Advances in Applied Mathematics 应用数学进展, 2022, 11(3), 1222-1241
Published Online March 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.113133

Hans汉斯

深度学习与进化算法耦合下的最优多维随机控制问题

张智豪[1]，徐　勉[2*]

[1]上海理工大学，理学院，上海
[2]河南中烟工业有限责任公司洛阳卷烟厂，河南 洛阳

收稿日期：2022年2月21日；录用日期：2022年3月16日；发布日期：2022年3月23日

摘　要

受困于维数诅咒，能够求解高维偏微分方程(PDEs)的算法一直以来都极其有限。鄂维南和韩劼群在2017年提出的算法通过将未知解的梯度看作策略函数，利用深度学习可以较为有效的解决高维偏微分方程，但却无法解决带有真正策略函数的问题。本文提出了一种新算法，通过多层神经网络表示策略函数映射，将方程的解映射为适应度函数，把网络中的参数看作自变量，通过进化算法优化整个策略函数；同时配合鄂维南和韩劼群的算法求解问题。通过在Riccati方程和投资消费问题等的实际算例模拟下，表明了算法的准确性和实际意义。

关键词

偏微分方程，倒向随机微分方程，高维，深度学习，随机控制，进化算法

Solving Multi-Dimensional Optimal Stochastic Control Problems with Deep Learning and Evolution Algorithm

Zhihao Zhang[1], Mian Xu[2*]

[1]College of Science, University of Shanghai for Science and Technology, Shanghai
[2]Luoyang Cigarettes Factory of China Tobacco Henan Industrial Co., LTD., Luoyang Henan

Received: Feb. 21st, 2022; accepted: Mar. 16th, 2022; published: Mar. 23rd, 2022

Abstract

Because of the curse of dimensionality, developing efficient algorithms for solving high-dimensional

*通讯作者。

文章引用: 张智豪, 徐勉. 深度学习与进化算法耦合下的最优多维随机控制问题[J]. 应用数学进展, 2022, 11(3): 1222-1241. DOI: 10.12677/aam.2022.113133

partial differential equations (PDEs) has been an extremely difficult task. The algorithm which Weinan E and Jiequn Han proposed in 2017 views the gradient of the unknown solution as policy function, and through deep learning can effectively solve high-dimensional partial differential equations, but this method cannot deal with stochastic control problems with real policy function. We propose a new algorithm for solving this problem, which use multilayer neural network to represent the map of policy function and view the parameters in the neural network as independent variables. Then, we use the evolution algorithm to optimal the policy function. At the same time, we cooperate with Weinan E and Jiequn Han's algorithm to solve this problem. Numerical results on 5-dimensional Riccati equation and 12-dimensional Investment and Consumption Problem suggest the accuracy and practical significance of the algorithm.

Keywords

Partial Differential Equation, Backward Stochastic Differential Equation, High-Dimensional, Deep Learning, Stochastic Control, Evolution Algorithm

1. 引言

由于维数诅咒的存在[1]，建立行之有效的高维偏微分方程(Partial differential equation, PDE)的算法一直以来都是一个非常有挑战性的工作。比较传统的求解方法是使用近似动态规划[2]，但这种方法需要我们用一个近似函数替代掉真值函数，然后通过更新值函数的方式来求解出问题。在过去的二十多年中，由于倒向随机微分方程(Backward stochastic differential equation, BSDE)的出现及对其相关理论的充分研究，我们可以将随机偏微分方程(PDEs)的解通过 Feynman-Kac 公式(Feynman-Kacformula)表达为一个与其耦合的倒向随机微分方程的解[3] [4] [5]，在这样的理论基础之上，通过蒙特卡罗方法可以建立求解随机偏微分方程(PDEs)在任意时空间位置解的算法[6]。在信息时代的当下，鄂维南和韩劼群所开发的算法[7] [8] [9]，是将偏微分方程和倒向随机微分方程相结合，令所求方程解的梯度充当策略函数，然后把所给定的终端条件和倒向随机微分方程的解之间的误差视为损失函数，进而通过神经网络逼近策略函数来求解出问题，这种算法在 100 维的非线性偏微分方程中也表现出了良好的效果。在此之后，这种通过深度学习算法来求解高维随机偏微分方程的思路应用的越来越广泛[10] [11] [12] [13]，在误差减小、效率提升等方面的研究也获得了长足的进步[14] [15]。

但是，鄂维南和韩劼群的算法也有一些不完备的地方，在面对带有策略函数的随机控制问题[16]时，这种算法就会面临整个问题出有两种策略函数需要解决的局面。考虑到鄂维南和韩劼群的算法不需要设计近似函数和处理值函数，且能够同时处理线性非线性的高维偏微分方程问题，具有较强的普适性。因此，我们仍期望从他们的算法出发，开发出能够解决含有策略函数的随机控制问题算法。

近年来，深度学习作为机器学习领域中的新技术越来越为人所知，其在机器视觉、自然语言处理和自动驾驶中的广泛应用，以及处理高维复杂问题时的优异表现让人看到了它的巨大潜能。而深度学习所具有的万能近似定理[17]也表明，无论是什么样的函数，我们总能够通过一个足够庞大的前馈网络来将它表示出来。这个定理的存在启发我们或许可以用深度学习的相关技术来解决我们面临的问题。

利用深度学习的知识，对于随机控制问题中的策略函数映射，我们采用多层神经网络的结构将它形

式化的表达出来，而对于神经网络中本来应该通过反向传播算法进行优化的诸多参数[18]，我们利用进化算法来进行求解。也就是说我们人为设计了一个神经网络结构表示随机控制问题的策略函数，并且通过进化算法对网络进行优化。这样的做法优点在于我们只需通过评估网络输出的结果就可以更新网络参数，而不需要像传统的神经网络算法一样，设计损失函数的形式，利用随机梯度下降算法(Stochastic gradient descent, SGD)最优化损失函数来更新参数[17]。对于问题的余下部分，我们仍然沿用鄂维南和韩劼群的方法。

通过对有限时间指标的离散化，我们在每个时间步设置两个子网络，分别对随机控制问题的策略函数和解的梯度进行逼近。然后依据时间步的推进，将这两种类型的子网络有机的叠加在一起，形成一个非常深的网络，以此来求解我们的问题。

在本文中，我们首先对提出的算法进行了推导和架构，接着给出了整个算法的结构形式和主要部分的算法过程，并且将算法在 5-Riccati 方程[19] [20]和 12-维投资消费问题[21]中进行了测试，结果表明算法的提出具有一定的实际意义。

2. 数值方法

我们考虑一类经典的偏微分方程(PDEs)，它具有如下的表达形式：

$$\begin{aligned}\frac{\partial u}{\partial t}(t,x)+\underset{v\in\mathcal{U}}{optimal}\Big\{&\frac{1}{2}Tr\left(\sigma(t,x;v)\sigma^{\top}(t,x;v)\left(Hess_x u\right)(t,x)\right)\\&+\left\langle\left(\nabla_x u\right)(t,x),b(t,x;v)\right\rangle\\&+f\left(t,x,u(t,x),\sigma^{\top}(t,x;v)\left(\nabla_x u\right)(t,x);v\right)\Big\}=0\end{aligned}\tag{2.1}$$

其终端时刻条件为 $u(T,x)=h(x)$。

在这里，t 和 x 表示时间和 d-维的空间向量，v 表示在容许控制集 $\mathcal{U}$ 上取值的 m-维反馈控制过程，即问题的策略函数。此外，b 为 d-维向量值函数，σ 表示 $d\times d$ 维的矩阵值函数，$\sigma^{\top}$ 为 σ 的转置，∇u 和 $Hess_x u$ 表示函数 u 关于 x 的梯度和海森(Hessian)矩阵。

我们关心的是这个方程在 $t=0$ 时刻，$x=\xi$ 时，在最优控制 $\bar{v}\in\mathcal{U}$ 条件下的解。这里显然 $\xi\in\mathbb{R}^d$。

想要求解这个问题，最关键的一点是将上述的求解一个偏微分方程的问题重新表述，转化为求解一个随机控制问题。

2.1. 化偏微分方程问题(PDEs)为随机控制问题

假设 $(\Omega,\mathcal{F},\mathbb{P})$ 是一个概率空间，$\{W_t\}_{0\le t\le T}$ 是定义在这个概率空间上的 d-维标准布朗运动。$\{\mathcal{F}_t\}_{0\le t\le T}$ 是定义在 $(\Omega,\mathcal{F},\mathbb{P})$ 上，由布朗运动 $\{W_t\}_{0\le t\le T}$ 生成的自然域流；$\mathcal{A}$ 是所有具有连续轨道的 $\mathbb{R}^d$-维 $\mathcal{F}$-适应随机过程的集合。我们可以知道，反馈控制 v 属于集合 $\mathcal{U}$，其中 $\mathcal{U}\triangleq\left\{v\,|\,v\in\mathcal{L}^2_{\mathcal{F}}\left(0,T;\mathbb{R}^m\right)\right\}$ 在紧集 $U\in\mathbb{R}^m$ 上取值。

令 $\{X_t\}_{0\le t\le T}$ 为 d-维随机过程，我们关心的随机控制问题的状态方程可以表达为：

$$\begin{cases}\mathrm{d}X_t^{t,\xi;v}=b\left(t,X_t^{t,\xi;v};v_t\right)\mathrm{d}t+\sigma\left(t,X_t^{t,\xi;v};v_t\right)\mathrm{d}W_t\\X_0=\xi\in\mathbb{R}^d\end{cases}\tag{2.2}$$

