) are randomly sampled from \( \Omega \). We also obtain the delay term data \( \{x^{3-\gamma_i}, u^{3-\gamma_i}\} \) \( i = 1, \ldots, m \) and the inhomogeneous term data \( \{x^3, f^3\} \) by randomly selecting points on \( \Omega \), then obtain the boundary data \( \{x^n_3, u^n_3\} \) by randomly selecting points on \( \partial \Omega \).

5. Steps 3 and 4 are repeated until the last time step.

However, we exploit the Gaussian process regression to predict the next solution at each time step. Figure 1 shows the basic idea of MP-NGPs. At each iteration, the optimized hyper-parameters of the current time step are saved and deemed as initial values of the hyper-parameters in the next time step, which can significantly speed up the training process.
2.6. Processing of Fractional-Order Operators

For simplicity, we consider the following delay problem to show the processing of the fractional-order operators,

$$u_t = g(u, x) \left[ \frac{\partial^{\alpha}}{\partial x^\alpha} u + \frac{\partial^{\beta}}{\partial x^\beta} u \right] + h(u, x) \frac{\partial^2}{\partial x^2} u(x, t - s) + f,$$

where continuous real function $u(x, t)$ is absolutely integrable in $\mathbb{R}^2$. Assume arbitrary-order partial derivatives of $u(x, t)$ to be continuous function and absolutely integrable in $\mathbb{R}^2$.

Consider the backward Euler scheme to (11) with $g(u^n, x)$ and $h(u^n, x)$ replaced by $g(\mu^{n-1}, x)$ and $h(\mu^{n-1}, x)$, respectively,

$$u^{n-1} = u^n - \Delta t \cdot g(\mu^{n-1}, x) \frac{\partial^2}{\partial x^2} u^n - \Delta t \cdot h(\mu^{n-1}, x) \frac{\partial^2}{\partial x^2} u^{n-\tau} - \Delta t \cdot f^n.$$  

From (12), we can obtain

$$u^{n-1} = L^0_x u^n + L^1_x u^{n-\tau} + L^2_x f^n,$$

where $L^0_x u^n = u^n - \Delta t \cdot g(\mu^{n-1}, x) \frac{\partial^2}{\partial x^2} u^n - \Delta t \cdot h(\mu^{n-1}, x) \frac{\partial^2}{\partial x^2} u^{n-\tau}$, and $L^2_x f^n = -\Delta t \cdot f^n$. Assume the Gaussian process prior as follows,

$$u^n(x) \sim \mathcal{GP}(0, k_{0, 0}^u(x, x'; \theta_0)),$$
$$u^{n-\tau}(x) \sim \mathcal{GP}(0, k_{0, \tau}^{n-\tau, n}(x, x'; \theta_1)),$$
$$f^n(x) \sim \mathcal{GP}(0, k_{0, f}^{n, f}(x, x'; \theta_2)).$$

We can easily obtain the following results from Section 2.2,

$$k_{n, n}^{n-1, n-1}(x, x') = L^0_x L^0_x k_{0, 0}^u(x, x') + L^1_x L^1_x k_{0, \tau}^{n-\tau, n-\tau}(x, x') + L^2_x L^2_x k_{0, f}^{n, f}(x, x').$$

How to deal with the first term on the right side of (15) is one of the most important tasks. Denote $k_{n}(x, x') = L^0_x L^0_x k_{0, 0}^u(x, x')$, according to [40], the Fourier transform is carried out with respect to the spatial variable $(x, x')$ on $k_{n}(x, x')$ to obtain $k_{n}^{\omega, \omega'}(\omega, \omega')$, as the excellent properties of the squared exponential covariance function $k_{0, 0}^{\omega, \omega'}(x, x')$ obviously
assure the feasibility of the Fourier transform of derivatives of \( k_{n,n}^{\alpha,n}(x,x') \), according to which \( k_{n,n}^{\alpha,n}(x,x') \) and its partial derivatives vanish for \( \sqrt{x^2 + x'^2} \to +\infty \). Then we can follow the following intermediate kernels:

\[
\begin{align*}
\hat{k}_{n,n}^{\alpha,n-1}(\omega, \omega') &= \left[ 1 - \Delta t \cdot g' \cdot (-i \omega')^\alpha - \Delta t \cdot g' \cdot (-i \omega')^\beta \right] \hat{\mu}_{n,n}^{\alpha,n}(\omega, \omega'), \\
\hat{k}_{n,n}^{\alpha,n}(\omega, \omega') &= \left[ 1 - \Delta t \cdot g \cdot (-i \omega)^\alpha - \Delta t \cdot g \cdot (-i \omega)^\beta \right] \hat{\mu}_{n,n}^{\alpha,n}(\omega, \omega'),
\end{align*}
\]

where \( g = g(n^{-1}(x), x) \), \( g' = g(n^{-1}(x'), x') \).

At last, the inverse Fourier transform can be performed on \( \hat{k}_{n,n}^{\alpha,n-1}(\omega, \omega') \) and \( \hat{k}_{n,n}(\omega, \omega') \) to obtain the kernels \( k_{n,n}^{\alpha,n-1}(x,x') \) and \( k_{n,n}(x,x') \).

2.7. Numerical Tests

To verify the validity of the method proposed, three examples are provided. Firstly, we introduce the maximum absolute error \( E_\infty \) to denote the difference between the conditional posterior mean of the prediction solution and exact solution as follows:

\[
E_\infty(t^n) = \max_{x^*} |u^n(x^*) - \bar{u}^n(x^*)|,
\]

where \( t^n \) is the current time, \( x^* \) is a point random sampling from \( \Omega \), \( u^n(x^*) \) is the exact solution, and \( \bar{u}^n(x^*) \) is the posterior mean of \( u^n(x^*) \) (prediction solution).

We also introduce a notation that \( \max_{1 \leq n \leq N} E_\infty(\Delta t) = \max_{1 \leq n \leq N} E_\infty(t^n) \), which can be seen as a measure of the prediction error under a different time step size \( \Delta t \).

Then we introduce the relative spatial error \( L^2 \) to represent the relative error of prediction [29],

\[
L^2(t^n) = \sqrt{\sum_{x^*} [u^n(x^*) - \bar{u}^n(x^*)]^2} / \sqrt{\sum_{x^*} |u^n(x^*)|^2}.
\]

Example 1.

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \sqrt{\alpha} \sin u \cdot \left[ (1 - x) \frac{\partial}{\partial x} u(x,t) + \sin(2\pi x) \frac{\partial}{\partial x} u(x,t) \right] + u \frac{\partial^2}{\partial x^2} u(x,t - 0.2) + f(x,t), (x,t) \in (0,1) \times (0,1], \\
u(0,t) &= 1 + t^2, u(1,t) = e^{0.5} (1 + t^2), t \in (0,1], \\
u(x,t) &= e^{0.5x} (1 + t^2), -0.2 \leq t \leq 0,
\end{align*}
\]

where \( 0 < \alpha, \beta < 1 \). Example 1 is a fractional partial differential equation with one delay. The fractional partial derivative \( \frac{\partial}{\partial x} u(x,t) \) and \( \frac{\partial}{\partial x} u(x,t) \) is defined by (2) in the Caputo sense [40].

We fix \( (\alpha, \beta) \) to be \((0.25, 0.75), (0.5, 0.5), \) and \((0.75, 0.25)\) in turn. The exact solution of (30) is \( u(x,t) = e^{0.5x} (1 + t^2) \), and the inhomogeneous term \( f(x,t) \) differs from each other when \( (\alpha, \beta) \) is changed,

\[
f(x,t) = \begin{cases} 
q(x,t) = \frac{\sqrt{\pi} \sin \omega(1-x)(1+t^2)}{21(0.75)} f^x_{-\infty} \frac{\omega^2}{(x-p)^{2.5}} \, dv - \frac{\sqrt{\pi} \sin \omega(2\pi x)(1+t^2)}{21(0.75)} f^x_{-\infty} \frac{\omega^2}{(x-p)^{2.5}} \, dv, (\alpha, \beta) = (0.25, 0.75), \\
q(x,t) = \frac{\sqrt{\pi} \sin \omega(1-x)(1+t^2)}{21(0.5)} f^x_{-\infty} \frac{\omega^2}{(x-p)^{2.5}} \, dv - \frac{\sqrt{\pi} \sin \omega(2\pi x)(1+t^2)}{21(0.5)} f^x_{-\infty} \frac{\omega^2}{(x-p)^{2.5}} \, dv, (\alpha, \beta) = (0.5, 0.5), \\
q(x,t) = \frac{\sqrt{\pi} \sin \omega(1-x)(1+t^2)}{21(0.25)} f^x_{-\infty} \frac{\omega^2}{(x-p)^{2.5}} \, dv - \frac{\sqrt{\pi} \sin \omega(2\pi x)(1+t^2)}{21(0.25)} f^x_{-\infty} \frac{\omega^2}{(x-p)^{2.5}} \, dv, (\alpha, \beta) = (0.75, 0.25),
\end{cases}
\]

where \( q(x,t) = 2e^{0.5x} - 0.25(1 + (t - 0.2)^2)(1 + t^2) \).

Numerical tests of solving (19) are performed with noiseless data and noisy data, respectively, where the backward Euler scheme is employed. We only add noise into the initial data \( u^0 \), and the additive noise of \( u^0 \) is \( \epsilon u^0 \sim N(0,0.1^2 I) \). Figures 2–4 show the posterior distribution of the predicted solution at some time nodes, where we fix the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) to be 20 and \( \Delta t \) to be 0.01, and take \( (\alpha, \beta) \) to be \((0.25, 0.75), (0.5, 0.5), \) and \((0.75, 0.25)\) in turn. In each of these figures, (a) presents the results of the
experiments with noiseless data, and (b) presents the results of the experiments with noisy data. Moreover, Table 1 shows us the maximum absolute error $E_{\infty}$ between the conditional posterior mean of the prediction solution and the exact solution at some time nodes in detail, where noiseless data is used in the experiments. Table 2 shows us the maximum absolute error $E_{\infty}$ at some time nodes, where noisy data are used. It is found that the method works well when solving nonlinear delay fractional equations. It is easy to observe that higher prediction accuracy is obtained by using noiseless data in experimental tests.

**Table 1.** Fractional equation of Example 1: maximum absolute error $E_{\infty}$ at some time nodes, where $\Delta t = 0.01$, $(\alpha, \beta)$ is fixed to be $(0.25, 0.75)$, $(0.5, 0.5)$, and $(0.75, 0.25)$ in turn, and noiseless data are used.

<table>
<thead>
<tr>
<th>$t$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\alpha, \beta)$</td>
<td>$(0.25, 0.75)$</td>
<td>$4.006275 \times 10^{-3}$</td>
<td>$4.169444 \times 10^{-3}$</td>
<td>$5.313127 \times 10^{-3}$</td>
<td>$7.771295 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>$(0.5, 0.5)$</td>
<td>$4.289846 \times 10^{-3}$</td>
<td>$5.309005 \times 10^{-3}$</td>
<td>$8.047385 \times 10^{-3}$</td>
<td>$1.130858 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>$(0.75, 0.25)$</td>
<td>$4.731742 \times 10^{-3}$</td>
<td>$6.114796 \times 10^{-3}$</td>
<td>$8.990011 \times 10^{-3}$</td>
<td>$1.248342 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t$</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\alpha, \beta)$</td>
<td>$(0.25, 0.75)$</td>
<td>$1.022259 \times 10^{-2}$</td>
<td>$7.038555 \times 10^{-3}$</td>
<td>$3.247575 \times 10^{-3}$</td>
<td>$5.014752 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>$(0.5, 0.5)$</td>
<td>$1.270620 \times 10^{-2}$</td>
<td>$7.946964 \times 10^{-3}$</td>
<td>$2.464540 \times 10^{-3}$</td>
<td>$7.199783 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>$(0.75, 0.25)$</td>
<td>$1.318964 \times 10^{-2}$</td>
<td>$8.319928 \times 10^{-3}$</td>
<td>$2.440425 \times 10^{-3}$</td>
<td>$6.997903 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

**Table 2.** Fractional equation of Example 1: maximum absolute error $E_{\infty}$ at some time nodes, where $\Delta t = 0.01$, $(\alpha, \beta)$ is fixed to be $(0.25, 0.75)$, $(0.5, 0.5)$, and $(0.75, 0.25)$ in turn, and noisy data are used.

