文章编号：2096-1618(2024)04-0499-13

常微分方程的数值求解与方法

黑亚芳， 胡建成
(成都信息工程大学应用数学学院，四川 成都 610225)

摘要：针对各种微分方程数值求解的方法，如有限差分法、有限元法等方法，在求解微分方程时存在计算存储量、计算时间等都随着微分方程维数的增加而剧烈增长的问题，严重制约了高维问题的求解。神经网络因其能够无限逼近任意非线性函数的特性，为求解微分方程提供了一种新的思路。通过神经网络训练，得到微分方程的近似解是连续函数，且具有足够的精度，因此可以得到解的任意阶导数。该方法的优势在于当问题维数增大时，计算量和存储量增加相对较小，可以克服维数灾难求解高维问题；同时，具有良好的泛化性和求解复杂区域问题的能力。提出一种求解微分方程的神经网络方法，即通过物理约束耦合神经网络的方法。通过数值算例说明，神经网络方法在求解微分方程问题上有高效率、很好的泛化等优点，能够保证优化算法的收敛性，且近似解具有足够的精度，为微分方程求解提供了一种有效的途径。

关 键 词：常微分方程；数值计算；神经网络
中图分类号：O241.81 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2024.04.017

0 引言

常微分方程是数学中的一个重要分支，是描述物理、化学、生物等领域中许多自然现象和现象变化规律的重要数学工具，已经存在了几个世纪。许多科学和工业应用等领域都需要求解常微分方程来描述人们所感兴趣的一些科学自然现象，在物理和工程领域中很多问题最终都可以转化为求解微分方程的解。到目前为止，求解常微分方程已经发展出许多方法，如有限差分法(FDM)和有限元法(FEM)[1-5]。

近年来，随着计算机的快速发展，科学计算能力和计算机对数据的存储能力都有了惊人的提高，这就掀起了对机器学习研究的热潮，神经网络(neural networks，NNs)已被非常有效地应用于各种应用[6]，如计算机视觉和自然语言处理。在这个思想的指引下科学家开始将神经网络与微分方程模型相结合，以更好地理解神经网络的运作机理和应用神经网络完成各种任务的能力，使用微分方程描述神经网络中神经元之间的连接和交互方式。这种方法可以用于模拟和预测神经网络的动态行为，并且可以将自动微分[7]作为一种无网格方法，从而打破维数的诅咒[8-9]。Raissi 等[10]提出了基于物理的神经网络(physics-informed neural networks，PINNs)，是一个典型的框架，用于将数据与从监管机构获得的法律相结合。该框架通过将控制方程的残余形式与数据驱动部分(如初始条件和边界条件)相结合来实现。因此，所提出的神经网络可以同时满足物理约束部分和数据驱动部分。框架将网络结构与决定性的物理定律联系起来训练过程，大大提高了神经网络的建模能力。

本文构造了一种物理约束耦合神经网络的算法框架(PCNNs)，依赖于神经网络的函数逼近能力。该形式采用神经网络作为基本近似元素，对其参数(权重和偏差)进行调整，以使适当的误差函数最小化。为了训练网络，采用了优化技术，这反过来又需要计算相对于网络参数的误差梯度。在提出的算法中，模型函数表示为两个项的和：第一项满足初始条件或边界条件，不包含可调参数；第二项包括神经网络，以满足微分方程的解。由于已知具有隐藏层的多层感知器可以将任何函数近似到任意精度，因此将神经网络作为处理微分方程的模型是合理的。基于神经网络微分方程求解模型具有许多优点：首先，神经网络的解是可微的；其次，基于神经网络的微分方程求解方法提供了一个具有很好的泛化性质的解；再次，所需要的模型参数数量远远少于任何其他求解技术，因此得到了紧凑的求解模型，对内存空间的要求非常低；最后，该方法的主要优点是在保持令人满意的精度的同时，大大降低了权重更新所涉及的计算复杂性。