这个随机控制问题的成本函数与一个和(2.2)式耦合的倒向随机微分方程(BSDE)的解相关：

$$\begin{cases}\mathrm{d}Y_t^{t,\xi;v}=-f\left(t,X^{t,\xi;v},Y^{t,x;v},Z^{t,\xi;v};v\right)+Z_t^{t,\xi;v}\mathrm{d}W_t\\Y_T^{t,\xi;v}=h\left(X_T^{t,\xi;v}\right)\end{cases}\tag{2.3}$$

成本函数(Cost function)定义为：

$$J(t,x;v)\triangleq Y_t^{t,\xi;v} \tag{2.4}$$

通过这个随机控制问题，我们可以找到在给定的(t,x)和最优的反馈控制$\bar{v}\in\mathcal{U}$等条件下，成本函数(2.4)的最大值或最小值。

在对f的合理假设下，我们可以发现对于所有的$t\in(0,T)$，偏微分方程(PDEs) (2.1)和倒向随机微分方程(BSDEs) (2.3)存在如下的关系[3] [4] [5]：

$$Y_t^{t,\xi;v}=u\left(t,X_t^{t,\xi;v}\right)\in\mathbb{R},\quad Z_t^{t,\xi;v}=\sigma^{\top}\left(t,X_t^{t,\xi;v};v\right)\left(\nabla_x u\right)\left(t,X_t^{t,\xi;v}\right)\in\mathbb{R}^d \tag{2.5}$$

(2.5)中的左式通常被称作为非线性的 Feynman-Kac 公式。

通过上述变换，我们将求解偏微分方程(PDEs) (2.1)的问题转化为求解一个与之相匹配的随机控制问题(2.2)~(2.4)。从而，通过求解随机控制问题(2.2)~(2.4)，我们可以利用$J(0,\xi)$来计算出(2.1)式在最优反馈控制$\bar{v}$下的值$u(0,\xi)$。

我们通过两个简短的步骤来说明求解$u(0,\xi)$的数值算法：

1) 在反馈控制$v\in\mathcal{U}$中，选定一条样本轨道$\hat{v}$，我们可以得到与之相对应的随机控制问题的解$J(0,\xi;\hat{v})$。并且，对于不同的样本轨道$\hat{v}$，我们总能够得到与之相对应的$J(0,\xi;\hat{v})$。因此，通过对反馈控制v的充分采样，我们能够得到充分多的$\hat{v}-J$对。

2) 对于 1 中的所有$\hat{v}-J$对，我们能够找到一个反馈控制$\bar{v}\in\hat{v}$，使得$J(0,\xi)$达到最大值或者最小值。

在下一章节中，我们首先讨论第 1 步的具体实施方法。

2.2. 偏微分方程(PDEs)的神经网络算法

为了推导出求解$J(0,\xi;\cdot)$的数值算法，我们假设在(2.2)~(2.4)中选取了反馈控制v的一条轨道$\hat{v}$，并将(2.5)中的两个公式插入到(2.3)中，我们可以得到对于所有的$t\in[0,T]$：

随机控制问题的状态方程改写为：

$$X_t=\xi+\int_0^t b\left(s,X_s;\hat{v}_s\right)\mathrm{d}s+\int_0^t\sigma\left(s,X_s;\hat{v}_s\right)\mathrm{d}W_s \tag{2.6}$$

与状态方程相耦合的倒向随机微分方程(BSDE)便为：

$$\begin{aligned}u(t,X_t)=h\left(X_T\right)&+\int_t^T f\left(s,X_s,u\left(s,X_s\right),\sigma^{\top}\left(s,X_s;\hat{v}_s\right)\left(\nabla_x u\right)\left(s,X_s\right);\hat{v}_s\right)\mathrm{d}s\\&-\int_t^T\left\langle\sigma^{\top}\left(s,X_s;\hat{v}_s\right)\left(\nabla_x u\right)\left(s,X_s\right),\mathrm{d}W_s\right\rangle\end{aligned} \tag{2.7}$$

成本函数为：

$$J(t,x;\hat{v})\triangleq u\left(t,X_t^{t,\xi;\hat{v}}\right) \tag{2.8}$$

我们将$u(0,\xi)\approx\theta_{u0}$，$\left(\nabla_x u\right)(0,\xi)\approx\theta_{\nabla u0}$视为模型的参数，然后对(2.6)，(2.7)式采取时间离散化。

具体来说，令$t_0,t_1,\cdots,t_N\in[0,T],N\in\mathbb{N}$，其满足：

$$0=t_0<t_1<\cdots<t_N=T \tag{2.9}$$

当$N\in\mathbb{N}$充分大时，对(2.6)、(2.7)式，我们考虑一种简单的欧拉格式(Euler scheme)：

对于$n=1,\cdots,N-1$，我们能够得到

$$X_{t_{n+1}}-X_{t_n}\approx b\left(t_n,X_{t_n};\hat{v}_{t_n}\right)\Delta t_n+\sigma\left(t_n,X_{t_n};\hat{v}_{t_n}\right)\Delta W_n \tag{2.10}$$

和

$$\begin{aligned} & u\left(t_{n+1}, X_{t_{n+1}}\right)-u\left(t_n, X_n\right) \\ & \approx-f\left(t_n, X_{t_n}, u\left(t_n, X_{t_n}\right), \sigma^{\top}\left(t_n, X_{t_n}\right)\left(\nabla_x u\right)\left(t_n, X_{t_n}\right) ; \hat{v}_{t_n}\right) \Delta t_n \\ & +\left\langle\sigma^{\top}\left(t_n, X_{t_n} ; \hat{v}_{t_n}\right)\left(\nabla_x u\right)\left(t_n, X_{t_n}\right), \Delta W_n\right\rangle \end{aligned} \tag{2.11}$$

其中

$$\Delta t_n=t_{n+1}-t_n, \quad \Delta W_n=W_{t_{n+1}}-W_{t_n} \tag{2.12}$$

在这样的时间离散化下，可以轻易地通过式(2.10)来采样获得轨道 $\left\{X_{t_n}\right\}_{0 \leq n \leq N}$。

我们通过在每个时间步 $t=t_n$ 使用多层反馈神经网络数值算法来逼近函数 $x \mapsto \sigma^{\top}(t, x)\left(\nabla_x u\right)(t, x)$ 的值[9]：

$$\sigma^{\top}\left(t_n, X_{t_n}\right)\left(\nabla_x u\right)\left(t_n, X_{t_n}\right) \approx\left(\sigma^{\top} \nabla_x u\right)\left(t_n, X_{t_n} \mid \theta_n\right) \tag{2.13}$$

其中 θ_n 表示网络在 $n=1, \cdots, N-1$ 时的参数。

如果已知 $\theta_{u 0}$ 的值，那么我们就可以近似地求解出(2.13)的值，然后将其代入(2.11)，再对(2.11)进行递归计算来求解出 $u\left(t_n, X_{t_n}\right), n \in\{1, \cdots, N\}$ 的值。

具体来说，这个方法是将轨道 $\left\{X_{t_n}\right\}_{0 \leq n \leq N}$ 和标准布朗运动 $\left\{W_{t_n}\right\}_{0 \leq n \leq N}$ 作为模型的输入数据，而 $\hat{u}\left(X_{t_n}, W_{t_n}\right), n \in[0, N]$ 则视为模型输出，它是 $u\left(t, X_t\right)$ 的近似解。

我们使用均方误差损失函数(Mean squared error, MSE)来衡量给定的终端条件 $h\left(X_{t_N}\right)$ 和计算出来的近似值 $\hat{u}\left(X_{t_N}, W_{t_N}\right)$ 之间的差距[18]：

$$L(\theta)=\mathbb{E}\left[\left|h\left(X_{t_N}\right)-\hat{u}\left(X_{t_N}, W_{t_N}\right)\right|^2\right] \tag{2.14}$$

整个网络的参数为 $\theta=\left\{\theta_{u 0}, \theta_{\nabla u 0}, \theta_1, \cdots, \theta_{N-1}\right\}$。

现在，我们可以使用随机梯度下降算法(SGD)，通过优化参数集 θ 来最小化均方误差损失函数(MSE)。对于使得均方误差损失函数(MSE)达到最小的参数集 θ，其中 $\theta_{u 0} \in \theta$ 是对 $u(0, \xi)$ 的近似，而 $u(0, \xi)$ 就是我们想要的 $J(0, \xi ; \hat{v})$ 的值，这里的 $\hat{v}$ 是我们对反馈控制 v 采样获得的一条轨道。因此我们就获得了一个 $\hat{v}-J$ 对。

我们使用伪代码的形式简略的呈现一下这个计算过程：

算法 1：深度神经网络算法

输入：　N：时间离散化的步数

P：神经网络的迭代步数

x：状态方程(2.2)的初始值

$\hat{v}$：反馈控制 v 的轨道

当神经网络的迭代步数 P 未满足时：

令时间步为：$n=0$

$X_n \leftarrow x$

$z_n \leftarrow \theta_{\nabla u0}$ /* $\theta_{\nabla u0} \in \theta$ */

$u_n \leftarrow \theta_{u0}$ /* $\theta_{u0} \in \theta$ */

当 $n < N-1$ 时：

$u_{n+1} \leftarrow u(t_n, X_n) - f(t_n, X_n, u_n, z_n, \hat{v}_n)\Delta t_n + \langle z_n, \Delta W_n \rangle$