<table>
<thead>
<tr>
<th>$t$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\alpha, \beta)$</td>
<td>$(0.25, 0.75)$</td>
<td>$9.751693 \times 10^{-3}$</td>
<td>$9.933069 \times 10^{-3}$</td>
<td>$7.915402 \times 10^{-3}$</td>
<td>$5.985707 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>$(0.5, 0.5)$</td>
<td>$1.037415 \times 10^{-2}$</td>
<td>$1.226151 \times 10^{-2}$</td>
<td>$1.153429 \times 10^{-2}$</td>
<td>$9.480904 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>$(0.75, 0.25)$</td>
<td>$1.107313 \times 10^{-2}$</td>
<td>$1.362223 \times 10^{-2}$</td>
<td>$1.362442 \times 10^{-2}$</td>
<td>$1.230488 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t$</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\alpha, \beta)$</td>
<td>$(0.25, 0.75)$</td>
<td>$3.930905 \times 10^{-3}$</td>
<td>$4.989465 \times 10^{-3}$</td>
<td>$1.076718 \times 10^{-2}$</td>
<td>$2.762690 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>$(0.5, 0.5)$</td>
<td>$3.722859 \times 10^{-3}$</td>
<td>$4.222247 \times 10^{-3}$</td>
<td>$7.953070 \times 10^{-3}$</td>
<td>$2.115376 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>$(0.75, 0.25)$</td>
<td>$5.542730 \times 10^{-3}$</td>
<td>$3.738703 \times 10^{-3}$</td>
<td>$8.613066 \times 10^{-3}$</td>
<td>$2.476015 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

**Example 2.**

\[
\begin{align*}
\frac{d^\alpha u}{dt^\alpha} &= (u^2 + 1) \left[ \int_0^t u(v, t)dv + \frac{\partial^2 u(x, t)}{\partial x^2} \right] + (u + 1) \frac{\partial^2 u(x, t - 0.2)}{\partial x^2} + f(x, t), (x, t) \in (0, 1) \times (0, 1], \\
u(0, t) &= 2(1 + t^3), u(1, t) = 1 + t^3, t \in [0, 1], \\
u(x, t) &= (2 - x^2)(1 + t^3), -0.2 \leq t \leq 0,
\end{align*}
\]

(21)

Example 2 is a partial integro-differential equation with one delay. The exact solution of (21) is $u(x, t) = (2 - x^2)(1 + t^3)$, and the inhomogeneous term is $f(x, t) = -3t^2(-2 + x^2) + 2[1 + (-0.2 + t^3)[1 - (1 + t^3)(-2 + x^2)] + \frac{1}{3}(1 + t^3)[6 - 6x + x^3][5 - 4x^2 + x^4 + 2t^3(-2 + x^2)^2 + t^6(-2 + x^2)^2]$. Numerical tests of solving (21) are performed with noiseless data and noisy data, respectively, where the backward Euler scheme is employed. We add noise into the initial data $u^0$ and inhomogeneous term data $f^n$ ($n = 1, 2, \cdots, 100$), and the additive noisy of $u^0$ and $f^n$ is $\epsilon_u \sim N(0, 0.2^2 I)$ and $\epsilon_f \sim N(0, 4^2 I)$, respectively. Figure 5 shows the posterior distribution of the predicted solution at different time nodes, where we fix the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) to be 10 and $\Delta t$ to be 0.01. In Figure 5, the five pictures on the upper side present the results of the experiments with noiseless data, and the other five pictures on the lower side present the results of the experiments with noisy data. Moreover, Table 3 shows us the maximum absolute error $E_{\infty}$ at some time nodes in detail, where noiseless
data and noisy data are used in the experiments, respectively. It is found that the method works well when solving nonlinear delay integro-differential equations. It is worth noting that the computational accuracy of the prediction solution is not sensitive to noise, as the level of noise added into the data is not low.

Figure 2. Fractional equation of Example 1: the posterior distribution of the predicted solution at different time nodes, where \((\alpha, \beta)\) is fixed to be \((0.25, 0.75)\). The red solid line represents the exact solution. The blue dash line represents the posterior mean. The green shade indicates the two posterior standard deviation centered on the posterior mean. Noiseless data and noisy data are used, respectively.

Figure 3. Fractional equation of Example 1: the posterior distribution of the predicted solution at different time nodes, where \((\alpha, \beta)\) is fixed to be \((0.5, 0.5)\). Noiseless data and noisy data are used, respectively.
Figure 4. Fractional equation of Example 1: the posterior distribution of the predicted solution at different time nodes, where $(\alpha, \beta)$ is fixed to be $(0.75, 0.25)$. Noiseless data and noisy data are used, respectively.

Figure 5. Integro-differential equation: the posterior distribution of the predicted solution at different time nodes, where $\Delta t = 0.01$. Noiseless data and noisy data are used, respectively.
Example 3.

\[
\begin{align*}
\frac{\partial u}{\partial t} &= (u \sin u + x_1 x_2 x_3) \sum_{i=1}^{3} \frac{\partial^2 u}{\partial x_i^2} \cdot u(x, t - 0.1) + (u x_2 x_3 - 1) \frac{\partial^2}{\partial x_2^2} u(x, t - 0.2) + f(x, t), \\
(x_1, x_2, x_3, t) &\in (0, 2\pi)^3 \times (0, 1), \\
\left.u(x, t)\right|_{x \in \partial(0, 2\pi)^3} &= \frac{1}{2} + \frac{\sin t}{t} - t \in (0, 1], \\
u(x, t) &= \left(\frac{1}{2} + \frac{\sin t}{t}\right) \prod_{i=1}^{3} (\cos x_i + \sin x_i), \quad t \leq 0.
\end{align*}
\] (22)

Example 3 defines a problem with two delays and three spatial dimensions. The exact solution of (22) is \(u(x_1, x_2, x_3, t) = \left(\frac{1}{2} + \frac{\sin t}{t}\right) \prod_{i=1}^{3} (\cos x_i + \sin x_i),\) and the inhomogeneous term is

\[
f(x_1, x_2, x_3, t) = \frac{1}{10} \cos t((\cos x_1 + \sin x_1)(\cos x_2 + \sin x_2)(\cos x_3 + \sin x_3) - \frac{1}{10}(5 - \sin(0.2 - t))(\cos x_1 + \sin x_1) \\
(\cos x_2 - \sin x_2)(\cos x_3 - \sin x_3)(-\frac{1}{10} + \frac{1}{10} x_2 x_3(5 + \sin t))(\cos x_1 + \sin x_1)(\cos x_2 + \sin x_2)(\cos x_3 + \sin x_3)) \\
- \frac{1}{10}(5 - \sin(0.1 - t))(\cos x_1 - \sin x_1)(\cos x_2 - \sin x_2)(\cos x_3 + \sin x_3)(x_1^2 + \frac{1}{10} (5 + \sin t)^2(\cos x_1 + \sin x_1)^2 \\
(\cos x_2 + \cos x_2)^2(\cos x_3 + \sin x_3)^2) - \left(\frac{1}{10}(5 + \sin t)(\cos x_1 + \sin x_1)(\cos x_2 + \sin x_2)\right. \\
\left.(-\cos x_3 - \sin x_3) + \frac{1}{10}(5 + \sin t)(\cos x_1 + \sin x_1)(-\cos x_2 - \sin x_2)(\cos x_3 + \sin x_3) + \frac{1}{10}(5 + \sin t)(-\cos x_1 - \sin x_1)(\cos x_3 + \sin x_3) \\
(\cos x_2 + \sin x_2)(\cos x_3 + \sin x_3)))(x_1 x_2 x_3 + \frac{1}{10}(5 + \sin t)(\cos x_1 + \sin x_1)(\cos x_2 + \sin x_2)(\cos x_3 + \sin x_3)(\frac{1}{10}(5 + \sin t) \\
(\cos x_1 + \sin x_1)(\cos x_2 + \sin x_2)(\cos x_3 + \sin x_3))).
\] (23)

Numerical tests applying the backward Euler scheme to solve (22) will be performed with noiseless data. Figure 6 shows the relative spatial error \(L^2\) and the maximum absolute error \(E_\infty\) with the increase in time, where we fix the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) to be 20 and \(\Delta t\) to be 0.01. The number of training data points on each boundary line is fixed to be 5. Furthermore, Table 4 shows the relative spatial error \(L^2\) and the maximum absolute error \(E_\infty\) at some time nodes in detail. According to the experimental outcome, we can conclude that the method is still effective when solving nonlinear time-dependent delay equations with high dimensions and multi-delays.

### Table 4. Problem of Example 3: Relative spatial error \(L^2\) and maximum absolute error \(E_\infty\) at some time nodes, where \(\Delta t\) is fixed to be 0.01.

<table>
<thead>
<tr>
<th>t</th>
<th>(L^2)</th>
<th>(E_\infty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.255236 \times 10^{-2}</td>
<td>3.240473 \times 10^{-2}</td>
</tr>
<tr>
<td>0.2</td>
<td>1.287883 \times 10^{-2}</td>
<td>3.420406 \times 10^{-2}</td>
</tr>
<tr>
<td>0.3</td>
<td>1.319996 \times 10^{-2}</td>
<td>3.573738 \times 10^{-2}</td>
</tr>
<tr>
<td>0.4</td>
<td>1.311049 \times 10^{-2}</td>
<td>3.626811 \times 10^{-2}</td>
</tr>
<tr>
<td>0.5</td>
<td>1.311049 \times 10^{-2}</td>
<td>3.617718 \times 10^{-2}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t</th>
<th>(L^2)</th>
<th>(E_\infty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>1.339241 \times 10^{-2}</td>
<td>3.835969 \times 10^{-2}</td>
</tr>
<tr>
<td>0.7</td>
<td>1.301079 \times 10^{-2}</td>
<td>3.759043 \times 10^{-2}</td>
</tr>
<tr>
<td>0.8</td>
<td>1.289006 \times 10^{-2}</td>
<td>3.745873 \times 10^{-2}</td>
</tr>
<tr>
<td>0.9</td>
<td>1.318331 \times 10^{-2}</td>
<td>3.913333 \times 10^{-2}</td>
</tr>
<tr>
<td>1.0</td>
<td>1.341626 \times 10^{-2}</td>
<td>4.018637 \times 10^{-2}</td>
</tr>
</tbody>
</table>
3. Runge–Kutta Methods

For the purpose of improving the accuracy of the methodology considered in Section 2, we will consider Runge–Kutta methods [47,48] based on the frame of MP-NGPs.