1 神经网络

神经网络必须配置成一组输入的应用程序产生所需的一组输出。存在各种设置连接强度的方法：一种

收稿日期：2023-02-09
基金项目：四川省科技计划重点研发资助项目(2019YFS0143)

方法是使用先验知识显式地设置权重;另一种方法是通过给神经网络喂食、教授模式并让其根据某种学习规则改变权重来训练神经网络。神经网络领域广泛使用“学习”一词来描述这一过程,描述为:根据示例的统计数据确定一组优化的权重。

1.1 神经网络结构

神经网络是受生物学启发的机器学习(ML)算法,能够从观测数据中学习。神经网络是处理元素、单元或节点的相互连接,其功能类似于人类神经元。与其他机器学习技术类似,神经网络模型是通过一组可训练的参数来描述的,简单地称为权重。这些参数是在一个过程中推断出来的,该过程的目标是找到一个局部最优选择,从而导致数据集(通常称为训练数据集)上网络的“准确”输入输出映射,其中预期结果是已知的。

神经网络方法可以求解常微分方程,依赖于神经网络的函数近似性质。该形式采用神经网络作为基本近似元素。神经网络的训练可以通过任何优化技术来完成,这反过来又需要通过基函数逼近来计算关于网络参数的误差梯度。神经网络可以从相应神经元系统的输入到输出进行非线性映射,适用于分析无解析解或不易计算的初值/边值问题所定义的问题。多层神经网络的应用之一是实值多变量函数在封闭解析形式下的全局逼近,即这种神经网络是通用逼近器。

这里选取简单的前馈神经网络为例做简单的介绍。前馈神经网络(feedforward neural network,FNN)是一种单向多层的网络模型,每层都是由若干个神经元组成,不同的神经元属于不同的层,每一层的神经元可以接受前一层的神经元信号,并产生信号输出到下一层。第 0 层叫做输入层,最后一层叫做输出层,中间的叫做隐藏层,整个网络中无反馈,信号从输入层到输出层单向传播。前馈神经网络是深度神经网络中的一种,具体结构如图 1 所示。每个神经元节点是激活函数,代表神经网络处理非线性问题的能力。两两相连的神经元节点之间有一个权值,代表神经网络的认知记忆能力。

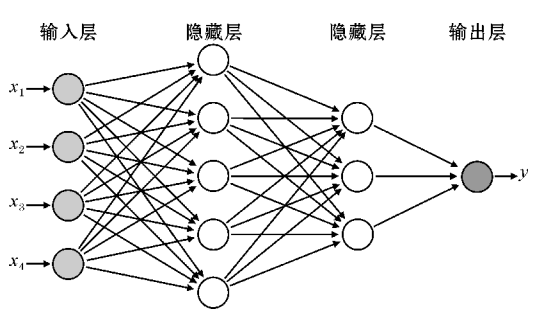

图 1 FNN 模型结构

前馈神经网络的信息传播:

$$\begin{cases} z^{(l)} = W^{(l)} \cdot a^{(l-1)} + b^{(l)} \\ a^{(l)} = f_l(z^{(l)}) \end{cases}$$

式中,l 表示神经网络的层数,m^l 表示第 l 层神经元的个数,$f_l(\cdot)$ 表示 l 层神经元的激活函数,$z^{(l)} \in R^{m^l}$ 表示 l 层神经元的净输入,$W^{(l)} \in R^{m^l \times m^{l-1}}$ 表示 $l-1$ 到第 l 层的权重矩阵,$a^{(l)} \in R^{m^l}$ 表示 l 层神经元的输出,$b^{(l)} \in R^{m^l}$ 表示 $l-1$ 到第 l 层的偏置。这样前馈神经网络通过逐层的信息传递,得到网络最后的输出 $a^{(l)}$。于是整个网络可以看作是一个符合函数,将向量 X 作为第 1 层的输入 $a^{(0)}$,将第 L 层的输出 $a^{(L)}$ 作为整个函数的输出。

$$X = a^{(0)} \to z^{(1)} \to a^{(1)} \to z^{(2)} \to \cdots \to a^{(L-1)} \to z^{(L)} \to a^{(L)} = \varphi(x; W, b)$$