$X_{n+1} \leftarrow X_n + b(t_n, X_n; \hat{v}_n)\Delta t_n + \sigma(t_n, X_n; \hat{v}_n)\Delta W_n$

$z_{n+1} \leftarrow X_{n+1}$ /* 通过神经网络获得 */

$n = n+1$

循环结束

返回：X_{N-1}，z_{N-1}，u_{N-1}

$u_N \leftarrow u(t_{N-1}, X_{N-1}) - f(t_{N-1}, X_{N-1}, u_{N-1}, z_{N-1}, \hat{v}_{N-1})\Delta t_{N-1} + \langle z_{N-1}, \Delta W_{N-1} \rangle$

$X_N \leftarrow X_{N-1} + b(t_{N-1}, X_{N-1}; \hat{v}_{N-1})\Delta t_{N-1} + \sigma(t_{N-1}, X_{N-1}; \hat{v}_{N-1})\Delta W_{N-1}$

终端时刻 u 的值：$h(X_N)$

$L(\theta) = \mathbb{E}\left[\left|h(X_N) - u_N\right|^2\right]$ /* 损失函数(2.14) */

$\theta \leftarrow \theta + \Delta\theta$ /* 反向传播更新参数 */

循环结束

返回：θ_{u_0}

在下一章中，我们讨论第 2 步的做法。

2.3. 反馈控制函数的进化算法

在 2.2 节中，对于反馈控制 v 采样获得的轨道 $\hat{v}$，我们可以获得与之对应的 $J(0, \xi; \hat{v})$，即 $\hat{v} - J$ 对。在这个章节中，我们会推导出一个数值算法，通过优化 $\hat{v}$ 来获得我们想要的最优解 $J(0, \xi)$。

我们可以将问题变形为：

$$V = \underset{\hat{v}}{optimal}\, J(0, \xi; \hat{v}) \tag{2.15}$$

其中 $V \in \mathbb{R}$ 就是我们的优化目标。我们将 $\hat{v} \in \mathbb{R}^m$ 视为方程的自变量，这个问题就可以看作是单目标优化问题(Single-Objective Optimization Problem)。

通常来说，在单目标优化问题中，可以利用进化算法(Evolution algorithm, EA)将自变量编码为种群，将目标函数映射为适应度函数，由此获得种群中每个个体相对应的适应度，接着以适应度为标准，通过若干代的变异、交叉和选择，算法可以得到最优的适应度，从而得到最优的目标函数值[22] [23]。

然而，在我们的问题中，自变量 $\hat{v}$ 并非数字而是一个随机过程。在这种情况下，我们不能简单的把 $\hat{v}$ 像数值一样进行编码来进行优化求解。

我们沿用 2.1 节中的假设条件，令 $t \in [0, T]$，$X_t \in \mathbb{R}^d$ (见(2.6)式)。反馈控制 v 的表达式我们定义为：

$$v := v(t, X_t) \tag{2.16}$$

其由 $\mathbb{R}^m$ 维均方可积 $\{\mathcal{F}_t\}_{0 \le t \le T}$-适应过程组成。

我们将时间离散化方法(2.9)应用于(2.16)，则随机过程可被离散化为：

$$v_{t_n} \coloneqq v\left(t_n, X_{t_n}\right) \tag{2.17}$$

其中 $n=0,\cdots,N-1$， X_{t_n} 的值可以由(2.10)得到。

假设映射 $x \mapsto v(t,x)$ 已经给定，那么我们可以很轻松地通过 t_n 和 X_{t_n} 计算出的 $v\left(t_n, X_{t_n}\right)$ 值。

然而，大多数的情况下我们并不知道映射 $x \mapsto v(t,x)$ 的形式，所以，我们需要找到一个方法表达出这个映射。

作为一种新兴的技术，深度神经网络模型在人工智能领域具有广泛的应用，并且它可以通过将简单函数进行复合来表达非常复杂的函数方程，所以，我们可以利用神经网络模型的这种性质来表达出映射 $x \mapsto v(t,x)$。

首先，对于任意的 $n=0,\cdots,N-1$ 和 X_{t_n}，我们有：

$$v'\left(X_{t_n}\right) \approx f_n^{(M)}\left(f_n^{(M-1)}\left(\cdots\left(f_n^{(2)}\left(f_n^{(1)}\left(X_{t_n}\right)\right)\right)\right)\right) \tag{2.18}$$

这里的 M 表示多层神经网络的层数， $f_n^{(m)}$ 则表示从第 $m-1$ 层到第 m 层的映射，其中 $m=1,\cdots,M$。需要注意的是， $m=0$ 时表示输入层[17] [18]。

当 $m=1,\cdots,M-1$ 时，假设 $h_n^{m-1} \in \mathbb{R}^{\rho_n^m}$ 和 $h_n^m \in \mathbb{R}^{k_n^m}$ 分别是函数 $f_n^{(m)}$ 的自变量和因变量，那么 $f_n^{(m)}$ 的表达式可以定义为：

$$h_n^m = g\left(K_n^{m,\top} h_n^{m-1} + b_n^m\right) \tag{2.19}$$

这里的权重矩阵 $K_n^m \in \mathbb{R}^{\rho_n^m \times k_n^m}$ 和偏置 $b_n^m \in \mathbb{R}^{k_n^m}$ 均为函数的参数。其中 $g(\cdot)$ 为激活函数，它能够对输入的自变量中的每个元素按照给定的规则产生作用[18]。并且，在(2.19)中得到的 h_n^m 会作为下一层函数 $f_n^{(m+1)}$ 的自变量传递下去。

需要注意的是输出层函数 $f_n^{(M)}$ 不需要使用激活函数。

为了方便记录，我们用 ϕ_n^m 表示函数 $f_n^{(m)}$ 的参数 $\left(K_n^m, b_n^m\right)$，因此，在 $t=t_n$ 时刻时(2.18)式的全部参数为 $\phi_n = \left\{\phi_n^1,\cdots,\phi_n^M\right\}$。

因此，当 $n=0,\cdots,N-1$ 和 X_{t_n} 时，我们可以将式改写为：

$$v'\left(X_{t_n}\right) = K_n^{M,\top} g\left(g\left(\cdots g\left(g\left(X_{t_n};\phi_n^{(1)}\right);\phi_n^{(2)}\right);\phi_n^{(M-2)}\right);\phi_n^{(M-1)}\right) + b_n^M \tag{2.20}$$

其参数为 $\phi_n = \left\{\phi_n^1,\cdots,\phi_n^M\right\}$。

同样的，为了方便记录，我们把(2.20)式缩写为 $v'\left(X_{t_n}\right) = f_n\left(X_{t_n};\phi_n\right)$， $t=t_n$。

显然，这个网络以 X_{t_n} 作为输入数据，并且以 $v'\left(X_{t_n}\right)$ 作为输出数据，而 $v'\left(X_{t_n}\right)$ 则是 $v\left(t_n, X_{t_n}\right)$ 的一个近似。在这里我们指出，我们得到的 v' 就是 2.2 节中我们采样获得的轨道 $\hat{v}$。

由此可知，当 $n \in \{0,\cdots,N-1\}$ 时， $v\left(t_n, X_{t_n};\Phi\right) \in \mathbb{R}^m$ 的值可以由(2.20)式和(2.10)式联合起来递归地计算得出，其中 $\Phi = \left\{\phi_0,\cdots,\phi_{N-1}\right\}$ 表示函数 v 的全体参数。因此，(2.15)式的自变量就可以看成是 Φ，我们的优化问题就变形为： $V = \underset{\Phi}{optimal} J\left(0,\xi;\hat{v}\left(X;\Phi\right)\right)$。

现在我们就可以使用进化算法[23] [24] (EA)通过 Φ 来优化 V 的值，就像使用进化算法来解决一个经典的单目标优化问题一样。

我们同样的使用伪代码的形式简略的呈现一下这个计算过程

算法 2：进化算法

输入：

F：比例因子

Cr：交叉概率

NP：种群规模

$V=\underset{\Phi}{optimal}\,J\left(0,\xi;\hat{v}\left(X;\Phi\right)\right)$：目标函数

D：Φ 中的元素个数

令进化代数 $G=0$。对种群 $P_G=\left\{\boldsymbol{X}_{1,G},\cdots,\boldsymbol{X}_{NP,G}\right\}$ 进行随机初始化，其中 $\boldsymbol{X}_{r,G}=\left[x_{1,r,G},\cdots,x_{D,r,G}\right]$，$r\in\left[1,\cdots,NP\right]$ 表示种群中的个体，它的每个元素都服从范围 $\left[\boldsymbol{X}_{\min},\boldsymbol{X}_{\max}\right]$ 的均匀分布。

当停止条件不满足时

从 $r=1$ 到 NP 有

对于种群中的每个个体来说

由种群中第 r 个向量 $\boldsymbol{X}_{r,G}$ 生成相应的变异向量 $\boldsymbol{V}_{r,G}$

$$\boldsymbol{V}_{r,G}=\boldsymbol{X}_{r_{c_1},G}+F\cdot\left(\boldsymbol{X}_{r_{c_2},G}-\boldsymbol{X}_{r_{c_3},G}\right),\quad c_1,c_2,c_3\in\left[1,NP\right]$$

再由变异向量 $\boldsymbol{V}_{r,G}$ 和向量 $\boldsymbol{X}_{r,G}$ 联合生成交叉向量 $\boldsymbol{U}_{r,G}=\left\{u_{1,r,G},\cdots,u_{D,r,G}\right\}$

$$u_{r,l,G}=\begin{cases}v_{r,l,G} & \text{if } \left(random_{r,l}\left[0,1\right]\le Cr \text{ or } l=l_{random}\in\left[1,\cdots,G\right]\right)\\ x_{r,l,G} & \text{otherwise}\end{cases}$$