3.1. MP-NGPs Based on Runge–Kutta Methods

Apply Runge–Kutta methods with q stages to the first equation of (1) with appropriate approximation of nonlinear terms, and we can obtain [47]

\[
\frac{\mu_{n+1} - \mu_n}{\Delta t} = \sum_{j=1}^{q} b_j \left[ g(\mu^n, x) L_2 \mu^n + \sum_{k=1}^{m} h_k(\mu^n, x) L_2^k \mu^n \right] + f^{n+\zeta_j},
\]

\[
\frac{\mu^{n+\zeta_j} - \mu_n}{\Delta t} = \sum_{j=1}^{q} a_{ij} \left[ g(\mu^n, x) L_2 \mu^n + \sum_{k=1}^{m} h_k(\mu^n, x) L_2^k \mu^n \right] + f^{n+\zeta_j},
\]

Reforming (24), we have

\[
u^n_{q+1} := u^n = u^{n+1} - \Delta t \sum_{j=1}^{q} b_j \left[ g(\mu^n, x) L_2 \mu^n + \sum_{k=1}^{m} h_k(\mu^n, x) L_2^k \mu^n \right] + f^{n+\zeta_j},
\]

\[
u^n_i := u^n = u^{n+\zeta_j} - \Delta t \sum_{j=1}^{q} a_{ij} \left[ g(\mu^n, x) L_2 \mu^n + \sum_{k=1}^{m} h_k(\mu^n, x) L_2^k \mu^n \right] + f^{n+\zeta_j},
\]

where \(u^{n+\zeta_j} = u(x, t^n + \zeta_j \Delta t), u^{n+\zeta_j-\tau} = u(x, t^n + \zeta_j \Delta t - \tau),\) and \(f^{n+\zeta_j} = f(x, t^n + \zeta_j \Delta t).\) Prior assumptions are set on \(u^{n+1}, u^{n+\zeta_j}, f^{n+\zeta_j}, u^{n+\zeta_j-\tau}, u^{n+1}_q, u^n_q (i = 1, 2, \ldots, q; k = 1, 2, \ldots, m).\) For simplicity, we introduce the Runge–Kutta methods by solving the following problem:

\[
u_1 = g(u, x) L_2 u + h(u, x) L_2^1 u(x, t - s) + f.
\]

The trapezoidal rule is applied under the structure of Runge–Kutta methods to (26) with \(g(u^{n+1}, x), h(u^{n+1}, x), g(u^n, x)\) and \(h(u^n, x)\) replaced by \(g(\mu^n, x), h(\mu^n, x), g(\mu^n, x)\) and \(h(\mu^n, x),\) respectively. Then we have

\[
u^{n+1} - \nu_n = \frac{1}{2} \left[ g(\mu^n, x) L_2 \nu^{n+1} + h(\mu^n, x) L_2^1 \nu^{n+1-\tau} + f^{n+1} \right] + \frac{1}{2} \left[ g(\mu^n, x) L_2 \nu^n + h(\mu^n, x) L_2^1 \nu^{n-\tau} + f^n \right].
\]
Reforming (27), we have
\[ u_1^n := u^n, \]
\[ u_0^n := u_3^n - \frac{1}{2} \Delta t [g(u^n, x) L x u^{n+1} + h(u^n, x) L x u^{n+1} - f^n + f^{n+1}] \] (28)

However, Equation (28) coincides with the structure of Runge–Kutta methods when \( q = 2, \xi_1 = 0, \xi_2 = 1, b_1 = b_2 = 1/2, a_{11} = a_{12} = 0, \) and \( a_{21} = a_{22} = 1/2. \) Reforming (28), we obtain
\[ u_1^n := u^n, \]
\[ u_2^n := u_3^n - \frac{1}{2} \Delta t [g(u^n, x) L x u^{n+1} + h(u^n, x) L x u^{n+1} - f^n + f^{n+1}] \] (29)
where \( L x, L x_1, L x_2 \) and \( L x_3 \) are linear operators.

Assume the following prior hypotheses:
\[ u^n(x) \sim \mathcal{GP}(0, k^{n+1, n+1}_{u,u}(x, x'; \theta_1)), \]
\[ u^{n+1}(x) \sim \mathcal{GP}(0, k^{n+1, n+1}_{u,u}(x, x'; \theta_1)), \]
\[ u^{n+1}(x) \sim \mathcal{GP}(0, k^{n+1, n+1}_{u,u}(x, x'; \theta_1)), \]
\[ f^n(x) \sim \mathcal{GP}(0, k^{n+1, n+1}_{f,f}(x, x'; \theta_2)), \]
\[ f^{n+1}(x) \sim \mathcal{GP}(0, k^{n+1, n+1}_{f,f}(x, x'; \theta_2)), \]
(30)
to be mutually independent Gaussian processes, and assume covariance functions to be the following exponential form of (6). By the properties of the Gaussian processes, we have
\[ y \sim \mathcal{GP}(0, K), \]
where
\[ K = \begin{bmatrix}
    k^{n+1, n+1}_{u,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{u,u}
\end{bmatrix}, \]
(31)

\[ \begin{bmatrix}
    k^{n+1, n+1}_{u,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} \\
    k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{u,u}
\end{bmatrix}^{-1} \begin{bmatrix}
    u^{n+1}_1 \\
    u^{n+1}_2 \\
    u^{n+1}_3 \\
    f^{n+1}_1 \\
    f^{n+1}_2 \\
    f^{n+1}_3
\end{bmatrix} \]

\[ \begin{bmatrix}
    k^{n+1, n+1}_{u,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{u,u} & k^{n+1, n+1}_{k,b} \\
    k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{u,u} \\
    k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{k,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{h,b} & k^{n+1, n+1}_{u,u}
\end{bmatrix}^{-1} \begin{bmatrix}
    u^{n+1}_1 \\
    u^{n+1}_2 \\
    u^{n+1}_3 \\
    f^{n+1}_1 \\
    f^{n+1}_2 \\
    f^{n+1}_3
\end{bmatrix} \]

Other kernels are assumed to be 0.
We can write the posterior distribution of the prediction \( u^{n+1}(x_{n+1}) \) as follows:
\[
\begin{align*}
\mathbf{u}^{n+1}(x_{n+1}^+)^m & \sim \mathcal{N} \left( \mathbf{q}^T \mathbf{K}^{-1} \mathbf{u}^n, \mathbf{f}^n \right), \\
\mathbf{u}^{n+1}(x_{n+1}^-) & = \mathcal{L}(\mathbf{q}^T \mathbf{K}^{-1} \mathbf{u}^n, \mathbf{f}^n).
\end{align*}
\]

where \( \mathbf{q}^T := \left[ k_{u,u}^{n+1,1}(x_{n+1}^+, x_{n+1}^-), k_{u,u}^{n+1,1-\tau}(x_{n+1}^+, x_{n+1}^-), k_{u,u}^{n+1,1}(x_{n+1}^+, x_{n+1}^-) \right] \), and \( k_{u,u}^{n+1,1}(x_{n+1}^+, x_{n+1}^-) \) are artificially generated data, we cannot directly use the outcome of the conditional distribution of (32) in the next time step. To propagate the uncertainty, we should marginalize \( \mathbf{u}^{n-\tau}, \mathbf{u}^{n+1-\tau} \) by employing

\[
\begin{align*}
\mathbf{u}^{n+1-\eta} & \sim \mathcal{N} \left( \mathbf{\mu}^{n+1-\eta}, \mathbf{\Sigma}^{n+1-\eta, n+1-\eta} \right), n + 1 - \eta \leq \tau, \\
\mathbf{u}^{n+1-\eta} & \sim \mathcal{N} \left( \mathbf{\mu}^{n+1-\eta}, \mathbf{\Sigma}^{n+1-\eta, n+1-\eta} \right), n + 1 - \eta \geq \tau.
\end{align*}
\]

to get

\[
\begin{align*}
\mathbf{u}^{n+1}(x_{n+1}^+)^m & \sim \mathcal{N} \left( \mathbf{\mu}^{n+1}(x_{n+1}^+), \mathbf{\Sigma}^{n+1,n+1}(x_{n+1}^+, x_{n+1}^-) \right), n + 1 \leq \tau, \\
\mathbf{u}^{n+1}(x_{n+1}^-) & \sim \mathcal{N} \left( \mathbf{\mu}^{n+1}(x_{n+1}^-), \mathbf{\Sigma}^{n+1,n+1}(x_{n+1}^+, x_{n+1}^-) \right), n + 1 \geq \tau.
\end{align*}
\]
where \( \mu^{n+1}(x^{n+1}) = q^T K^{-1} \begin{bmatrix} u^{n+1}_h \\ \mu^{n+1}_{\mu-\tau} \\ f^{n+1} \\ u^{n}_h \\ \mu^{n}_{\mu-\tau} \\ f^n \\ u^n_h \\ \mu^n_{\mu-\tau} \\ f^n \\
\end{bmatrix} \) (if \( n + 1 - \tau < 0 \), replace \( \mu^{n+1-\tau} \) by \( u^{n+1-\tau} \), if \( n - \tau < 0 \), replace \( \mu^{n-\tau} \) by \( u^{n-\tau} \)), \( \Sigma^{n+1,n+1}(x^{n+1}_1, x^{n+1}_2) = k^{n+1,n+1}_h(x^{n+1}_1, x^{n+1}_2) - q^T K^{-1} q + q^T K^{-1} S K^{-1} q, S = \text{diag}(0, \Sigma^{n+1,n+1-\tau}, 0, 0, \Sigma^{n-\tau,n-\tau}, 0, \Sigma^{n,n+1}_2)(\text{if } n + 1 - \tau < 0, \text{replace } \Sigma^{n+1,n+1-\tau} \text{ by } 0; \text{if } n - \tau < 0, \text{replace } \Sigma^{n-\tau,n-\tau} \text{ by } 0) \), and \( \Sigma^{n,n}_2 = \begin{bmatrix} \Sigma^{n,n}_{11} & \Sigma^{n,n}_{12} \\ \Sigma^{n,n}_{12} & \Sigma^{n,n}_{22} \end{bmatrix} \).

On the basis of Equation (34), we have \( u^{n+1} \sim N(\mu^{n+1}, \Sigma^{n+1,n+1}) \), and obtain the artificial data \( \{x^{n+1}_a, u^{n+1}\} \) of the next step, where \( \mu^{n+1} = \mu^{n+1}(x^{n+1}_a), \Sigma^{n+1,n+1} = \Sigma^{n+1,n+1}(x^{n+1}_a, x^{n+1}_a) \).

### 3.2. Numerical Test for the Problem with One Delay Applying the Trapezoidal Rule

In this subsection, the method of applying the backward Euler scheme and the trapezoidal scheme is applied to solve Example 4, and numerical tests are carried out. Moreover, the impact of data noise and the amount of training data on maximum absolute error \( E_{\infty} \) is investigated in this subsection.

**Example 4.**

\[
\begin{align*}
\frac{\partial u}{\partial t} &= (u^2 + \sin u + 2x + 1) \frac{\partial^2 u}{\partial x^2} + (ux) \frac{\partial u}{\partial x} + f(x,t) + (x,t) \in (0,2\pi) \times (0,1], \\
u(0,t) &= e^{-t}, u(2\pi,t) = e^{-t}, t \in (0,1] \\
u(x,0) &= e^{-t}(\cos x + \sin x), t \leq 0.
\end{align*}
\]

The exact solution of (35) is \( u(x,t) = e^{-t}(\cos x + \sin x) \), and the inhomogeneous term is \( f(x,t) = e^{-t}(\cos x + \sin x) \{2x + e^{-2t}(\cos x + \sin x)^2 + \sin[e^{-t}(\cos x + \sin x)]\} - e^{0.2-2t}x \cos 2x \).

Table 5 presents the maximum absolute error \( E_{\infty} \) between the conditional posterior mean of the prediction solution and the exact solution at five time nodes for the different time step size \( \Delta t \) under the backward Euler scheme, where the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) is fixed to be 20. It is found that the method applying the backward Euler scheme is first-order accurate in time, which is just in line with the first-order convergence property of the backward Euler scheme we applied. Since calculating the convergence order in time requires the maximum absolute error under two consecutive time steps, we omit the display of the first value in the last column of Tables 5 and 6.