1.2 通用逼近定理

1989 年 Cybenko[11] 和 Hornik 等[12] 证明了多层感知器的通用逼近定理。令 I_n 表示 n 维包含所有可能输入样本 $x = (x_1, x_2, \cdots, x_n)$,其中 $x_i \in [0,1]$,$i = 1,2,\cdots,n$。让 $C(I_n)$ 是 I_n 上连续函数的空间,给定一个连续的 sigmoid 函数 $\varphi(\cdot)$,通用近似定理说明了有限和的形式:

$$y_k = y_k(x, w) = \sum_{i=1}^{N_2} w_{ki}^3 \varphi\left(\sum_{j=0}^{n} w_{kj}^3 x_j\right), k = 1,2,\cdots,m$$

在 $C(I_n)$ 上密集。这仅仅意味着给定任何函数 $y \in C(I_n)$ 和 $\varepsilon > 0$,有一个上述形式的和 $y(x,w)$ 是满足 $|y(x,w) - y(x)| < \varepsilon$, $\forall x \in I_n$。

1.3 关于网络输入的梯度计算

方程的有效最小化可以被认为是一个训练神经网络的过程,其中对应于每个输入向量的误差是必须变为零的方程的值。这个误差值的计算不仅涉及网络输出(就像传统训练中的情况一样),还涉及输出对网络权重的导数,不仅需要计算网络的梯度,还需要计算网络导数对其输入的梯度。

计算关于输入向量的梯度,考虑一个具有输入单元的多层感知器神经网络,一个具有 sigmoid 单元的隐藏层和一个线性输出单元。

对于给定的 $x = (x_1, x_2, \cdots, x_n)$ 输入向量,网络的输出:

$$N(x,p) = \sum_{i=1}^{m} v_j \varphi(z_j), \quad z_j = \sum_{i=1}^{n} w_{ji} x_i + u_j$$

式中,w_{ji} 表示输入单元 i 到隐藏层单元 j 的权重,v_j 表示从隐藏单元 j 到输出单元的权重,u_j 表示偏差,$\varphi(z_j)$ 是 sigmoid 激活函数。可以直观地表明:

$$\frac{\partial^k N}{\partial x_j^k} = \sum_{i=1}^{H} v_i w_{ij}^k \varphi_i^{(k)} \tag{1}$$

式中,$\varphi_i=\varphi(z_i)$ 和 $\varphi^{(k)}$ 表示 sigmoid 的 k^{th} 阶导数。此外,很容易证实:

$$\frac{\partial^{\lambda_1}}{\partial x_1^{\lambda_1}}\frac{\partial^{\lambda_2}}{\partial x_2^{\lambda_2}}\cdots\frac{\partial^{\lambda_n}}{\partial x_n^{\lambda_n}}N=\sum_{i=1}^{n}v_iP_i\varphi_i^{(\Lambda)} \tag{2}$$

式中,$P_i=\prod_{k=1}^{n}w_{ik}^{\lambda_k}$ 和 $\Lambda=\sum_{i=1}^{n}\lambda_i$。

式(2)表明,网络对其任何输入的导数等价于具有一个隐藏层的前馈神经网络 $N_g(x)$,具有相同的权重 w_{ij} 和阈值 u_i,并且每个权重 v_i 都被替换为 v_iP_i。此外,将每个隐藏单元的传递函数替换为 sigmoid 函数的 Λ^{th} 阶导数。