/* l_{random} 能够确保 $\boldsymbol{U}_{r,G}$ 中至少有个分量来自 $\boldsymbol{V}_{r,G}$ */

返回 $\boldsymbol{U}_{r,G},\boldsymbol{X}_{r,G}$

按照 $\Phi=\left\{\phi_0,\cdots,\phi_{N-1}\right\}$ 的形状，将 $\boldsymbol{U}_{r,G}$ 和 $\boldsymbol{X}_{r,G}$ 进行变形[25,26]，即 $\boldsymbol{U}_{r,G}\mapsto\Phi_1$，$\boldsymbol{X}_{r,G}\mapsto\Phi_2$

如果 $J\left(0,\xi;\hat{v}\left(X;\Phi_1\right)\right)\le J\left(0,\xi;\hat{v}\left(X;\Phi_2\right)\right)$

$\boldsymbol{X}_{r,G+1}=\boldsymbol{U}_{r,G}$

否则

$\boldsymbol{X}_{r,G+1}=\boldsymbol{X}_{r,G}$

结束

结束

进化代数 $G=G+1$

结束

3. 偏微分方程和随机控制的数值算例

在本节当中，我们通过几个具有实际意义的偏微分方程(PDEs)算例来说明第二章 2.2 节和 2.3 节推导出的算法。在接下来的例子之中，我们会对 2.2 节中的近似计算使用小批量(mini-batches)样本的 Adam 优化器[17] [18]，对 2.3 节中 V 的优化我们采用一种改进的进化算法来进行计算[24]。

如图 1，在我们的实践过程中，对于 2.3 节，我们采用 N 个全连接反馈神经网络的结构形式来表示 $\hat{v}_n^{\Phi}$，其中 $\Phi=\left\{\phi_0,\cdots,\phi_{N-1}\right\},n\in\left\{0,1,\cdots,N-1\right\}$。每一个网络都表示在 $t=t_n$ 时刻 $X_{t_n}\mapsto\hat{v}_n$ 的映射，它包括一个输入层(d-维)，H 个隐藏层(具有相同的维数)以及一个输出层(m 维)。这些网络的所有权重都服从均匀分布。此外，我们利用 $N-1$ 个多层神经网络来计算 $\left(\sigma^{\top}\nabla_x u\right)\left(t_n,X_{t_n}\mid\theta_n\right),n\in\left\{1,2,\cdots,N-1\right\}$，其中 θ_n 表示神经网络在 $t=t_n$ 时刻的参数[8]，在这里，神经网络的参数通过正态分布或者均匀分布完成初始化。整个网络结

构有$(D+1)(N-1)+(H+1)N$层的若干参数需要优化。

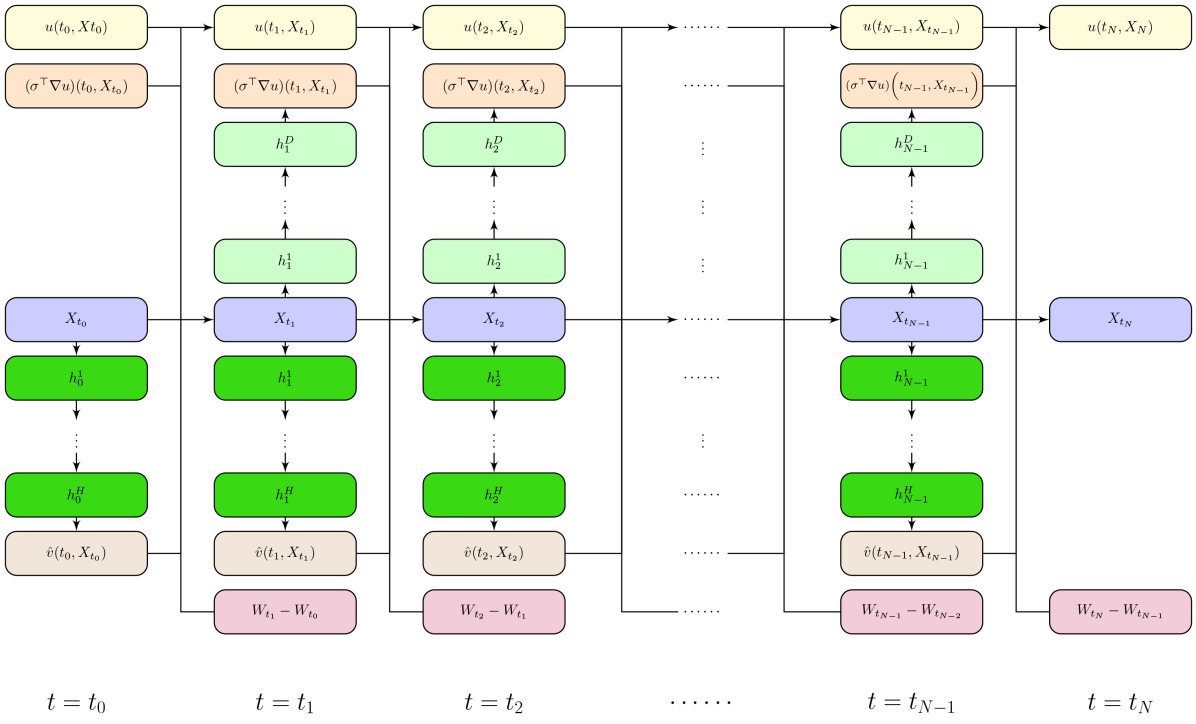

Figure 1. Numerical algorithm structure of stochastic control problems
图 1. 随机控制问题的数值算法结构图

3.1. 倒向随机 Riccati 方程

在本节当中，我们将推导出来的算法应用在倒向随机 Riccati 方程[19] [20]中来测试它的数值效果。令$t \in [0,T]$，$x, W \in \mathbb{R}^d$，$v \in \mathbb{R}^m$，$u \in \mathbb{R}$，$z \in \mathbb{R}^{1\times d}$。当$d=2$，$m=1$，$\xi = (1,\cdots,1)$时，有(2.2)式中

$$b(t, x_t; v_t) = (Ax_t + Bv_t) = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix} x_t + \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} v_t \tag{3.1}$$

$$\begin{aligned} \sigma(t, x_t; v_t)\mathrm{d}W_t &= \sum_{i=1}^{d} \left(C^i x_t + D^i v_t\right)\mathrm{d}W_t^i \\ &= \left[\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 0.1 \end{pmatrix} v_t\right]\mathrm{d}W_t^1 \\ &\quad + \left[\begin{pmatrix} 0.1 & 0 \\ 0 & 0 \end{pmatrix} x_t + \begin{pmatrix} 0.1 \\ 0 \end{pmatrix} v_t\right]\mathrm{d}W_t^2 \end{aligned} \tag{3.2}$$

同样地，在(2.3)式中

$$f(t, x, u, z; v) = \left(\langle Qx_t, x_t\rangle + \langle Nx_t, x_t\rangle\right) = x_t^\top \begin{pmatrix} 0.5 & 0 \\ 0 & 0.3 \end{pmatrix} x_t + 0.1 v_t^\top v_t \tag{3.3}$$

并且它的终端时刻条件我们定义为$h(x) = \langle Mx, x\rangle = x^\top \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix} x$。

此时，当$t \in [0,T]$，$x \in \mathbb{R}^d$，使得$u(T,x) = h(x)$时，偏微分方程(PDEs) (2.1)的解u满足：

$$\frac{\partial u}{\partial t}(t,x)+\inf_{v}\left\{\frac{1}{2}\sum_{\alpha,\beta}\sum_{i=1}^{d}\left(C_{\alpha}^{i}x+D_{\alpha}^{i}v_{t}\right)\left(C_{\beta}^{i}x+D_{\beta}^{i}v_{t}\right)\left(D_{\alpha,\beta}u\right)(t,x)\right.$$
$$\left.+\sum_{\gamma}\left(A_{\gamma}x+B_{\gamma}v_{t}\right)\left(D_{\gamma}u\right)(t,x)+x^{\top}Qx+v_{t}^{\top}Nv_{t}\right\}=0 \tag{3.4}$$

Table 1. Numerical simulation of 2-dimensional Riccati equation
表 1. Riccati 方程在 2-维形势下的数值模拟

进化代数 g	当前代最优 $u(0,\xi)$	$u(0,\xi)$ 均值	$u(0,\xi)$ 标准差
1	1.52320	1.72797	0.10471
20	1.31463	1.33062	0.00932
40	1.29945	1.30080	0.00102
60	1.29622	1.29698	0.00035
80	1.29590	1.29661	0.00037
100	1.29582	1.29688	0.00028