Figure 7a shows the impact of the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) on \( \max E_{\infty}(\Delta t) = \max_{1 \leq n \leq N} E_{\infty}(t^n) \), where we take \( \Delta t = 10^{-1}, 10^{-2} \) and \( 10^{-3} \) in turn. The results show that when \( \Delta t \) is equal to \( 10^{-1} \), the prediction accuracy quickly reaches saturation with the increase in the number of training data points, but when \( \Delta t \) is taken smaller, increasing the amount of training data still has the potential to improve the prediction accuracy, and the saturation behavior occurs much later.

Moreover, we investigate the impact of noise on the prediction accuracy. We add noise into the initial data \( u^0 \), delay term data \( u^{-\tau} \) \( (\tau = 1, 2, \cdots, 19) \) and inhomogeneous term data \( f^n \) \( (n = 1, 2, \cdots, 100) \) to investigate the impact of the level of the noise on the accuracy. The additive noisy of \( u^0, u^{-\tau} \) and \( f^n \) is \( \varepsilon_{u^0} \sim N(0, \sigma^2 I), \varepsilon_{u^{-\tau}} \sim N(0, \sigma^2 I) \) and \( \varepsilon_{f^n} \sim N(0, \frac{1}{2} \sigma^2 I) \), respectively. Take \( t = 0.2, 0.6, 1.0 \), and observe the change of \( E_{\infty} \) under different \( \sigma \) at these time nodes (see Figure 7b). The experiment outcome proves that the
bigger $\sigma$, the lower the prediction accuracy. With the progress of the iteration, the impact of the noise on the accuracy of the current time step becomes smaller, and the accuracy gradually tends to increase, as the hyper-parameters that contain the knowledge of the considered problem are trained better and better as the iteration progress continues.

Table 6 shows the maximum absolute error $E_\infty$ between the conditional posterior mean of the prediction solution and the exact solution at five time nodes for the different time step size $\Delta t$ under the trapezoidal scheme, where the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) is fixed to be 20. It is found that the method applying the trapezoidal rule is still first-order accurate in time, which is apparently not consistent with the second-order convergence property of the trapezoidal rule. Therefore, we can conclude from these results that the convergence rate of MP-NGPs using Runge–Kutta methods is also the first-order. Fortunately, the accuracy of the method applying the trapezoidal scheme (Runge–Kutta methods) does exceed the backward Euler scheme when other experimental conditions are fixed, which means that applying the Runge–Kutta methods on MP-NGPs is still very useful when the current accuracy is not satisfying. The results of Figure 8 also prove that the Runge–Kutta methods do exceed the backward Euler scheme in prediction accuracy.

![Figure 7](image)

**Figure 7.** Results of Example 4, (a) impact of the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) on $\max E_\infty(\Delta t) = \max \max E_\infty(t^n)$, when the backward Euler scheme is employed. Fix $\Delta t$ to be $10^{-1}$, $10^{-2}$ and $10^{-3}$ in turn. (b) Impact of the level of the additive noise on the maximum absolute error $E_\infty$, when the backward Euler scheme is used. Fix the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) to be 20, and take $\Delta t = 0.01$.

Table 5. Problem of Example 4: maximum absolute error $E_\infty$ at five time nodes for different $\Delta t$ under the backward Euler scheme and the convergence order in time.

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>$t = 0.2$</th>
<th>$t = 0.4$</th>
<th>$t = 0.6$</th>
<th>$t = 0.8$</th>
<th>$t = 1.0$</th>
<th>$\max E_\infty(\Delta t)$</th>
<th>$\max E_\infty(2\Delta t)$</th>
<th>$\max E_\infty(3\Delta t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$5.196883 \times 10^{-2}$</td>
<td>$4.358087 \times 10^{-2}$</td>
<td>$3.27721 \times 10^{-2}$</td>
<td>$2.416694 \times 10^{-2}$</td>
<td>$1.781159 \times 10^{-2}$</td>
<td>$5.196883 \times 10^{-2}$</td>
<td>$1.966$</td>
<td>$1.966$</td>
</tr>
<tr>
<td>0.1</td>
<td>$2.643025 \times 10^{-2}$</td>
<td>$2.202658 \times 10^{-2}$</td>
<td>$1.643945 \times 10^{-2}$</td>
<td>$1.206984 \times 10^{-2}$</td>
<td>$8.92375 \times 10^{-3}$</td>
<td>$2.643025 \times 10^{-2}$</td>
<td>$1.966$</td>
<td>$1.966$</td>
</tr>
<tr>
<td>0.05</td>
<td>$1.354968 \times 10^{-2}$</td>
<td>$1.112114 \times 10^{-2}$</td>
<td>$8.246767 \times 10^{-3}$</td>
<td>$6.030003 \times 10^{-3}$</td>
<td>$4.610772 \times 10^{-3}$</td>
<td>$1.354968 \times 10^{-2}$</td>
<td>$1.951$</td>
<td>$1.951$</td>
</tr>
<tr>
<td>0.025</td>
<td>$6.893477 \times 10^{-3}$</td>
<td>$5.593641 \times 10^{-3}$</td>
<td>$4.121273 \times 10^{-3}$</td>
<td>$3.001243 \times 10^{-3}$</td>
<td>$2.359743 \times 10^{-3}$</td>
<td>$6.893477 \times 10^{-3}$</td>
<td>$1.966$</td>
<td>$1.966$</td>
</tr>
<tr>
<td>0.0125</td>
<td>$3.459075 \times 10^{-3}$</td>
<td>$2.800549 \times 10^{-3}$</td>
<td>$2.065297 \times 10^{-3}$</td>
<td>$1.507394 \times 10^{-3}$</td>
<td>$1.192749 \times 10^{-3}$</td>
<td>$3.459075 \times 10^{-3}$</td>
<td>$1.991$</td>
<td>$1.991$</td>
</tr>
<tr>
<td>0.00825</td>
<td>$1.753664 \times 10^{-3}$</td>
<td>$1.414149 \times 10^{-3}$</td>
<td>$1.036199 \times 10^{-3}$</td>
<td>$7.571422 \times 10^{-4}$</td>
<td>$5.988719 \times 10^{-4}$</td>
<td>$1.763992 \times 10^{-3}$</td>
<td>$1.963$</td>
<td>$1.963$</td>
</tr>
</tbody>
</table>
Table 6. Problem of Example 4: maximum absolute error $E_{\infty}$ at five time nodes for different $\Delta t$ under the trapezoidal scheme and the convergence order in time.

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>$t = 0.2$</th>
<th>$t = 0.4$</th>
<th>$t = 0.6$</th>
<th>$t = 0.8$</th>
<th>$t = 1.0$</th>
<th>$\max E_{\infty}(\Delta t)$</th>
<th>$\max E_{\infty}(2\Delta t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$4.495057 \times 10^{-2}$</td>
<td>$1.990879 \times 10^{-2}$</td>
<td>$1.849233 \times 10^{-2}$</td>
<td>$1.384659 \times 10^{-2}$</td>
<td>$1.045600 \times 10^{-2}$</td>
<td>$4.495057 \times 10^{-2}$</td>
<td>0.4990879</td>
</tr>
<tr>
<td>0.1</td>
<td>$1.63829 \times 10^{-2}$</td>
<td>$1.275052 \times 10^{-2}$</td>
<td>$9.718359 \times 10^{-3}$</td>
<td>$7.421787 \times 10^{-3}$</td>
<td>$5.774761 \times 10^{-3}$</td>
<td>$1.747229 \times 10^{-2}$</td>
<td>0.275052</td>
</tr>
<tr>
<td>0.05</td>
<td>$8.15066 \times 10^{-3}$</td>
<td>$6.595744 \times 10^{-3}$</td>
<td>$5.033765 \times 10^{-3}$</td>
<td>$3.856745 \times 10^{-3}$</td>
<td>$2.79750 \times 10^{-3}$</td>
<td>$7.184114 \times 10^{-3}$</td>
<td>0.5033765</td>
</tr>
<tr>
<td>0.025</td>
<td>$4.079031 \times 10^{-3}$</td>
<td>$3.540249 \times 10^{-3}$</td>
<td>$2.562043 \times 10^{-3}$</td>
<td>$1.969803 \times 10^{-3}$</td>
<td>$1.528144 \times 10^{-3}$</td>
<td>$4.088401 \times 10^{-3}$</td>
<td>0.2562043</td>
</tr>
<tr>
<td>0.0125</td>
<td>$2.029881 \times 10^{-3}$</td>
<td>$1.701985 \times 10^{-3}$</td>
<td>$1.300908 \times 10^{-3}$</td>
<td>$9.986143 \times 10^{-4}$</td>
<td>$7.749402 \times 10^{-4}$</td>
<td>$2.043320 \times 10^{-3}$</td>
<td>0.1701985</td>
</tr>
<tr>
<td>0.00625</td>
<td>$1.036153 \times 10^{-3}$</td>
<td>$8.602172 \times 10^{-4}$</td>
<td>$6.599498 \times 10^{-4}$</td>
<td>$5.096151 \times 10^{-4}$</td>
<td>$3.956461 \times 10^{-4}$</td>
<td>$1.040213 \times 10^{-3}$</td>
<td>0.0860217</td>
</tr>
</tbody>
</table>

Figure 8. Problem of Example 1: (a) The posterior distribution of the predicted solution at different time nodes, where the backward Euler scheme is used. Fix the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) to be 30, and take $\Delta t = 0.01$. (b) The posterior distribution of the predicted solution at different time nodes, where the trapezoidal scheme is used. The other experimental conditions are the same as those with the backward Euler scheme. (c) Time versus maximum absolute error $E_{\infty}$ until the final time step is reached, where the backward Euler scheme and the trapezoidal scheme are both considered.

4. Processing of Neumann and Mixed Boundary Conditions

In the above sections, we did not mention the processing of other boundary conditions, except Dirichlet boundary conditions. This section will introduce how to solve two-dimensional PDEs with Neumann or mixed boundary conditions. That is, the spatial vector $x$ is a one-dimensional variable.
As everyone knows, Dirichlet, Neumann and mixed boundary conditions have the following general form, respectively, in the problem (1) of two dimensions:

\[ u(lb, t) = \phi_1(t), \quad u(ub, t) = \phi_2(t), \quad 0 < t \leq T, \]  

\[ \left[ \frac{\partial u}{\partial x} \right]_{x=lb} = \psi_1(t), \quad \left[ \frac{\partial u}{\partial x} \right]_{x=ub} = \psi_2(t), \quad 0 < t \leq T, \]  

\[ \left[ \lambda_{11}(t) \frac{\partial u}{\partial x} + \lambda_{12}(t) u \right]_{x=lb} = \xi_1(t), \quad \left[ \lambda_{21}(t) \frac{\partial u}{\partial x} + \lambda_{22}(t) u \right]_{x=ub} = \xi_2(t), \quad 0 < t \leq T, \]  

(36)  
(37)  
(38)

where \( \phi_1(t), \phi_2(t), \psi_1(t), \psi_2(t), \lambda_{11}(t), \lambda_{12}(t), \lambda_{21}(t), \lambda_{22}(t), \xi_1(t), \xi_2(t) \) are given functions.

In order to avoid the repetitive introduction, we will focus on the processing of Neumann and mixed boundary conditions in this section.

4.1. Neumann Boundary Conditions

For the Neumann boundary conditions (37), introduce \( v^n = \frac{\partial u^n}{\partial x} \), where \( v^n(x) = v(x, t^n) = \frac{\partial u^n}{\partial x} \). Note \( \mathcal{L}_{x}^u \mathcal{L}_{x}^u = \frac{\partial u^n}{\partial x} \), and \( \mathcal{L}_{x}^u \) is a linear operator.