$N_g(x)$ 的梯度相对于原始网络参数的梯度可以很容易地得到:

$$\begin{cases}\dfrac{\partial N_g}{\partial v_i}=P_i\sigma_i^{(\Lambda)}\\[2ex]\dfrac{\partial N_g}{\partial u_i}=v_iP_i\sigma_i^{(\Lambda+1)}\\[2ex]\dfrac{\partial N_g}{\partial w_i}=x_iv_iP_i\sigma_i^{(\Lambda+1)}+v_i\lambda_iw_{ij}^{\lambda-1}\left(\prod_{k=1,k\neq j}w_{ik}^{\lambda_k}\right)\sigma_i^{(\Lambda)}\end{cases} \tag{3}$$

一旦定义了误差对网络参数的导数,就可以直接采用几乎任何最小化技术。本文采用 Adam 最小化技术,它是二次收敛的并且有很好的性能。还必须指出,对于给定的网格点,在有并行软件的情况下,每个网络对参数的导数可以同时得到。此外,在反向传播的情况下,可以采用在线或批处理方式进行权重更新。

2　方法描述

2.1　微分方程的一般公式

考虑下列既表示常微分方程又表示偏微分方程的一般微分方程:

$$G(x,\psi(x),\nabla\psi(x),\nabla^2\psi(x),\cdots)=0,x\in D \tag{4}$$

在某些初始条件(I. Cs)或边界条件(B. Cs)的限制下,这里 $x=(x_1,x_2,\cdots,x_n)\in R^n$,$D\subset R^n$ 表示定义域。$\psi(x)$ 是要计算的未知标量值解。其中 G 是定义微分方程结构的函数,∇ 是微分算子。所提出的方法也可以应用于高阶微分方程。对于式(4),首先将计算域离散化,域 D 和它的边界 S 分别离散成 $\hat{D}$ 和 $\hat{S}$,然后将问题转化为以下方程组:

$$G(x_i,\psi(x_i),\nabla\psi(x_i),\nabla^2\psi(x_i))=0,\forall x_i\in\hat{D} \tag{5}$$

受到初始条件(I. Cs)或边界条件(B. Cs)的限制。

如果 $\psi_t(x,p)$ 表示参数 p 可调的试验解,则问题转化为

$$\min_pG(x_i,\psi_t(x_i,p),\nabla\psi_t(x_i,p),\nabla^2\psi_t(x_i,p))^2 \tag{6}$$

受给定条件所施加的约束。因为当式(6)的最小值越接近于 0 时,就越接近式(5)的解。

2.2　方法形式

在提出的物理约束耦合神经网络的方法中,试验解 $\psi_t(x,p)$ 采用神经网络和神经结构的参数(权重、偏差)。为试解 $\psi_t(x,p)$ 选择一种形式,使构造满足I. Cs或B. Cs。$\psi_t(x,p)$ 可以写成两项的和:

$$\psi_t(x,p)=A(x)+F(x,N(x,p))$$

式中,$A(x)$ 不包含可调参数且满足初始条件或边界条件,F 的构造是为了不影响初始条件或边界条件,并且 $N(x,p)$ 是一个单输出神经网络,其参数 p 和输入 n 单元由输入数据 x 提供。F 实际上是神经网络的操作模型,它的权重和偏差需要调整以处理最小化问题。因此,问题已经从原来的约束优化问题减少到无约束问题。由于试解的形式更容易解决,因此神经网络将微分方程问题转化为函数逼近问题。

神经网络 $N(x,p)$ 如下:

$$N(x,p)=\sum_{j=1}^{m}v_j\sigma(z_j),z_j=\sum_{i=1}^{n}w_{ji}x_i+u_j$$

式中,w_{ji} 表示输入单元 i 到隐藏层单元 j 的权重,v_j 表示从隐藏单元 j 到输出单元的权重,u_j 表示偏差,$\sigma(z_j)$ 是 sigmoid 激活函数。

神经网络权重决定了预测结果与期望结果的接近程度。如果神经网络权重不能做出正确的预测,那么只需要调整偏差。每个隐藏单元的 sigmoid 激活为 $\sigma(x)=\dfrac{1}{1-e^{-x}}$。