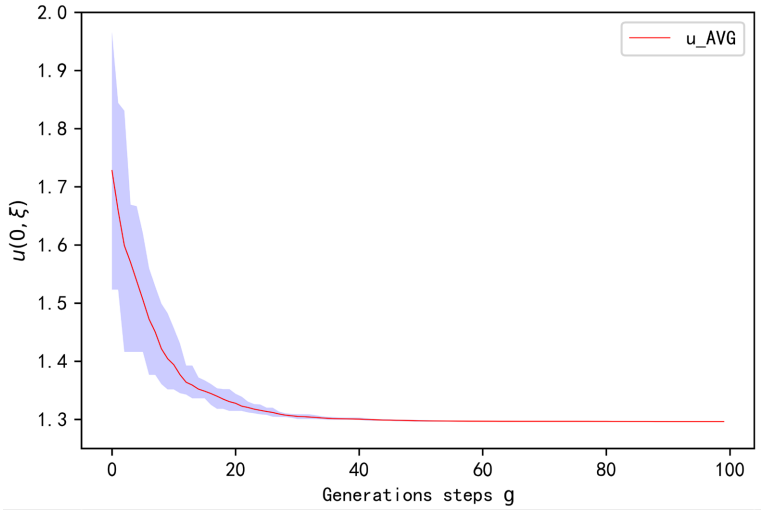

Figure 2. Solution of 2-dimensional Riccati equation against generations steps g
图 2. 2-维 Riccati 方程的解随进化代数 g 的变化

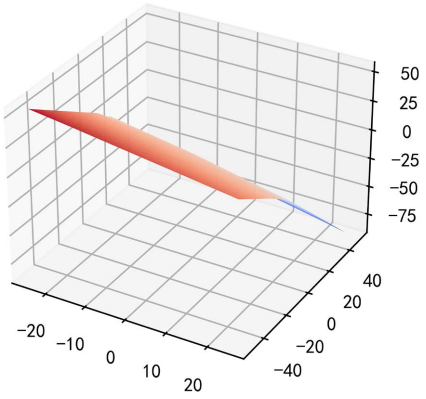

Figure 3. Value of function $f_n(x;\phi_n)$ against x, where $n\in\{0,1,\cdots,N-1\}$
图 3. 函数 $f_n(x;\phi_n)$ 随 x 的变化，其中 $n\in\{0,1,\cdots,N-1\}$

在表 1 中，我们令进化算法的进化代数为 100 代，每一代的种群规模 $NP=20$。在进化过程中，我们计算了每一代的最优 $u(0,\xi)$ 值，相应规模下 $u(0,\xi)$ 的均值及标准差。图 2 为 2-维 Riccati 方程的解 $u(t=0,\xi=(1,1))$ 在 20 个等距时间步的离散下随进化代数 g 的变化情况。图中的阴影部分表示每一代种群中 $\max u(0,\xi)$ 和 $\min u(0,\xi)$ 之间的差值。红色的线表示 $u(0,\xi)$ 的均值。在 100 代 9541.29 秒的进化后，该数值方法获得的解的标准差为 2.87E−04。图 3 为函数 $f_n(x;\phi_n)$，$n\in\{0,1,\cdots,N-1\}$，其中 ϕ_n 为参数，由我们在 2.2 节、2.3 节推导出的算法求解得出，此时，$x\in\mathbb{R}^2$，$x_1\in\{-20,\cdots,20\}, x_2\in\{-40,\cdots,40\}$。

在图 2 关于 $u(0,\xi)$ 的计算中，由我们推导出的算法可知，此时偏微分方程(3.4)的真实解可由 1.29581 代替。

此外，u 还可以通过 $u_t=x_t^{\top}K_tx_t$ 来计算获得[20]，其中 K_t 的值可以通过下列方程组配合倒向欧拉算法[27] (Backward Euler method)解出来

$$\begin{cases}\mathrm{d}K_t=A^{\top}K_t+K_tA+Q+\sum_{i=1}^{d}\left(C^i\right)^{\top}K_tD^i\\ \qquad -\left[K_tB+\sum_{i=1}^{d}\left(C^i\right)^{\top}K_tD^i\right]\left[N+\sum_{i=1}^{d}\left(D^i\right)^{\top}K_tD^i\right]^{-1}\left[K_tB+\sum_{i=1}^{d}\left(C^i\right)^{\top}K_tD^i\right]^{\top}\\ K_T=M\end{cases}\tag{3.5}$$

在这种方法下，我们有 $u(0,\xi)=1.29538$，这与我们所推导出来的算法之间的误差大小为 0.00043。

更进一步，我们考虑当 $d=5$，$m=5$，$\xi=(1,\cdots,1)$ 时的情况，令(2.2)式中 b_t 为：

$$b(t,x_t;v_t)=\left(Ax_t+Bv_t\right)=\begin{pmatrix}0.1&0&0&0&0\\0&0.1&0&0&0\\0&0&0.3&0&0\\0&0&0&0.1&0\\0&0&0&0&0.1\end{pmatrix}x_t+\begin{pmatrix}0.1&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix}v_t\tag{3.6}$$

σ_t 见后文附录。

此时，有(2.3)中函数 f 为：

$$\begin{aligned}f(t,x,u,z,v)&=\left(\langle Qx_t,x_t\rangle+\langle Nx_t,x_t\rangle\right)\\&=x_t^{\top}\begin{pmatrix}0.01&0&0&0&0\\0&0.1&0&0&0\\0&0&0.02&0&0\\0&0&0&0.1&0\\0&0&0&0&0.02\end{pmatrix}x_t+v_t^{\top}\begin{pmatrix}0.01&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0.01&0\\0&0&0&0&0\end{pmatrix}v_t\end{aligned}\tag{3.7}$$

相应地，终端时刻条件为 $h(x)=\langle Mx,x\rangle=x^{\top}\begin{pmatrix}0.02&0&0&0&0\\0&0.12&0&0&0\\0&0&0.12&0&0\\0&0&0&0.2&0\\0&0&0&0&0.2\end{pmatrix}x$。

在表 2 中，我们令进化算法的进化代数为 150 代，每一代的种群规模 $NP=30$。在进化过程中，我们计算了每一代的最优 $u(0,\xi)$ 值，相应规模下 $u(0,\xi)$ 的均值及标准差。图 4 为 5-维 Riccati 方程的解 $u(t=0,\xi=(1,\cdots,1))$ 在 20 个等距时间步的离散下随进化代数 g 的变化情况。图中的阴影部分表示每一代

种群中 $\max u(0,\xi)$ 和 $\min u(0,\xi)$ 之间的差值。红色的线表示 $u(0,\xi)$ 的均值。在 150 代 24699.45 秒的进化后，该数值方法获得的解的标准差为 1.01E−03。图 5 为函数 $f_n(x;\phi_n)$，$n\in\{0,1,\cdots,N-1\}$，其中 ϕ_n 为参数，由我们在 2.2 节、2.3 节推导出的算法求解得出，此时，$x\in\mathbb{R}^5$，$x_1\in\{-20,\cdots,20\}$, $x_2\in\{-40,\cdots,40\}$，$x_3,x_4,x_5\in\{1,\cdots,1\}$。

Table 2. Numerical simulation of 5-dimensional Riccati equation
表 2. Riccati 方程在 5-维形势下的数值模拟

进化代数 g	当前代最优 $u(0,\xi)$	$u(0,\xi)$ 均值	$u(0,\xi)$ 标准差
1	1.73978	2.41715	0.37363
30	1.21487	1.22176	0.00426
60	1.20480	1.20916	0.00203
90	1.20419	1.20681	0.00149
120	1.20332	1.20598	0.00127
150	1.20332	1.20535	0.00101

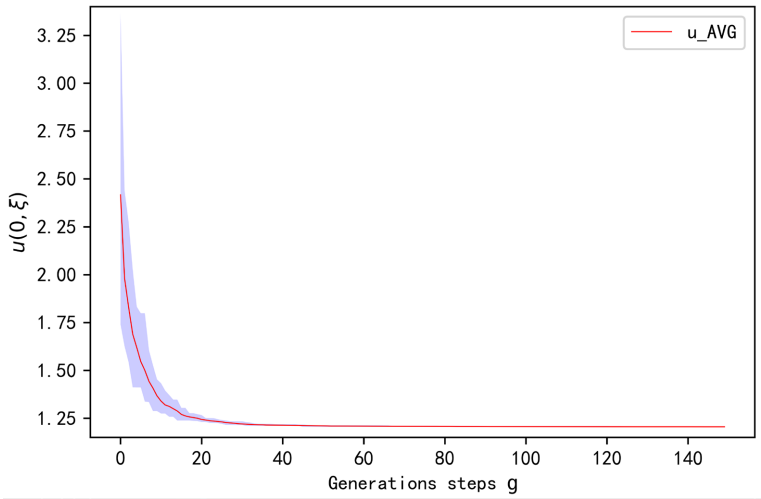

Figure 4. Solution of 5-dimensional Riccati equation against generations steps g
图 4. 5-维 Riccati 方程的解随进化代数 g 的变化

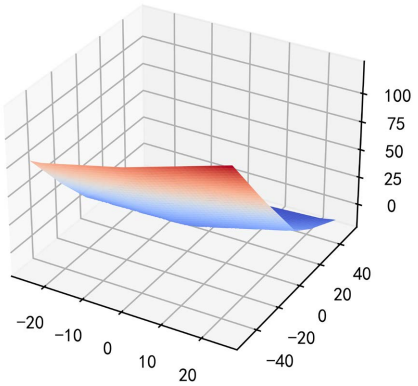

Figure 5. Value of function $f_n(x;\phi_n)$ against x, where $n\in\{0,1,\cdots,N-1\}$
图 5. 函数 $f_n(x;\phi_n)$ 随 x 的变化，其中 $n\in\{0,1,\cdots,N-1\}$

在图 4 关于 5-维 Riccati 方程的解$u(0,\xi)$的计算中，我们用 1.20332 来代替它真实值。此时，利用$u_t = x_t^{\top} K_t x_t$和(3.5)式可以得出$u(0,\xi)$值为 1.19665，两种算法之间的误差为 0.00667。