Return to the problem (1), then we can get \( y \sim \mathcal{G}(0, K) \), where

\[
y = \begin{bmatrix}
    v^n_b \\
v^n - \tau_1 \\
  \vdots \\
v^n - \tau_w \\
f^n \\
u^n - 1
\end{bmatrix},
\]

\[
K = \begin{bmatrix}
k^{n,n}_{v,v,l} & k^{n,n}_{v,v,t_1} & \cdots & k^{n,n}_{v,v,t_w} & k^{n,n}_{v,f} & k^{n,n-1}_{v,f} \\
k^{n,n-1}_{u,u,l} & k^{n,n-1}_{u,u,t_1} & \cdots & k^{n,n-1}_{u,u,t_w} & k^{n,n-1}_{u,u,t_1} & k^{n,n-1}_{u,u,t_2} \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
k^{n,n-1}_{f,f,l} & k^{n,n-1}_{f,f,t_1} & \cdots & k^{n,n-1}_{f,f,t_w} & k^{n,n-1}_{f,f} & k^{n,n-1}_{f,f}
\end{bmatrix},
\]

\[
k^{n,n}_{v,v,l} = \mathcal{L}_{x}^v \mathcal{L}_{x,x}^u \mathcal{L}_{x,xx}^u (x^n_b, x^n_u) + \sigma_{v,v}^2 I, \quad k^{n,n}_{f,f} = k^{n,n}_{f,f} (x^n_f, x^n_f) + \sigma_{f,f}^2 I, \quad x^n_b = \{lb, ub\},
\]

\[
k^{n,n-1}_{u,u,m} = k^{n,n-1}_{u,u,m} (x^n_u, x^n_u), \quad \sigma_{u,u}^2 = 2 \sum_{k=1}^{m} \sigma_{u,u}^2, \quad K = k^{n,n-1}_{u,u,m} + \sigma_{u,u}^2 I, \quad K = k^{n,n-1}_{u,u,m} + \sigma_{u,u}^2 I, \quad K = k^{n,n-1}_{u,u,m} + \sigma_{u,u}^2 I.
\]

(36)  
(37)  
(38)

where \( v^n_b = v(x^n_b, t^n) + \varepsilon_{v,b} \), \( \varepsilon_{v,b} \sim N(0, \sigma_{v,b}^2 I) \). The additive noise \( \varepsilon_{v,b} \) is assumed to be independent of other noise.

By (5), train the hyper-parameters \( \{\theta_j\}_{j=0, 1, \cdots, m+1} \) by minimizing NLML (39),

\[
- \log p(y|x^n_b, x^n_{u-\tau_1}, \cdots, x^n_{u-\tau_w}, x^n_{u-1}, \theta_0, \cdots, \theta_{m+1}) = \frac{1}{2} \log |K| + \frac{1}{2} y^T K^{-1} y + \frac{N}{2} \log 2 \pi,
\]

(39)

After the training part, the posterior distribution of \( u^n(x^n_u) \) can be directly written as

\[
u^n(x^n_u) \sim N \left( \begin{bmatrix}
v^n_b \\
  \vdots \\
v^n - \tau_w \\
f^n \\
u^n - 1
\end{bmatrix},
\left( k^{n,n}_{u,u,m} (x^n_u, x^n_u) - q^T K^{-1} q \right) \right),
\]

(40)
where \( q^T := \{ k^{n,T}_{n,p}(x^n_{s}, x^n_{b}), k^{n,T}_{n,T1}(x^n_{s}, x^n_{u}), \ldots, k^{n,T}_{n,Tm}(x^n_{s}, x^n_{u}), k^{n,T}_{n,T}(x^n_{s}, x^n_{f}) \}, \)

\( k^{n-1}_{n,p}(x^n_{s}, x^n_{u}) \), \( k^{n}_{n,p}(x^n_{s}, x^n_{b}) = C^n_{x,n} k^{n}_{n,p}(x^n_{s}, x^n_{b}) \).

On the basis of Section 2.4, in order to accurately propagate the uncertainty, marginalize \( u^{n-1} \) (and perhaps \( u^{n-T_i} \) when \( n > T_i \)) by employing (assume \( T_q < n - \eta \leq T_{q+1} \))

\[
\begin{aligned}
\begin{bmatrix}
\psi_1^n & \psi_2^n & \ldots & \psi_b^n \\
\mu_1^{n-T_1} & \mu_2^{n-T_1} & \ldots & \mu_0^n \\
\vdots & \ddots & \ddots & \vdots \\
\mu_1^{n-T_p} & \mu_2^{n-T_p} & \ldots & \mu_0^n \\
\mu_1^{n-T} & \mu_2^{n-T} & \ldots & \mu_0^n \\
\mu_1^{n} & \mu_2^{n} & \ldots & \mu_0^n \\
\end{bmatrix}

\begin{bmatrix}
\n \vdots \\
\n \vdots \\
\n \vdots \\
\n \vdots \\
\n \vdots \\
\n \vdots \\
\end{bmatrix}
\end{aligned}
\]

\[
\begin{aligned}
\begin{bmatrix}
\psi_1^n & \psi_2^n & \ldots & \psi_b^n \\
\mu_1^{n-T_1} & \mu_2^{n-T_1} & \ldots & \mu_0^n \\
\vdots & \ddots & \ddots & \vdots \\
\mu_1^{n-T_p} & \mu_2^{n-T_p} & \ldots & \mu_0^n \\
\mu_1^{n-T} & \mu_2^{n-T} & \ldots & \mu_0^n \\
\mu_1^{n} & \mu_2^{n} & \ldots & \mu_0^n \\
\end{bmatrix}

\begin{bmatrix}
\n \vdots \\
\n \vdots \\
\n \vdots \\
\n \vdots \\
\n \vdots \\
\n \vdots \\
\end{bmatrix}
\end{aligned}
\]

where \( \mu^n(x^n_{s}) = q^T K^{-1} \), \( \Sigma^n(x^n_{s}) = k^{n}_{n,p}(x^n_{s}, x^n_{b}) - q^T K^{-1} q + q^T K^{-1} S K^{-1} q \),

and \( S = \text{diag}(0, \Sigma^{n,1}, \Sigma^{n,1}, \ldots, 0, \ldots, \Sigma^{n,1}) \).

By the posterior distribution (42), obtain the simplified form \( u^n \sim N(\mu^n, \Sigma^n) \), and then obtain the artificial data \( \{x^n_{b}, u^n\} \) of the next time step, where \( \mu^n = \mu^n(x^n_{b}) \), and \( \Sigma^n = \Sigma^n(x^n_{b}) \).

Let us return to the problem (11) in Section 2.6, and the operator \( L_x^0 \) contains fractional-order operators. Then \( k^{n,T}_{n,p}(x, x') \) can be obtained by performing the inverse Fourier transform on \( k^{n,T}_{n,p}(\omega, \omega') \), where

\[
k^{n,T}_{n,p}(\omega, \omega') = (-i \omega) \left[ 1 - \Delta t \cdot g' \cdot (-i \omega')^p \right] k^{n,T}_{n,p}(\omega, \omega').
\]

### 4.2. Mixed Boundary Conditions

For the mixed boundary conditions (38), the processing of boundary information is more complicated. Introduce \( w_1^n = \lambda_{11}^n \frac{\partial u^n}{\partial x} + \lambda_{12}^n u^n \), \( w_2^n = \lambda_{21}^n \frac{\partial u^n}{\partial x} + \lambda_{22}^n u^n \),
Fractal Fract. 2022, 6, 606

Return to the problem (1), then we can obtain $y \sim GP(0, K)$, where

$$
\begin{align*}
K & = \begin{bmatrix}
    k_{n,1,n}^n & k_{n,2,n}^n & \cdots & k_{n,m,n}^n \\
    k_{n,2,1}^n & k_{n,2,2}^n & \cdots & k_{n,m,2}^n \\
    \vdots & \vdots & \ddots & \vdots \\
    k_{n,m,1}^n & k_{n,m,2}^n & \cdots & k_{n,m,m}^n
\end{bmatrix} \\
\end{align*}
$$

where $\lambda_{11}^n = \lambda_{11}(n^2)$, $\lambda_{12}^n = \lambda_{12}(n^2)$, $\lambda_{21}^n = \lambda_{21}(n^2)$, $\lambda_{22}^n = \lambda_{22}(n^2)$. Note $L_{x}^n u^n = L_{x}^n \frac{\partial u^n}{\partial x} + \lambda_{12}^n u^n$, $L_{x}^n u^n = L_{x}^n \frac{\partial u^n}{\partial x} + \lambda_{22}^n u^n$, where $L_{x}^n$ and $L_{x}^n$ are linear operators.

By (5), train the hyper-parameters $\{\theta_j\}$ by minimizing NLML (46),

$$
- \log p(y|x^n, x_{n-1}^n, \cdots, x_{1}^n, x_{0}^n, \theta_0, \cdots, \theta_{m+1}) = \frac{1}{2} \log |K| + \frac{1}{2} y^T K^{-1} y + \frac{N}{2} \log 2 \pi,
$$

After the training part, the posterior distribution of $u^n(x^n)$ can be directly written as

$$
\begin{align*}
u^n(x^n) & \sim N \left( \begin{bmatrix}
    u_{1,1}^n \\
u_{2,1}^n \\
    \vdots \\
    u_{m,1}^n \\
\end{bmatrix}, \begin{bmatrix}
    q^T K^{-1} q \\
    \vdots \\
    \vdots \\
    q^T K^{-1} q
\end{bmatrix} \right)
\end{align*}
$$

where

$$
\begin{align*}
u^n(x^n) & \sim N \left( \begin{bmatrix}
    u_{1,1}^n \\
u_{2,1}^n \\
    \vdots \\
    u_{m,1}^n \\
\end{bmatrix}, \begin{bmatrix}
    q^T K^{-1} q \\
    \vdots \\
    \vdots \\
    \vdots \\
\end{bmatrix} \right)
\end{align*}
$$

where $q^T = [k_{n,1,1}^n(x^n, x^n), k_{n,2,1}^n(x^n, x^n), \cdots, k_{n,m,1}^n(x^n, x^n)]$, $k_{n,1,2}^n(x^n, x^n), k_{n,2,2}^n(x^n, x^n), \cdots, k_{n,m,2}^n(x^n, x^n)$, $k_{n,1,m}^n(x^n, x^n), k_{n,2,m}^n(x^n, x^n), \cdots, k_{n,m,m}^n(x^n, x^n)$.