3　方法说明

3.1　一阶常微分方程

考虑一阶常微分方程:

$$\frac{d\psi(x)}{dx}=f(x,\psi) \tag{7}$$

这里 $x\in[0,1]$ 且 $\psi(0)=A$。

测试解写成如下形式:

$$\psi_t(x)=A+xN(x,p)$$

这里 $N(x,p)$ 是一个输入 x 和权重 p 的神经网络的输出。$\psi_t(x)$ 的形式满足约束条件最小化的误差为:

$$E[p]=\sum_i\left\{\frac{d\psi_t(x_i)}{dx}-f(x_i,\psi_t(x_i))\right\}^2$$

式中 x_i 是[0,1]中的点,由于 $d\psi_t(x)/dx=N(x,p)+xN(x,p)dx$,利用式(1) ~ (3)计算关于参数 p 的误差梯度很简单。这同样适用于所有后续的模型问题。

3.2　二阶常微分方程

考虑二阶常微分方程:

$$\frac{d\psi^2(x)}{dx^2}=f(x,\psi,\frac{d\psi(x)}{dx})$$

对初值问题 $\psi(0)=A$ 和 $\frac{\mathrm{d}}{\mathrm{d}x}\psi(0)=A'$，测试解写成如下形式：

$$\psi_t(x)=A+A'x+x^2N(x,p)$$

对两点的迪利克雷边值条件：$\psi(0)=A$ 和 $\psi(1)=B$，测试解写成如下形式：

$$\psi_t(x)=A(1-x)+Bx+x(1-x)N(x,p)$$

在上述两种二阶常微分方程的情况下，最小化的误差函数如式(6)所示。

3.3 *K* 阶常微分方程

考虑 K 阶常微分方程系统：

$$\frac{\mathrm{d}\psi_i}{\mathrm{d}x}=f_i(x,\psi_1,\psi_2,\cdots,\psi_K)$$

这里 $\psi_t(0)=A_i,(i=1,\cdots,K)$，现在，考虑每个测试解 $\psi_{t_i}(i=1,\cdots,K)$ 的一个神经网络，其形式：

$$\psi_{t_i}(x)=A_i+xN_i(x,p_i)$$

最小化的误差：

$$E[p]=\sum_{k=1}^{K}\sum_{i}\left\{\frac{\mathrm{d}\psi_{t_k}(x_i)}{\mathrm{d}x}-f_k(x_i,\psi_{t1},\psi_{t2},\cdots,\psi_{tK})\right\}^2$$

4 数值实验与分析

使用物理约束耦合神经网络模型，并使用不同的神经网络结构来验证该方法的有效性，实现与 PINN 对比。每个隐藏单元的 sigmoid 激活函数为 $\sigma(x)=\frac{1}{1+\mathrm{e}^{-x}}$。对于每一个测试问题，精确的解析解 $\psi_a(x)$ 是预先知道的，误差为 MSE 误差。

算例 1：

$$\frac{\mathrm{d}}{\mathrm{d}x}\psi+\left(x+\frac{1+3x^2}{1+x+x^3}\right)\psi=x^3+2x+x^2\,\frac{1+3x^2}{1+x+x^3} \tag{8}$$

式中 $\psi(0)=1$ 和 $x\in[0,1]$。解析解为 $\psi_a(x)=\frac{\mathrm{e}^{-x^2/2}}{1+x+x^3}+x^2$。根据式(7)，这里 $A=1$，因此测试解形式为 $\psi_t(x)=1+xN(x,p)$

使用神经网络对方程(8)进行求解，网络训练使用网格[0,1]中 10 个等距点。接下来采用不同的方法和神经网络对其进行求解。使用 PINNs 对算例 1 进行求解，结果如图 2 所示。使用物理约束耦合简单的三层前馈神经网络(PCANN)求解，其结果由图 3 给出。接下来采用物理约束耦合全连接神经网络(PCFNN)以及物理约束耦合并联全连接神经网络(PCPFNN)求解，其结果由图 4 ~ 5 给出，其误差和损失结果由表 1 ~ 4 给出。