3.2. 基于 CIR 模型和随机波动率的最优投资消费问题

在这一节当中，我们将算法应用在投资消费问题中。此时，将问题放在一个没有税收和交易成本，但具有 CIR 利率[28]和随机波动率的金融市场中[21]。

需要注意的是的在这个问题当中，偏微分方程(2.1)的反馈控制过程 v 将不再只是一个，而是由一对随机过程来担任，我们用(π, C)来表示[21]，其中π表示股票投资额，C表示消费利率。

一般的，有$t \in [0,T]$，$x \in \mathbb{R}^d$，$W \in \mathbb{R}^{2d}$，$\pi, C \in \mathbb{R}^m$，$r, \eta \in \mathbb{R}^d$，$u \in \mathbb{R}$，$z \in \mathbb{R}^{1\times d}$，此时，我们令$T=1$，$d=1$，$m=1$，$\xi=30$，则在中(2.2)有

$$b(t, r_t, \eta_t, x_t; \pi_t, c_t) = r_t x_t + \pi_t k \eta_t - c_t = r_t x_t + 0.6\pi_t \eta_t - c_t \tag{3.8}$$

$$\sigma(t, \eta_t, x_t; \pi_t) \mathrm{d}W_t^2 = \pi_t \sigma_1 \sqrt{\eta_t} \mathrm{d}W_t^2 = 1.2\pi_t \sqrt{\eta_t} \mathrm{d}W_t^2 \tag{3.9}$$

同时，r，η都是随机过程，且它们的形式由 CIR 模型给出

$$\begin{cases} \mathrm{d}r_t = (0.25 - 0.15 r_t)\mathrm{d}t + 0.2\sqrt{r_t}\mathrm{d}W_t^1 \\ r(0) = 0.05 \in \mathbb{R}^d \end{cases} \tag{3.10}$$

$$\begin{cases} \mathrm{d}\eta_t = (0.85 - 0.6\eta_t)\mathrm{d}t + 0.25\sqrt{\eta_t}\mathrm{d}W_t^2 \\ \eta(0) = 0.36 \in \mathbb{R}^d \end{cases} \tag{3.11}$$

在(2.3)式中，我们有$f(t,x,u,z;c) = \alpha \frac{c_t^{\delta}}{\delta} \mathrm{e}^{-\beta t} = 0.4 \frac{c_t^{0.1}}{0.1} \mathrm{e}^{-0.1t}$，并且此时的终端时刻条件为$h(x) = (1-\alpha)\frac{x_T^{\delta}}{\delta} \mathrm{e}^{-\beta T} = 0.6 \frac{x_T^{0.1}}{0.1} \mathrm{e}^{-0.1T}$。

在这种规定之下，我们可知偏微分方程(2.1)在$t \in [0,T]$，$x \in \mathbb{R}^d$时的解 u 满足：

$$\begin{aligned} &\frac{\partial u}{\partial t} + \max_{\pi, C} \Big\{ (r_t x + \pi_t k \eta_t - C_t)(D_x u) + \frac{1}{2}\pi_t^2 \sigma_1^2 \eta_t (D_x^2 u) + (\theta - C_t r_t)(D_r u) \\ &+ \frac{1}{2}\sigma_0^2 r_t (D_r^2 u) + (b - a\eta_t)(D_\eta u) + \frac{1}{2}\sigma_2^2 \eta_t (D_\eta^2 u) + \pi_t \sigma_1 \sigma_2 (D_{x\eta} u) + \alpha \mathrm{e}^{-\beta t} \frac{C_t^{\delta}}{\delta} \Big\} = 0 \end{aligned} \tag{3.12}$$

Table 3. Numerical simulation of 3-dimensional Investment and Consumption problem
表 3. 投资消费问题在 3-维形式下的数值模拟

进化代数 g	最优 $u(0,\xi)$	$u(0,\xi)$ 均值	$u(0,\xi)$ 标准差
1	12.0452	11.9905	0.0291071
100	12.1648	12.1635	0.0006916
200	12.1701	12.1687	0.0005018
300	12.1712	12.1704	0.0003169
400	12.1718	12.1709	0.0003175

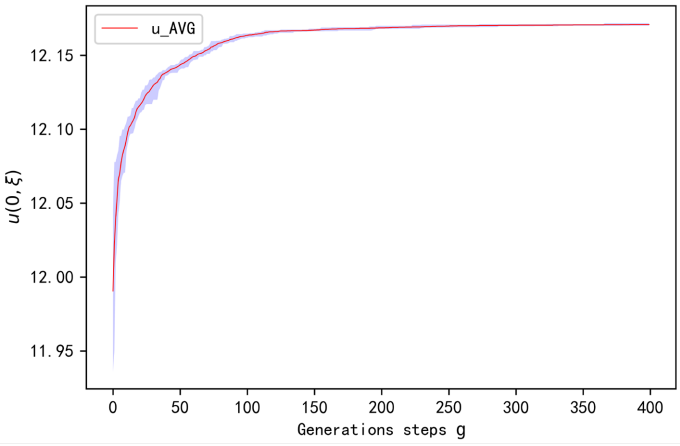

Figure 6. Solution of 3-dimensional Investment and Consumption problem against generations steps g
图 6. 3-维投资消费问题的解随进化代数 g 的变化

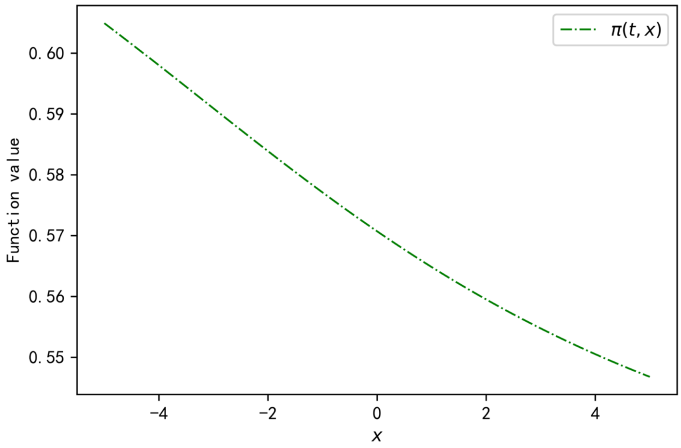

Figure 7. Value of function $f_{n,\pi}\left(x;\phi_{n,\pi}\right)$ against x, where $n\in\{0,1,\cdots,N-1\}$
图 7. 函数 $f_{n,\pi}\left(x;\phi_{n,\pi}\right)$ 随 x 的变化，其中 $n\in\{0,1,\cdots,N-1\}$

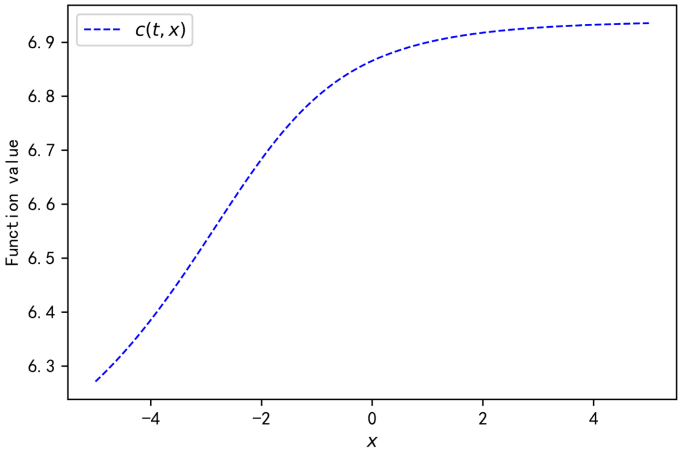

Figure 8. Value of function $f_{n,C}\left(x;\phi_{n,C}\right)$ against x, where $n\in\{0,1,\cdots,N-1\}$
图 8. 函数 $f_{n,C}\left(x;\phi_{n,C}\right)$ 随 x 的变化，其中 $n\in\{0,1,\cdots,N-1\}$

在表 3 中，我们令进化算法的进化代数为 400 代，每一代的种群规模 $NP=30$。在进化过程中，我们计算了每一代的最优 $u(0,\xi)$ 值，相应规模下 $u(0,\xi)$ 的均值及标准差。图 6 为 3-维投资消费问题(3.12)的解 $u(t=0,\xi=30)$ 在 20 个等距时间步的离散下随进化代数 g 的变化情况。图中的阴影部分表示每一代种群中 $\max u(0,\xi)$ 和 $\min u(0,\xi)$ 之间的差值。红色的线表示 $u(0,\xi)$ 的均值。在 400 代 40660.70 秒的进化后，该数值方法获得的解的标准差为 3.17596E−04。图 7 为函数 $f_{n,\pi}(x;\phi_{n,\pi})$，$n\in\{0,1,\cdots,N-1\}$，其中 $\phi_{n,\pi}$ 为映射 $x_{t_n}\mapsto\pi(t_n,x_{t_n})$ 中的参数，由我们在 2.2 节、2.3 节推导出的算法求解得出，自变量 $x\in\{-5,\cdots,5\}\in\mathbb{R}$；同样的，图 8 为函数 $f_{n,C}(x;\phi_{n,C})$。

在图 6 关于 $u(0,\xi)$ 的计算中，我们有偏微分方程(3.12)在 $t=0$，$r(0)=0.05$，$\eta(0)=0.36$，$\xi=30$ 真实解可由 12.1718 代替。