By Section 2.4, in order to accurately propagate the uncertainty, marginalize $u^n$ (and perhaps $u^{n-1}$, when $n > \tau_j$) by employing (assume $\tau_j < n - \eta \leq \tau_{j+1}$).
In this subsection, we will solve a nonlinear PDE with two delays, and Dirichlet, Neumann, and mixed boundary conditions will be considered in this PDE, in turn.

\[ u^{n-\eta} \approx N(\mu^{n-\eta}, \Sigma^{n-\eta, n-\eta}), \quad (48) \]

\[ u^n(x_n^m) = q^TK^{-1}u^{n-\tau_p}, u^n_{n+1}, \ldots, u^n_{n+\tau_p+1} \sim N(\mu^n(x_n^m), \Sigma^{n,n}(x_n^m, x_n^m)), \quad (49) \]

where \( \mu^n(x_n^m) = q^TK^{-1}u^{n-\tau_p}, u^n_{n+1}, \ldots, u^n_{n+\tau_p+1} \), and \( \Sigma^{n,n}(x_n^m, x_n^m) = k^{n,n}(x_n^m, x_n^m) - q^TK^{-1}q + q^TK^{-1}SK^{-1}q, \)

and \( S = \text{diag}(0, \Sigma^{n-\tau_1,n-\tau_1}, \ldots, \Sigma^{n-\tau_p,n-\tau_p}, 0, \ldots, \Sigma^{n-1,n-1}). \)

On the basis of the posterior distribution (49), obtain the simplified form \( u^n \sim N(\mu^n, \Sigma^{n,n}), \) and then obtain the artificial data \( \{x_n^m, u^n\} \) of the next time step, where \( \mu^n = \mu^n(x_n^m), \) and \( \Sigma^{n,n} = \Sigma^{n,n}(x_n^m, x_n^m). \)

Return to problem (11) in Section 2.6, and the operator \( L^\gamma \) contains fractional-order operators. Then \( k^{n,n-1}_{n,1}(x, x') \) and \( k^{n,n-1}_{n,2}(x, x') \) can be obtained by performing the inverse Fourier transform on \( k^{n,n-1}_{n,1}(x, x') \) and \( k^{n,n-1}_{n,2}(x, x') \), respectively, where

\[
\begin{align*}
k^{n,n-1}_{n,1}(\omega, \omega') &= [\lambda^{n}_{11}(-i\omega) + \lambda^{n}_{12}][1 - \Delta t \cdot g'(\omega')/(-i\omega')^n - \Delta t \cdot g'(-i\omega')/(-i\omega')^n]k^{n,n}_{n,1}(\omega, \omega'), \\
k^{n,n-1}_{n,2}(\omega, \omega') &= [\lambda^{n}_{21}(-i\omega) + \lambda^{n}_{22}][1 - \Delta t \cdot g'(\omega')/(-i\omega')^n - \Delta t \cdot g'(-i\omega')/(-i\omega')^n]k^{n,n}_{n,2}(\omega, \omega').
\end{align*}
\]

4.3. Numerical Tests for a Problem with Two Delays and Different Types of Boundary Conditions

In this subsection, we will solve a nonlinear PDE with two delays, and Dirichlet, Neumann, and mixed boundary conditions will be considered in this PDE, in turn.
Example 5.

\[
\begin{aligned}
\begin{cases}
\frac{\partial u}{\partial t} &= (u + 1)\frac{\partial^2 u}{\partial x^2} + \sin \frac{\partial u}{\partial x} + u \frac{\partial^2 u}{\partial x \partial t} u(x, t - 0.1) + \cos u \frac{\partial^2 u}{\partial x \partial t} u(x, t - 0.2) + f(x, t), \quad (x, t) \in (0, 2\pi) \times (0, 1), \\
u(x, t) &= (1 + t^2)(\frac{\partial u}{\partial x} + \sin x), \quad -0.2 \leq t \leq 0,
\end{cases}
\end{aligned}
\tag{51}
\]

Example 5 is a nonlinear PDE with two delays. The exact solution of (51) is \(u(x, t) = (1 + t^2)(\frac{x}{\pi} + \sin x)\), and the inhomogeneous term is \(f(x, t) = \frac{1}{2\pi}(-((1 + (-0.2 + t)^2)(1 + 2\pi \cos x) \cos((1 + t^2)(\frac{x}{\pi} + \sin x))) + 2t(x + 2\pi \sin x) \sin((1 + t^2)(\frac{x}{\pi} + \sin x)) - (1 + (-0.1 + t)^2)(1 + 2\pi \cos x) \sin((1 + t^2)(\frac{x}{\pi} + \sin x)))\). The Dirichlet, Neumann and mixed boundary conditions of Example 5 are as follows:

\[
u(0, t) = 0, \quad u(2\pi, t) = 1 + t^2, \quad 0 < t \leq 1,
\tag{52}
\]

\[
\left. \left[ \frac{\partial u}{\partial x} \right] \right|_{x=0} = (\frac{1}{2\pi} + 1)(1 + t^2), \quad \left[ \frac{\partial u}{\partial x} \right]_{x=2\pi} = (\frac{1}{2\pi} + 1)(1 + t^2), \quad 0 < t \leq 1.
\tag{53}
\]

\[
\left. (4 + t)\frac{\partial u}{\partial x} + (1 + t^2)u \right|_{x=0} = (\frac{1}{2\pi} + 1)(4 + t)(1 + t^2), \quad 0 < t \leq 1,
\tag{54}
\]

\[
\left. (1 + t)\frac{\partial^2 u}{\partial x^2} + (3 + 2\sin t)u \right|_{x=2\pi} = [(\frac{1}{\pi} + 1)(1 + t^2) + (3 + 2\sin t)](1 + t^2), \quad 0 < t \leq 1.
\]

Numerical tests of solving (51) will be performed with noiseless data. Figure 9 shows the posterior distribution of the predicted solution at different time nodes, where we fix the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) to be 10 and fix \(\Delta t\) to be 0.01. Figure 9a–c presents the results of the experiments with Dirichlet, Neumann and mixed boundary conditions, respectively, and Figure 9d shows the maximum absolute error \(E_\infty\) with the increase in time. Moreover, Table 7 shows us the maximum absolute error \(E_\infty\) at some time nodes in detail, where Dirichlet, Neumann and mixed boundary conditions are all considered. It is found that the method based on MP-NGPs works well when solving nonlinear delay PDEs with Dirichlet, Neumann or mixed boundary conditions. The prediction accuracy under different types of boundary conditions is close to each other.
Figure 9. (a) The posterior distribution of the predicted solution at different time nodes, where Dirichlet boundary conditions are employed. (b) The posterior distribution of the predicted solution at different time nodes, where Neumann boundary conditions are employed. (c) The posterior distribution of the predicted solution at different time nodes, where mixed boundary conditions are employed. (d) Time versus the maximum absolute error \( E_\infty \) until the final time step is reached.

Table 7. Example 5: Maximum absolute error \( E_\infty \) at some time nodes, where \( \Delta t = 0.01 \), where Dirichlet, Neumann and mixed boundary conditions are used, respectively.

<table>
<thead>
<tr>
<th>( t )</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirichlet</td>
<td>( 1.933660 \times 10^{-2} )</td>
<td>( 1.133342 \times 10^{-2} )</td>
<td>( 9.185798 \times 10^{-3} )</td>
<td>( 1.085891 \times 10^{-2} )</td>
<td>( 1.424333 \times 10^{-2} )</td>
</tr>
<tr>
<td>Neumann</td>
<td>( 2.687072 \times 10^{-3} )</td>
<td>( 2.880986 \times 10^{-3} )</td>
<td>( 3.562039 \times 10^{-3} )</td>
<td>( 6.123954 \times 10^{-3} )</td>
<td>( 1.009985 \times 10^{-2} )</td>
</tr>
<tr>
<td>Mixed</td>
<td>( 6.755766 \times 10^{-3} )</td>
<td>( 6.754411 \times 10^{-3} )</td>
<td>( 3.984626 \times 10^{-3} )</td>
<td>( 6.110999 \times 10^{-3} )</td>
<td>( 9.978279 \times 10^{-3} )</td>
</tr>
</tbody>
</table>

\( t \) | \( 0.6 \) | \( 0.7 \) | \( 0.8 \) | \( 0.9 \) | \( 1.0 \) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirichlet</td>
<td>( 1.890978 \times 10^{-2} )</td>
<td>( 2.444410 \times 10^{-2} )</td>
<td>( 3.139654 \times 10^{-2} )</td>
<td>( 4.025414 \times 10^{-2} )</td>
<td>( 5.092472 \times 10^{-2} )</td>
</tr>
<tr>
<td>Neumann</td>
<td>( 1.704219 \times 10^{-2} )</td>
<td>( 2.634275 \times 10^{-2} )</td>
<td>( 3.677554 \times 10^{-2} )</td>
<td>( 4.920981 \times 10^{-2} )</td>
<td>( 6.465664 \times 10^{-2} )</td>
</tr>
<tr>
<td>Mixed</td>
<td>( 1.571632 \times 10^{-2} )</td>
<td>( 2.541023 \times 10^{-2} )</td>
<td>( 3.595499 \times 10^{-2} )</td>
<td>( 4.830915 \times 10^{-2} )</td>
<td>( 6.284298 \times 10^{-2} )</td>
</tr>
</tbody>
</table>

5. Processing of Wave Equations

So as to prove the efficiency of MP-NGPs, we give the processing of another type of time-dependent PDEs, wave equations,

\[
\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{\partial^2 u(x,t)}{\partial x^2} + \frac{\partial u(x,t-s)}{\partial x} + f(x,t), \ lb < x < ub, 0 < t \leq T,
\]
\[
u(lb,t) = \phi_1(t), u(ub,t) = \phi_2(t), 0 < t \leq T,
\]
\[
u(x,t) = \psi(x,t), lb < x \leq ub, -s \leq t \leq 0,
\]
\[
\frac{\partial u}{\partial x}(x,0) = \varphi(x), lb \leq x \leq ub,
\]

where \( \phi_1(t), \phi_2(t), \psi(x,t), \) and \( \varphi(x) \) are given functions. Apparently, delay terms and inhomogeneous terms are also considered in the wave equations.

We will give two methods of how to solve wave equations by the algorithm of MP-NGPs.
5.1. Second Order Central Difference Scheme

The first method is to apply a second-order central difference scheme of second-order derivatives on the first Equation of (55) to discretize the temporal variable:

\[ \frac{u^{n-1} + u^{n+1} - 2u^n}{\Delta t^2} = \frac{\partial^2}{\partial x^2} u^n + \frac{\partial}{\partial x} u^{n-\tau} + f^n, \]  

(56)

where \( n = 0, 1, \cdots, N - 1, \) \( N = T/\Delta t, \) \( u^n = u(x, t^n), \) \( u^{n-\tau} = u(x, t^{n-s}), \) \( \tau = s/\Delta t, \) \( f^n = f(x, t^n). \)

Rearrange (56) to obtain

\[ u^{n+1} = 2u^n + \Delta t^2 \frac{\partial^2}{\partial x^2} u^n - u^{n-1} + \Delta t^2 \frac{\partial}{\partial x} u^{n-\tau} + \Delta t^2 f^n. \]  

(57)

Obviously, the scheme (56) is a multi-steps method, while the backward Euler scheme is a one-step method. Then introduce notations that \( \mathcal{L}_{0}^2 u^n = 2u^n + \Delta t^2 \frac{\partial^2}{\partial x^2} u^n, \) \( \mathcal{L}_{1}^2 u^{n-1} = -u^{n-1}, \) \( \mathcal{L}_{2}^2 u^{n-\tau} = \Delta t^2 \frac{\partial}{\partial x} u^{n-\tau} \) and \( \mathcal{L}_{3}^2 f^n = \Delta t^2 f^n. \) So, we can obtain the simplified equation

\[ u^{n+1} = \mathcal{L}_{0}^2 u^n + \mathcal{L}_{1}^2 u^{n-1} + \mathcal{L}_{2}^2 u^{n-\tau} + \mathcal{L}_{3}^2 f^n. \]  

(58)

Assume the Gaussian process prior as follows:

\[ u^n(x) \sim \mathcal{GP}(0, k_{u,u}^n(x, x'; \theta_0)), \]
\[ u^{n-1}(x) \sim \mathcal{GP}(0, k_{u,u}^{n-1}(x, x'; \theta_1)), \]
\[ u^{n-\tau}(x) \sim \mathcal{GP}(0, k_{u,u}^{n-\tau}(x, x'; \theta_2)), \]
\[ f^n(x) \sim \mathcal{GP}(0, k_{f,f}^n(x, x'; \theta_3)). \]  

(59)