由表 1 ~ 4 可以看出，PINNs 在求解微分方程的时候效果偏差。在迭代 9000 次的时候，使用 PINNs 得到的 MSE 误差为0.023618，使用 PCANN 得到的 MSE 误差为0.000004，使用 PCFNN 和 PCPFNN 得到的 MSE 误差为0.000073。由此可见构造的物理约束耦合神经网络的方法能够更有效地求解微分方程。

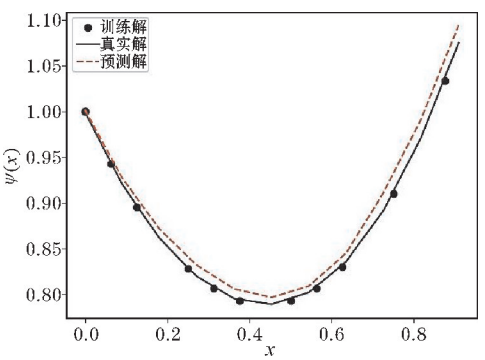

图 2 PINNs 求解算例 1

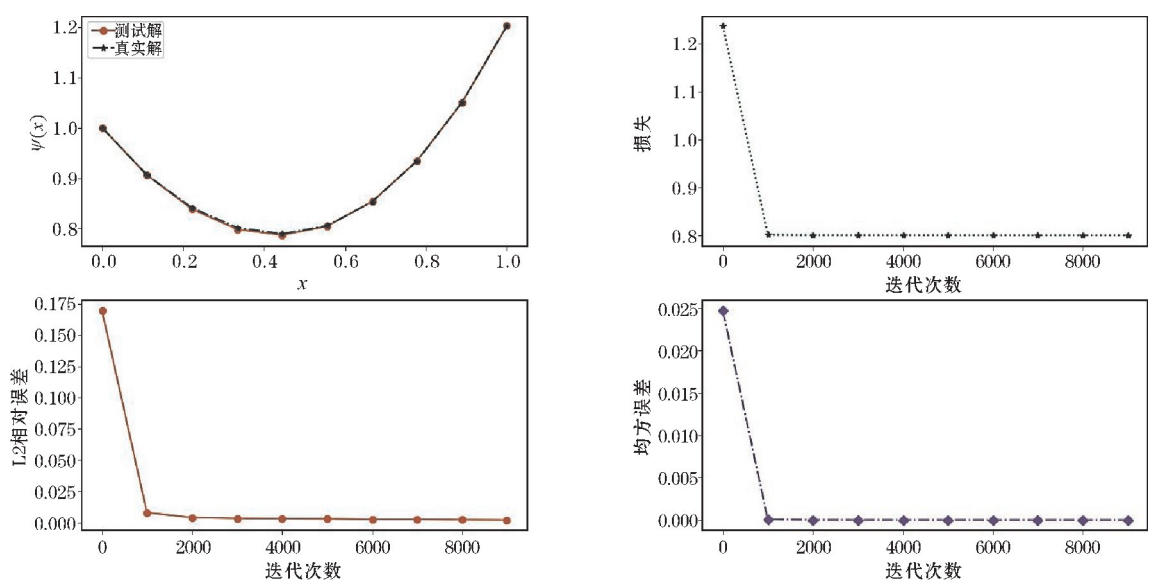

图 3 PCANN 求解算例 1

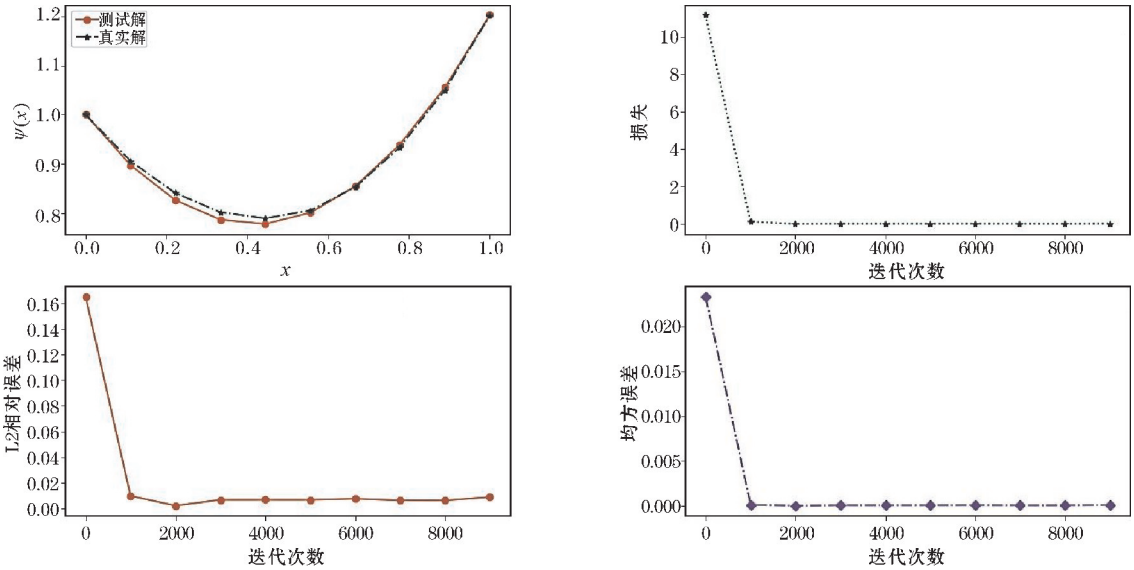