更进一步，我们考虑当 $\eta\in\mathbb{R}^{10}$，$W\in\mathbb{R}^{11}$ 时的情况。此时令 $\xi=25$，则有(2.2)中

$$b(t,r_t,\eta_t,x_t;\pi_t,c_t)=r_tx_t+\pi_t\langle k,\eta_t\rangle-c_t \tag{3.13}$$

$$\sigma(t,\eta_t,x_t;\pi_t)\mathrm{d}W_t=\pi_t\sum_{i=1}^{10}\sigma_1\sqrt{\eta_t^i}\mathrm{d}W_t^i \tag{3.14}$$

这里我们令 $k=\begin{pmatrix}0.6\\0.55\\0.5\\0.48\\0.58\\0.62\\0.4\\0.62\\0.52\\0.75\end{pmatrix}$，$\sigma_1=\begin{pmatrix}0.45\\0.62\\0.75\\0.5\\0.82\\0.64\\1.0\\0.82\\0.64\\0.8\end{pmatrix}$。

此时，由于随机过程 $\{\eta_t\}_{0\le t\le T}$ 为 10-维，所以它的形式变为

$$\begin{cases}\mathrm{d}\eta_t^1=\left(b^1-a^1\eta_t^1\right)\mathrm{d}t+\sigma_2^1\sqrt{\eta_t^1}\mathrm{d}W_t^1\\ \quad\vdots\\ \mathrm{d}\eta_t^{10}=\left(b^{10}-a^{10}\eta_t^{10}\right)\mathrm{d}t+\sigma_2^{10}\sqrt{\eta_t^{10}}\mathrm{d}W_t^{10}\\ \eta(0)=\eta_0\in\mathbb{R}^{10}\end{cases} \tag{3.15}$$

这个时候方程的初始时刻条件 η_0 我们定义为 $\eta_0=\begin{pmatrix}0.13\\0.25\\0.32\\0.37\\0.18\\0.42\\0.2\\0.15\\0.1\\0.12\end{pmatrix}$。需要指出的是，此时方程中的参数均为向

量形式，其具体形式为 $b=\begin{pmatrix}0.72\\0.78\\0.58\\0.75\\0.8\\0.58\\0.55\\0.65\\0.62\\0.58\end{pmatrix}$，$a=\begin{pmatrix}0.18\\0.22\\0.32\\0.18\\0.28\\0.3\\0.2\\0.17\\0.18\\0.24\end{pmatrix}$，$\sigma_2=\begin{pmatrix}0.13\\0.25\\0.12\\0.14\\0.22\\0.16\\0.14\\0.32\\0.15\\0.12\end{pmatrix}$。而 $b^1\cdots b^{10}$，$a^1\cdots a^{10}$，$\sigma_2^1\cdots\sigma_2^{10}$ 则为对应向量的相应分量。

需要指出的是，这时(3.13)，(3.14)，(3.15)式中的参数 k，σ_1，b，a，σ_2 及 $\{\eta_t\}_{0\le t\le T}$ 的初始时刻条件 η_0 均为向量，我们在附录中给出它们的具体值。在这样的设定下，我们令(2.3)式函数 f 的参数 $\alpha=0.3$，$\delta=0.08$，并且保持其它的所有参数按照 3-维情况下给出的各个值保持不变。

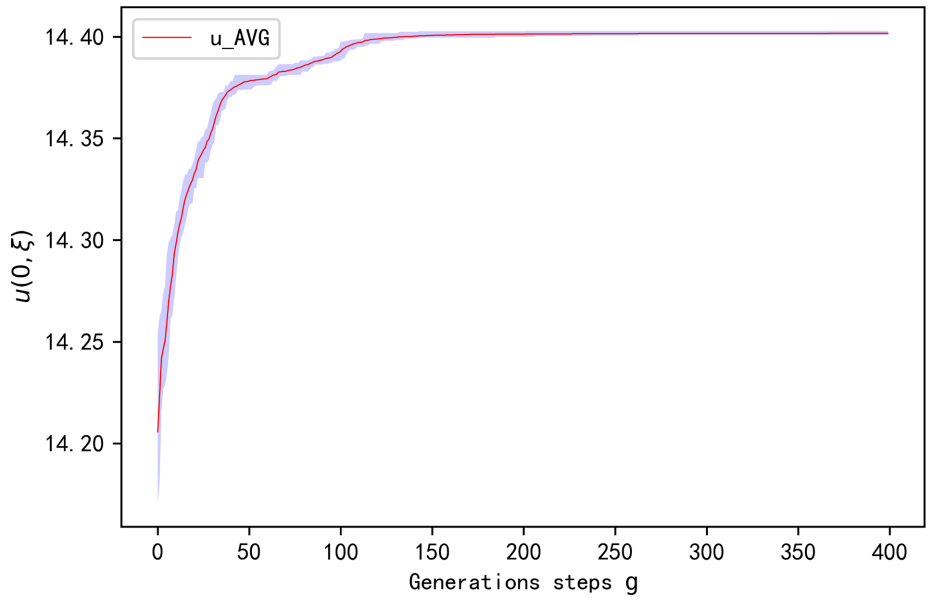

Figure 9. Solution of 12-dimensional Investment and Consumption problem against generations steps g
图 9. 12-维投资消费问题的解随进化代数 g 的变化

Table 4. Numerical simulation of 12-dimensional Investment and Consumption problem
表 4. 投资消费问题在 12-维形式下的数值模拟

进化代数 g	最优 $u(0,\xi)$	$u(0,\xi)$ 均值	$u(0,\xi)$ 标准差
1	14.2538	14.2057	0.0228500
100	14.3947	14.3922	0.0013374
200	14.4027	14.4013	0.0006898
300	14.4029	14.4016	0.0005776
400	14.4029	14.4017	0.0005779

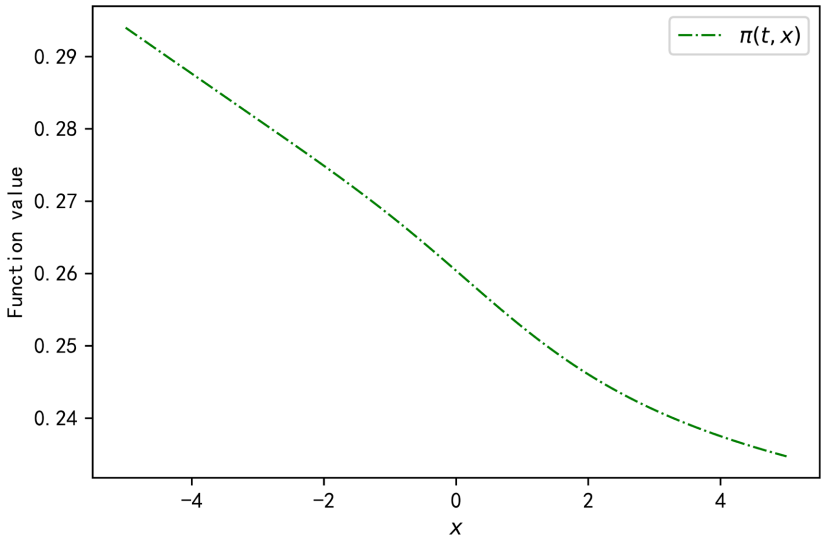

Figure 10. Value of function $f_{n,\pi}\left(x;\phi_{n,\pi}\right)$ against x, where $n\in\{0,1,\cdots,N-1\}$

图 10. 函数 $f_{n,\pi}\left(x;\phi_{n,\pi}\right)$ 随 x 的变化，其中 $n\in\{0,1,\cdots,N-1\}$

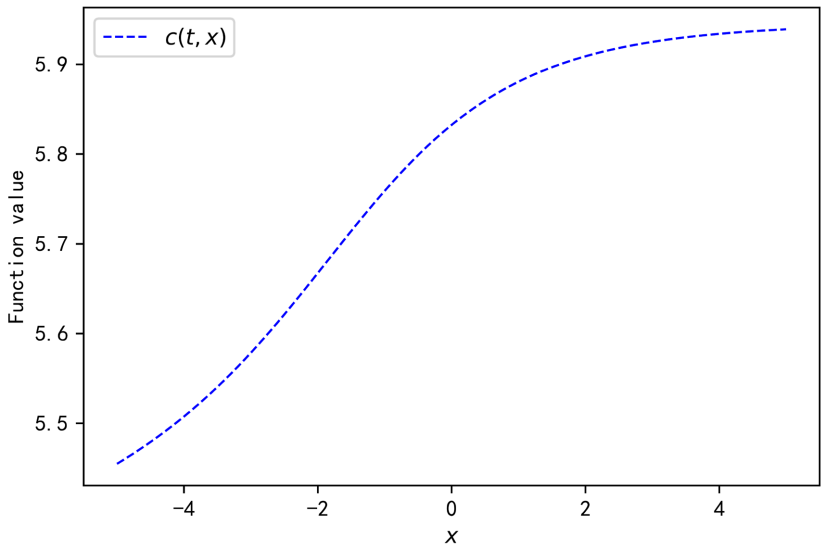

Figure 11. Value of function $f_{n,C}\left(x;\phi_{n,C}\right)$ against x, where $n\in\{0,1,\cdots,N-1\}$