According to the properties of Gaussian processes, then we can obtain \( y \sim \mathcal{GP}(0, K), \)

where

\[ y = \begin{bmatrix} u^n \\ u^{n-1} \\ u^{n-\tau} \\ u^{n+1} \\ f^n \\ \vdots \end{bmatrix}, \quad K = \begin{bmatrix} k_{u,u}^{n,n} & k_{u,u}^{n,n-1} & k_{u,u}^{n,n-\tau} & k_{u,u}^{n,n+1} & k_{u,u}^{n+1,n} \\ k_{u,u}^{n-1,n} & k_{u,u}^{n-1,n-1} & k_{u,u}^{n-1,n-\tau} & k_{u,u}^{n-1,n+1} & k_{u,u}^{n-1,n+1} \\ k_{u,u}^{n-\tau,n} & k_{u,u}^{n-\tau,n-1} & k_{u,u}^{n-\tau,n-\tau} & k_{u,u}^{n-\tau,n+1} & k_{u,u}^{n-\tau,n+1} \\ k_{u,u}^{n+1,n} & k_{u,u}^{n+1,n-1} & k_{u,u}^{n+1,n-\tau} & k_{u,u}^{n+1,n+1} & k_{u,u}^{n+1,n+1} \\ k_{f,f}^{n,n} & k_{f,f}^{n,n-1} & k_{f,f}^{n,n-\tau} & k_{f,f}^{n,n+1} & k_{f,f}^{n+1,n} \\ k_{f,f}^{n-1,n} & k_{f,f}^{n-1,n-1} & k_{f,f}^{n-1,n-\tau} & k_{f,f}^{n-1,n+1} & k_{f,f}^{n-1,n+1} \\ k_{f,f}^{n-\tau,n} & k_{f,f}^{n-\tau,n-1} & k_{f,f}^{n-\tau,n-\tau} & k_{f,f}^{n-\tau,n+1} & k_{f,f}^{n-\tau,n+1} \\ k_{f,f}^{n+1,n} & k_{f,f}^{n+1,n-1} & k_{f,f}^{n+1,n-\tau} & k_{f,f}^{n+1,n+1} & k_{f,f}^{n+1,n+1} \end{bmatrix}. \]

Any kernel not mentioned above is equal to \( 0. \)

By (5), train the hyper-parameters \( \{\theta_j|j = 0, 1, 2, 3\} \) by minimizing NLML (60):

\[ -\log p(y|x^{n+1}_b, x^{n-\tau}_u, x^n, x^{n-1}_u, x^n, \theta_0, \theta_1, \theta_2, \theta_3) = \frac{1}{2} \log |K| + \frac{1}{2} y^T K^{-1} y + \frac{N}{2} \log 2\pi, \]  

(60)

After the training part, the posterior distribution of \( u^{n+1}(x^n_b) \) can be directly written as
\( u^{n+1}(x_s^{n+1}) \sim N \left( \begin{bmatrix} u^n \\ u^{n-1} \\ u^{n-\tau} \\ f^n \\ u_b^n \end{bmatrix}, \begin{bmatrix} q^T K^{-1} & k_n^{n+1,n+1} & k_n^{n+1,n+1} & k_n^{n+1,n+1} & 0 \\ k_n^{n+1,n+1} & q^T K^{-1} & k_n^{n+1,n+1} & k_n^{n+1,n+1} & 0 \\ k_n^{n+1,n+1} & k_n^{n+1,n+1} & q^T K^{-1} & k_n^{n+1,n+1} & 0 \\ k_n^{n+1,n+1} & k_n^{n+1,n+1} & k_n^{n+1,n+1} & q^T K^{-1} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right) \), \quad (61)

where \( q^T = [k_n^{n+1,n+1}(x_s^{n+1}, x_b^{n+1}), k_n^{n+1,n+1}(x_s^{n+1}, x_b^{n+1}), k_n^{n+1,n+1}(x_s^{n+1}, x_b^{n+1}), k_n^{n+1,n+1}(x_s^{n+1}, x_b^{n+1}), k_n^{n+1,n+1}(x_s^{n+1}, x_b^{n+1})] \).

On the basis of Section 2.4, in order to accurately propagate the uncertainty, marginalize \( u^n, u^{n-1} \) (and \( u^{n-\tau} \), when \( n > \tau \)) by employing

\[
\begin{align*}
\begin{bmatrix}
    u^{n+1} \\
    u^{n+1} \\
    u^{n+1}
\end{bmatrix}
    &=
    \begin{bmatrix}
    u^n \\
    u^n \\
    u^n
\end{bmatrix}
    +
    \begin{bmatrix}
    u^{n-1} \\
    u^{n-1} \\
    u^{n-1}
\end{bmatrix}
    +
    \begin{bmatrix}
    f^n \\
    f^n \\
    f^n
\end{bmatrix}
    \sim N \left( \begin{bmatrix} \mu^{n+1} \\ \mu^{n+1} \end{bmatrix}, \begin{bmatrix} \Sigma^{n+1,n+1} \\ \Sigma^{n+1,n+1} \end{bmatrix} \right),
\end{align*}
\]

(62)

\[
\begin{align*}
\begin{bmatrix}
    u^{n+1} \\
    u^{n+1} \\
    u^{n+1}
\end{bmatrix}
    &=
    \begin{bmatrix}
    u^n \\
    u^n \\
    u^n
\end{bmatrix}
    +
    \begin{bmatrix}
    u^{n-1} \\
    u^{n-1} \\
    u^{n-1}
\end{bmatrix}
    +
    \begin{bmatrix}
    f^n \\
    f^n \\
    f^n
\end{bmatrix}
    \sim N \left( \begin{bmatrix} \mu^{n+1} \\ \mu^{n+1} \end{bmatrix}, \begin{bmatrix} \Sigma^{n+1,n+1} \\ \Sigma^{n+1,n+1} \end{bmatrix} \right),
\end{align*}
\]

(63)

where \( \mu^{n+1}(x_s^{n+1}) = q^T K^{-1} \begin{bmatrix} \mu^n \\ \mu^{n-1} \\ \mu^{n-\tau} \\ f^n \\ u_b^n \end{bmatrix} \) (if \( n = 0 \), replace \( \mu^n \) by \( u^n \); if \( n < 2 \), replace \( \mu^{n-1} \) by \( u^{n-1} \)).

By \( u^{n+1} \), if \( n \leq \tau \), replace \( \mu^{n-\tau} \) by \( u^{n-\tau} \), \( \Sigma^{n+1,n+1}(x_s^{n+1}, x_s^{n+1}) = k_{n,dt}^{n+1,n+1}(x_s^{n+1}, x_s^{n+1}) - q^T K^{-1} q + q^T K^{-1} S K^{-1} q \), and \( S = \text{diag}(\Sigma^{n,n}, \Sigma^{n-1,n-1}, \Sigma^{n-1,n-1}, 0, 0) \) (if \( n = 0 \), replace \( \Sigma^{n,n} \) by \( 0 \); if \( n < 2 \), replace \( \Sigma^{n-1,n-1} \) by \( 0 \); if \( n \leq \tau \), replace \( \Sigma^{n-\tau,n-\tau} \) by \( 0 \)).

By the posterior distribution (63), obtain the simplified form \( u^{n+1} \sim N \left( \begin{bmatrix} \mu^{n+1} \\ \mu^{n+1} \end{bmatrix}, \begin{bmatrix} \Sigma^{n+1,n+1} \\ \Sigma^{n+1,n+1} \end{bmatrix} \right) \), and then obtain the artificial data \( \{x_s^{n+1}, u^{n+1}\} \) of the next time step, where \( \mu^{n+1} = \mu^{n+1}(x_s^{n+1}) \), and \( \Sigma^{n+1,n+1} = \Sigma^{n+1,n+1}(x_s^{n+1}, x_s^{n+1}) \).

5.2. Backward Euler Scheme

Firstly, note \( \frac{\partial}{\partial t} u(x, t) = v(x, t) \), then obtain

\[
\begin{align*}
\frac{\partial}{\partial t} u^n &= v^n, \\
\frac{\partial}{\partial x} v^n &= \frac{\partial^2}{\partial x^2} u^n + \frac{\partial}{\partial x} u^{n-\tau} + f^n, 
\end{align*}
\]

(64)
where \( v^n = u(x, t^n) \). Apply the backward Euler scheme on the first equation of (55) to discretize time,

\[
\begin{aligned}
\frac{u^n - u^{n-1}}{\Delta t} &= v^n, \\
\frac{v^n - v^{n-1}}{\Delta t} &= \frac{\partial^2}{\partial x^2} u^n + \frac{\partial}{\partial x} u^{n-\tau} + f^n.
\end{aligned}
\]  

(65)

Rearrange (65) to obtain

\[
\begin{aligned}
u^{n+1} &= u^n - \Delta t v^n, \\
v^{n} &= \frac{v^n - v^{n-1}}{\Delta t} + \Delta t \frac{\partial^2}{\partial x^2} u^n - \Delta t \frac{\partial}{\partial x} u^{n-\tau} - \Delta t f^n.
\end{aligned}
\]  

(66)

Then introduce notations that \( L^0_x u^n = u^n, L^1_x v^n = -\Delta t v^n, L^2_x v^n = v^n, L^3_x u^n = -\Delta t \frac{\partial^2}{\partial x^2} u^n, \) \( L^4_x u^{n-\tau} = -\Delta t \frac{\partial}{\partial x} u^{n-\tau} \), and \( L^5_x f^n = \Delta t f^n \). Then obtain the simplified form

\[
\begin{aligned}
u^{n+1} &= L^0_x u^n + L^1_x v^n, \\
v^{n} &= L^2_x v^n + L^3_x u^n + L^4_x u^{n-\tau} + L^5_x f^n.
\end{aligned}
\]  

(67)

Assume the Gaussian process prior as follows:

\[
\begin{aligned}
u^n(x) &\sim \mathcal{GP}(0, k_{u,u}(x, x'; \theta_0)), \\
v^n(x) &\sim \mathcal{GP}(0, k_{v,v}(x, x'; \theta_1)), \\
u^{n-\tau}(x) &\sim \mathcal{GP}(0, k_{u,v}(x, x'; \theta_2)), \\
f^n(x) &\sim \mathcal{GP}(0, k_{f,f}(x, x'; \theta_3)).
\end{aligned}
\]  

(68)

According to the properties of Gaussian processes, we can obtain \( y \sim \mathcal{GP}(0, K) \), where

\[
y = \begin{bmatrix} u^n \\ u^{n-\tau} \\ v^n \\ v^{n-1} \end{bmatrix}, \quad K = \begin{bmatrix} k_{u,u} & k_{u,v} & k_{u,f} & k_{u,v} \\ k_{u,v} & k_{v,v} & k_{v,f} & k_{v,v} \\ k_{u,f} & k_{v,f} & k_{f,f} & k_{f,f} \\ k_{u,v} & k_{v,v} & k_{v,f} & k_{v,v} \end{bmatrix},
\]

\[
k_{u,u} = k_{u,u}(x^n, x^n) + \sigma_u^2 I, k_{u,v} = k_{u,v}(x^n, x^{n-1}), k_{u,f} = k_{u,f}(x^n, f^n), \\
k_{v,v} = k_{v,v}(x^{n-1}, x^{n-1}) + \sigma_v^2 I, k_{v,f} = k_{v,f}(x^{n-1}, f^n), \\
k_{f,f} = k_{f,f}(f^n, f^n), k_{v,v} = k_{v,v}(x^n, x^n). 
\]

Any kernel not mentioned above is equal to 0.