图 4　PCFNN 求解算例 1

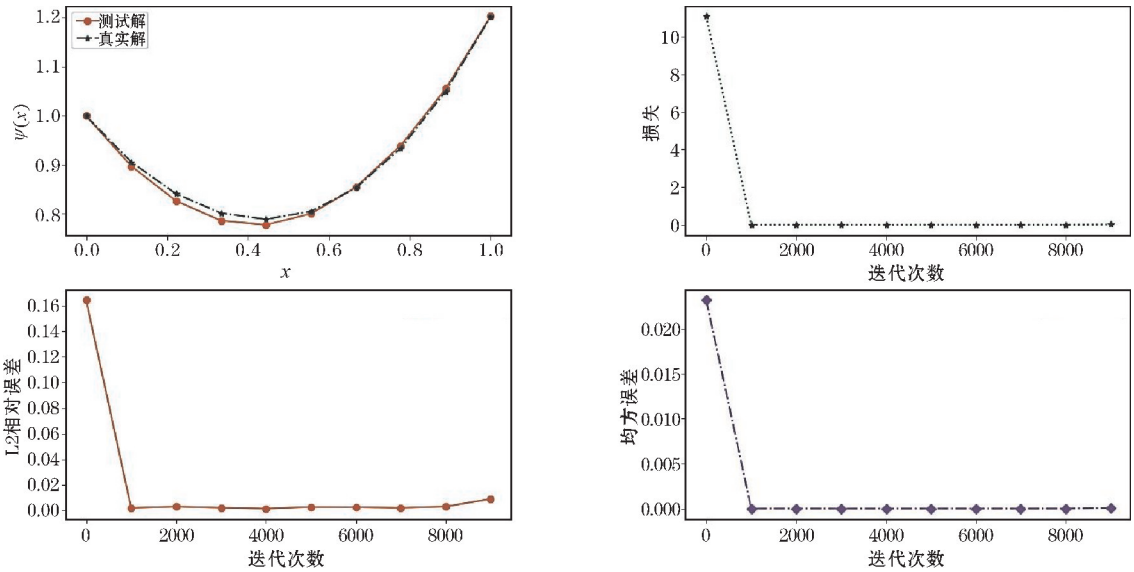

图 5　PCPFNN 求解算例 1

表 1　算例 1 用 PINNs 求解误差和损失

迭代次数	MSE 误差	损失
0	1.000000	2.375843
1000	0.688259	0.002560
2000	0.461463	0.003514
3000	0.298997	0.000959
4000	0.186306	0.000841
5000	0.111880	0.001085
6000	0.065893	0.000434
7000	0.039914	0.001512
8000	0.027264	0.000748
9000	0.023618	0.001060