图 11. 函数 $f_{n,C}\left(x;\phi_{n,C}\right)$ 随 x 的变化，其中 $n\in\{0,1,\cdots,N-1\}$

在表 4 中，我们令进化算法的进化代数为 400 代，每一代的种群规模 $NP=30$。在进化过程中，我们计算了每一代的最优 $u(0,\xi)$ 值，相应规模下 $u(0,\xi)$ 的均值及标准差。图 9 为 12-维投资消费问题(3.12)的解 $u(t=0,\xi=25)$ 在 20 个等距时间步的离散下随进化代数 g 的变化情况。图中的阴影部分表示每一代种群中 $\max u(0,\xi)$ 和 $\min u(0,\xi)$ 之间的差值。红色的线表示 $u(0,\xi)$ 的均值。在 400 代 44232.06 秒的进化后，该数值方法获得的解的标准差为 5.77984E−04。图 10 为函数 $f_{n,\pi}\left(x;\phi_{n,\pi}\right)$，其中 $\phi_{n,\pi}$ 为映射 $x_{t_n}\mapsto\pi\left(t_n,x_{t_n}\right)$ 中的参数，由我们在 2.2 节、2.3 节推导出的算法求解得出，自变量 $x\in\{-5,\cdots,5\}\in\mathbb{R}$；同样的，图 11 为函数 $f_{n,C}\left(x;\phi_{n,C}\right)$。

在图 9 关于 $u(0,r(0),\eta_0,\xi)$ 的计算中，我们将它的真实解由 14.4029 代替。

4. 结论

本文通过提出一种新型的算法，避开求解值函数，能够普适性的解决带有策略函数的线性、非线性随机控制问题，获得足够精确的数值解。在实际算例的背景下，显示出算法能够求解工程、金融等领域的相关问题，具有一定的实际意义。由于使用深度学习算法和进化算法两种智能算法进行架构，不可避免地造成了整个算法时间相对较长，在迁移应用中需要针对问题做好理论支撑，避免冗余设置造成的计算时间增加。

参考文献

[1] Bellman, R. (1984) Dynamic Programming. Princeton University Press, Princeton.

[2] Powell, W.B. (2011) Approximate Dynamic Programming Solving the Curses of Dimensionality. Wiley, Princeton.

[3] Pardoux, E. and Peng, S. (1992) Backward Stochastic Differential Equations and Quasilinear Parabolic Partial Differential Equations. In: Rozovskii, B.L. Sowers, R.B. and Eds., *Stochastic Partial Differential Equations and Their Applications*, Springer-Verlag, Berlin/Heidelberg, Vol. 176, 200-217. https://doi.org/10.1007/BFb0007334

[4] Pardoux, E. and Tang, S. (1999) Forward-Backward Stochastic Differential Equations and Quasilinear Parabolic PDEs. *Probability Theory and Related Fields*, **114**, 123-150. https://doi.org/10.1007/s004409970001

[5] Yong, J. and Zhou, X.Y. (1999) Stochastic Controls: Hamiltonian Systems and HJB Equations. Springer, New York.

[6] Zhang, J. (2017) Backward Stochastic Differential Equations. Springer, New York, 86.

[7] Han, J.Q. and E, W.N. (2016) Deep Learning Approximation for Stochastic Control Problems.

[8] E, W.N., Han, J.Q. and Jentzen, A. (2017) Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations. *Communications in Mathematics and Statistics*, **5**, 349-380. https://doi.org/10.1007/s40304-017-0117-6

[9] Han, J., Jentzen, A. and E, W.N. (2018) Solving High-Dimensional Partial Differential Equations Using Deep Learning. *Proceedings of the National Academy of Sciences*, **115**, 8505-8510. https://doi.org/10.1073/pnas.1718942115

[10] Berner, J., Grohs, P. and Jentzen, A. (2020) Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black—Scholes Partial Differential Equations. *SIAM Journal on Mathematics of Data Science*, **2**, 631-657. https://doi.org/10.1137/19M125649X

[11] Grohs, P., Jentzen, A. and Salimova, D. (2019) Deep Neural Network Approximations for Monte Carlo Algorithms.

[12] Beck, C., Hornung, F., Hutzenthaler, M., *et al*. (2020) Overcoming the Curse of Dimensionality in the Numerical Approximation of Allen-Cahn Partial Differential Equations via Truncated Full-History Recursive Multilevel Picard Approximations. *Journal of Numerical Mathematics*, **28**, 197-222. https://doi.org/10.1515/jnma-2019-0074

[13] Zang, Y., Bao, G., Ye, X., *et al*. (2020) Weak Adversarial Networks for High-Dimensional Partial Differential Equations. *Journal of Computational Physics*, **411**, Article ID: 109409. https://doi.org/10.1016/j.jcp.2020.109409

[14] Fujii, M., Takahashi, A. and Takahashi, M. (2019) Asymptotic Expansion as Prior Knowledge in Deep Learning Method for High Dimensional BSDEs. *Asia-Pacific Financial Markets*, **26**, 391-408. https://doi.org/10.1007/s10690-019-09271-7

[15] Naito, R. and Yamada, T. (2020) An Acceleration Scheme for Deep Learning-Based BSDE Solver Using Weak Expansions. *International Journal of Financial Engineering*, **7**, Article ID: 2050012. https://doi.org/10.1142/S2424786320500127

[16] Øksendal, B. (2003) Stochastic Differential Equations. Springer, Berlin, 21-74.

[17] Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning. The MIT Press, Cambridge, 151-153, 197-203, 306-310.

[18] Saito, K. 深度学习入门: 基于 Python 的理论与实现[M]. 陆宇杰, 译. 北京: 人民邮电出版社, 2018: 121-162.

[19] Tang, S. (2015) Dynamic Programming for General Linear Quadratic Optimal Stochastic Control with Random Coefficients. *SIAM Journal on Control and Optimization*, **53**, 1082-1106. https://doi.org/10.1137/140979940

[20] Tang, S. (2003) General Linear Quadratic Optimal Stochastic Control Problems with Random Coefficients: Linear Stochastic Hamilton Systems and Backward Stochastic Riccati Equations. *SIAM Journal on Control and Optimization*,

42, 53-75. https://doi.org/10.1137/S0363012901387550

[21] Chang, H. and Rong, X. (2013) An Investment and Consumption Problem with CIR Interest Rate and Stochastic Volatility. *Abstract and Applied Analysis*, **2013**, Article ID: 219397. https://doi.org/10.1155/2013/219397

[22] 雷英杰. MATLAB 遗传算法工具箱及应用[M]. 西安: 西安电子科技大学出版社, 2005.

[23] Price, K.V., Storn, R., Lampinen, J.A., *et al*. (2011) Differential Evolution: A Practical Approach to Global Optimization with 48 Tabeles. Springer, Berlin.

[24] Das, S. and Suganthan, P.N. (2011) Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation*, **15**, 4-31. https://doi.org/10.1109/TEVC.2010.2059031

[25] Mckinney, W. 利用 Python 进行数据分析[M/OL]. 徐敬一, 译. 北京: 机械工业出版社, 2018. https://zh.1lib.tw/book/5422102/5e0803?signAll=1&ts=1826, 2022-01-19.

[26] Idris, I., 张驭宇. Python 数据分析基础教程: NumPy 学习指南[M]. 第 2 版. 北京: 人民邮电出版社, 2014.

[27] 李荣华, 刘播. 微分方程数值解法[M]. 北京: 高等教育出版社, 2009: 107-149.

[28] Brigo, D. and Mercurio, F. (2006) Interest Rate Models—Theory and Practice: With Smile, Inflation and Credit. Springer-Verlag, Berlin.

附录

3.1 节中 5-维 Riccati 方程的参数设计

当 $d=5$， $m=5$ 时，我们有(2.2)式中 $\sigma\left(t,x_t;v_t\right)\mathrm{d}W_t=\sum_{i=1}^{5}\left(C^i x_t+D^i v_t\right)\mathrm{d}W_t^i$ 。

其中

$$C^1=\begin{pmatrix}0.1&0&0&0&0\\0&0&0&0&0\\0&0&0.1&0&0\\0&0&0&0.1&0\\0&0&0&0&0.02\end{pmatrix},\quad C^2=\begin{pmatrix}0.02&0&0&0&0\\0&0.3&0&0&0\\0&0&0.01&0&0\\0&0&0&0.1&0\\0&0&0&0&0\end{pmatrix},\quad C^3=\begin{pmatrix}0.1&0&0&0&0\\0&0&0&0&0\\0&0&0.1&0&0\\0&0&0&0.1&0\\0&0&0&0&0.03\end{pmatrix},$$

$$C^4=\begin{pmatrix}0.02&0&0&0&0\\0&0.03&0&0&0\\0&0&0&0&0\\0&0&0&0.1&0\\0&0&0&0&0\end{pmatrix},\quad C^5=\begin{pmatrix}0.02&0&0&0&0\\0&0.1&0&0&0\\0&0&0.02&0&0\\0&0&0&0.11&0\\0&0&0&0&0.02\end{pmatrix}.$$

$$D^1=\begin{pmatrix}0&0&0&0&0\\0&0.01&0&0&0\\0&0&0.02&0&0\\0&0&0&0&0\\0&0&0&0&0.01\end{pmatrix},\quad D^2=\begin{pmatrix}0.03&0&0&0&0\\0&0&0&0&0\\0&0&0.1&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix},\quad D^3=\begin{pmatrix}0.01&0&0&0&0\\0&0.01&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix},$$

$$D^4=\begin{pmatrix}0&0&0&0&0\\0&0.02&0&0&0\\0&0&0&0&0\\0&0&0&0.02&0\\0&0&0&0&0\end{pmatrix},\quad D^5=\begin{pmatrix}0.03&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0.02&0\\0&0&0&0&0\end{pmatrix}.$$