By (5), train the hyper-parameters \( \{\theta_j\}_{j=0,1,2,3} \) by minimizing NLML (69):

\[
- \log p(y|x^n_u, x^{n-\tau}_u, x^f, x^{n-1}_u, x^{n-1}_v, x^{n-1}_f, \theta_0, \theta_1, \theta_2, \theta_3) = \frac{1}{2} \log |K| + \frac{1}{2} y^T K^{-1} y + \frac{N}{2} \log 2\pi.
\]  

(69)

After the training part, the joint posterior distribution of \( u^{n+1}(x^n_n) \) can be directly written as

\[
\begin{bmatrix}
U^n(x_u) \\
Z^n(x_u) \\
F^n \\
U^{n-1} \\
Z^{n-1}
\end{bmatrix} \sim N \begin{bmatrix}
U^n(x_u) \\
Z^n(x_u) \\
F^n \\
U^{n-1} \\
Z^{n-1}
\end{bmatrix} K^{-1} \begin{bmatrix}
U^n(x_u) \\
Z^n(x_u) \\
F^n \\
U^{n-1} \\
Z^{n-1}
\end{bmatrix},
\]

(70)
where
\[
q^T = \begin{bmatrix}
  k_{v_u}^{n+1}(x_{i+1}, x_u^0) & k_{v_u}^{n+1}(x_{i+1}, x_u^0) & \cdots & k_{v_u}^{n+1}(x_{i+1}, x_u^0) \\
  k_{v_u}^{n+1}(x_{i+1}, x_u^0) & k_{v_u}^{n+1}(x_{i+1}, x_u^0) & \cdots & k_{v_u}^{n+1}(x_{i+1}, x_u^0) \\
  \vdots & \vdots & \ddots & \vdots \\
  k_{v_u}^{n+1}(x_{i+1}, x_u^0) & k_{v_u}^{n+1}(x_{i+1}, x_u^0) & \cdots & k_{v_u}^{n+1}(x_{i+1}, x_u^0)
\end{bmatrix}.
\] (71)

On the basis of Section 2.4, in order to accurately propagate the uncertainty, marginalize \( u^{n-1}, v^{n-1} \) (and \( u^{n-1}, v^{n-1} \), when \( n > \tau \)) by employing
\[
\begin{align*}
\begin{bmatrix}
  u^{n-1} & u^{n-1} & \cdots & u^{n-1}
\end{bmatrix} & \sim N\left( \begin{bmatrix} \mu_v^{n-1} & \mu_u^{n-1} \end{bmatrix}, \begin{bmatrix} \Sigma_v^{n-1} & \Sigma_{v,u}^{n-1} \\ \Sigma_{v,u}^{n-1} & \Sigma_{u,u}^{n-1} \end{bmatrix} \right) \quad n \leq \tau, \\
\begin{bmatrix}
  u^{n} & u^{n} & \cdots & u^{n}
\end{bmatrix} & \sim N\left( \begin{bmatrix} \mu_v^{n} & \mu_u^{n} \end{bmatrix}, \begin{bmatrix} \Sigma_v^{n} & \Sigma_{v,u}^{n} \\ \Sigma_{v,u}^{n} & \Sigma_{u,u}^{n} \end{bmatrix} \right) \quad n > \tau,
\end{align*}
\] (72)

to obtain
\[
\begin{align*}
\begin{bmatrix}
  u^n(x_{i+1}) \\
v^n(x_{i+1})
\end{bmatrix} & \sim N\left( \begin{bmatrix} \mu_v^n & \mu_u^n \end{bmatrix}, \begin{bmatrix} \Sigma_v^n & \Sigma_{v,u}^n \\ \Sigma_{v,u}^n & \Sigma_{u,u}^n \end{bmatrix} \right), \\
\begin{bmatrix}
  u^n(x_{i+1}) \\
v^n(x_{i+1})
\end{bmatrix} & \sim N\left( \begin{bmatrix} \mu_v^n & \mu_u^n \end{bmatrix}, \begin{bmatrix} \Sigma_v^n & \Sigma_{v,u}^n \\ \Sigma_{v,u}^n & \Sigma_{u,u}^n \end{bmatrix} \right),
\end{align*}
\] (73)

where
\[
\begin{bmatrix}
  \mu_v^n(x_{i+1}) \\
\mu_u^n(x_{i+1})
\end{bmatrix} = q^T K^{-1} \begin{bmatrix} \mu_v^{n-1} \\
\mu_u^{n-1} \end{bmatrix} \quad (\text{if } n = 0, \text{ replace } \mu_v^{n-1} \text{ and } \mu_u^{n-1} \text{ by } u^{n-1} \text{ and } \mu_v^{n-1}, \mu_u^{n-1}, \text{ respectively; if } n \leq \tau, \text{ replace } \mu_v^{n-1} \text{ by } u^{n-1}),
\]
\[
\begin{align*}
\Sigma_v^n & = \begin{bmatrix} \Sigma_v^{n-1} & \Sigma_{v,u}^{n-1} \\ \Sigma_{v,u}^{n-1} & \Sigma_{u,u}^{n-1} \end{bmatrix} - q^T K^{-1} q + q^T K^{-1} S K^{-1} q, \\
S & = \text{diag}(0, \Sigma_{u,u}^{n-1}), \quad q = \text{diag}(0, \Sigma_{u,u}^{n-1}), \quad \Sigma_{u,u}^{n-1} = \Sigma_{u,u}^{n-1} - \Sigma_{u,u}^{n-1},
\end{align*}
\]

By the posterior distribution (73), obtain the simplified form \( u^n \sim N(\mu_v^n, \Sigma_{u,u}^{n}) \), and then obtain the artificial data \( \{x^n, u^n\} \) of the next time step, where \( \mu_v^n = \mu_v^n(x_{i+1}) \), and \( \Sigma_{u,u}^{n} = \Sigma_{u,u}^{n}(x_{i+1}, x_{i+1}) \).

This two methods reflect the flexibility of NGPs and MP-NGPs. Due to the limited space of this paper, we only introduced the processing of wave equations. The other types of time-dependent PDEs can also be solved by the idea of this two methods.

5.3. Numerical Tests

Example 6.

\[
\begin{align*}
\frac{\partial^2 u(x,t)}{\partial t^2} &= \frac{\partial^2 u(x,t)}{\partial x^2} + \frac{\partial u(x,t)}{\partial x} + f(x,t), 0 < x < 2\pi, 0 < t \leq 1, \\
u(0,t) &= e^{-t}, u(2\pi, t) = e^{-t}, 0 < t \leq 1, \\
u(x,t) &= e^{-t}(\cos 2x + \sin x), 0 \leq x \leq 2\pi, -0.2 \leq t \leq 0, \\
\frac{\partial u}{\partial t}(x, 0) &= -\cos 2x - \sin x, 0 \leq x \leq 2\pi,
\end{align*}
\] (74)
Example 6 is the wave equation with one delay. The exact solution of (74) is \( u(x, t) = e^{-t}(\cos 2x + \sin x) \), and the inhomogeneous term is \( f(x, t) = e^{-t}(5\cos 2x + 2\sin x) - e^{0.2-t}(\cos x - 2\sin 2x) \).

Numerical tests of solving (74) are performed with noiseless data. Figure 10 shows the posterior distribution of the predicted solution at different time nodes, where we fix the number of initial data points (inhomogeneous term data points, artificially generated data points and delay term data points) to be 15 and fix \( \Delta t \) to be 0.01. Figure 9a,b, presents the results of the experiments employing the second-order central difference scheme and the backward Euler scheme, respectively, and Figure 9c shows the maximum absolute error \( E_\infty \) with the increase in time. Moreover, Table 8 shows us the maximum absolute error \( E_\infty \) at some time nodes in detail. According to the results of numerical tests, we conclude that the two methods based on MP-NGPs effectively solve the wave equation with one delay. The second method applying the backward Euler scheme behaves better than the first method, applying the second-order central difference scheme on prediction accuracy.

**Figure 10.** Wave equation of Example 6: (a) The posterior distribution of the predicted solution at different time nodes, where the second order central difference scheme is used. (b) The posterior distribution of the predicted solution at different time nodes, where the backward Euler scheme is used. (c) Time versus maximum absolute error \( E_\infty \) until the final time step is reached, where the second order central difference scheme (SCD) and the backward Euler scheme (BE) are both considered.
Table 8. Wave equation of Example 6: Maximum absolute error $E_\infty$ at some time nodes, where $\Delta t = 0.01$, where the second order central difference scheme (SCD) and the backward Euler scheme (BE) are applied, respectively.

<table>
<thead>
<tr>
<th>$t$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCD</td>
<td>$3.521926 \times 10^{-2}$</td>
<td>$1.731871 \times 10^{-2}$</td>
<td>$2.562235 \times 10^{-2}$</td>
<td>$3.339859 \times 10^{-2}$</td>
<td>$3.854518 \times 10^{-2}$</td>
</tr>
<tr>
<td>BE</td>
<td>$9.577671 \times 10^{-3}$</td>
<td>$1.647835 \times 10^{-3}$</td>
<td>$2.197030 \times 10^{-3}$</td>
<td>$2.490295 \times 10^{-3}$</td>
<td>$2.566583 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t$</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCD</td>
<td>$4.179861 \times 10^{-3}$</td>
<td>$4.218729 \times 10^{-3}$</td>
<td>$3.356628 \times 10^{-2}$</td>
<td>$2.681414 \times 10^{-2}$</td>
<td>$2.301241 \times 10^{-2}$</td>
</tr>
<tr>
<td>BE</td>
<td>$2.471149 \times 10^{-3}$</td>
<td>$2.195480 \times 10^{-3}$</td>
<td>$1.811016 \times 10^{-3}$</td>
<td>$1.411214 \times 10^{-3}$</td>
<td>$1.308891 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

6. Discussions

In this paper, a new method, called multi-priors numerical Gaussian processes (MP-NGPs), is designed and proposed on the basis of numerical Gaussian processes (NGPs), to solve complex time-dependent PDEs. Compared with NGPs, the MP-NGPs proposed possess a structure with a much stricter mathematical deduction, and the application of MP-NGPs is more flexible. Delay terms and inhomogeneous terms can be introduced on MP-NGPs. The main idea of MP-NGPs is to realize the discretization of time, apply Gaussian process regression to transform the potential information of PDEs into a scheme controlled by several operators, and predict the solution from the statistical perspective. MP-NGPs build a Gaussian process regression at each time step to predict the solution at the next time step. The basic idea of MP-NGPs is very similar to that of NGPs, but the applications of MP-NGPs are explored more deeply than NGPs. Moreover, this methodology of MP-NGPs has natural advantages in handling noisy data and quantifying the uncertainty of the predicted solution. It should be mentioned that MP-NGPs can only solve a class of fractional-order PDEs composed of special fractional-order derivatives (2). Other kinds of fractional-order derivatives still deserve further research in MP-NGPs. Moreover, multi-spatial-dimension irregular domain problems, the inverse problems of PDEs, such as estimating constant or variable coefficients based on Gaussian processes and some specific application problems of MP-NGPs, may be the direction of our future investigation.

7. Conclusions

In this manuscript, the possibility of MP-NGPs in solving nonlinear delay partial differential equations with multi-delays is explored. These equations investigated include partial differential equations, partial integro-differential equations, fractional partial differential equations and wave equations. We also consider Neumann and mixed boundary conditions of PDEs on MP-NGPs. Numerical experiments prove that the method proposed can solve nonlinear delay partial differential equations with satisfactory precision, and the algorithm applying the backward Euler scheme is stably first-order accurate in time. So as to improve the prediction accuracy, Runge–Kutta methods under the frame of MP-NGPs are proposed in this manuscript. The algorithm applying the Runge–Kutta methods can effectively improve the prediction accuracy, although the temporal convergence rate of this algorithm is still of the first order according to the numerical experiments.


Funding: This work is supported by National Natural Science Foundation of P.R. China (No. 71974204).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.
Data Availability Statement: Not applicable.

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

References