表 2　算例 1 用 PCANN 求解误差和损失

迭代次数	MSE 误差	损失
0	0.024674	8.760549
1000	0.000055	0.020432
2000	0.000015	0.007304
3000	0.000010	0.006664
4000	0.000009	0.005907
5000	0.000008	0.005059
6000	0.000006	0.004152
7000	0.000005	0.003360
8000	0.000005	0.002852
9000	0.000004	0.002480

表 3 算例 1 用 PCFNN 求解误差和损失

迭代次数	MSE 误差	损失
0	0.023614	8.116918
1000	0.000002	0.001826
2000	0.000023	0.006240
3000	0.000007	0.001857
4000	0.000070	0.019456
5000	0.000058	0.019671
6000	0.000073	0.019844
7000	0.000073	0.019844
8000	0.000073	0.019844
9000	0.000073	0.019844

表 4 算例 1 用 PCPFNN 求解误差和损失

迭代次数	MSE 误差	损失
0	0.023225	11.078766
1000	0.000004	0.001688
2000	0.000009	0.004424
3000	0.000005	0.002028
4000	0.000002	0.001605
5000	0.000007	0.003942
6000	0.000006	0.003718
7000	0.000004	0.002321
8000	0.000009	0.004946
9000	0.000073	0.019844

算例 2：

$$\frac{\mathrm{d}}{\mathrm{d}x}\Psi+\frac{1}{5}\Psi=\mathrm{e}^{-\frac{x}{5}}\cos x \tag{9}$$

式中，$\Psi(0)=0$ 和 $x\in[0,2]$。解析解为 $\Psi_a(x)=\mathrm{e}^{-\frac{x}{5}}\cdot\sin x$。根据式(7)测试解形式为 $\Psi_t(x)=xN(x,p)$。

使用神经网络对方程(9)进行求解，网络训练使用网格[0,2]中 20 个等距点，接下来采用不同的方法和神经网络对其进行求解。使用 PINNs 对算例 2 进行求解，结果如图 6 所示。使用物理约束耦合简单的三层前馈神经网络(PCANN)求解，其结果由图 7 给出。接下来采用物理约束耦合全连接神经网络(PCFNN)以及物理约束耦合并联全连接神经网络(PCPFNN)求解，其结果由图 8 ~9 给出。误差和损失结果由表 5 ~8 给出，这里采用 MSE 误差。

由表 5 ~8 可以看出，PINNs 在求解微分方程的时候效果偏差。在迭代 9000 次的时候，使用 PINNs 得到的 MSE 误差为0.015747，使用 PCANN 得到的 MSE 误差为 0.000073，使用 PCFNN 得到的 MSE 误差为 0.000010，使用 PCPFNN 得到的 MSE 误差为 0.000006，由此可见构造的物理约束耦合神经网络的方法能够更有效地求解微分方程。

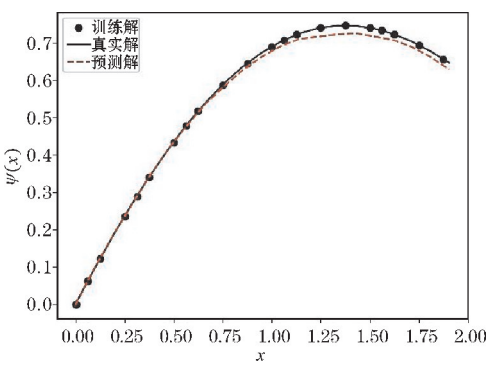

图 6 PINNs 求解算例 2

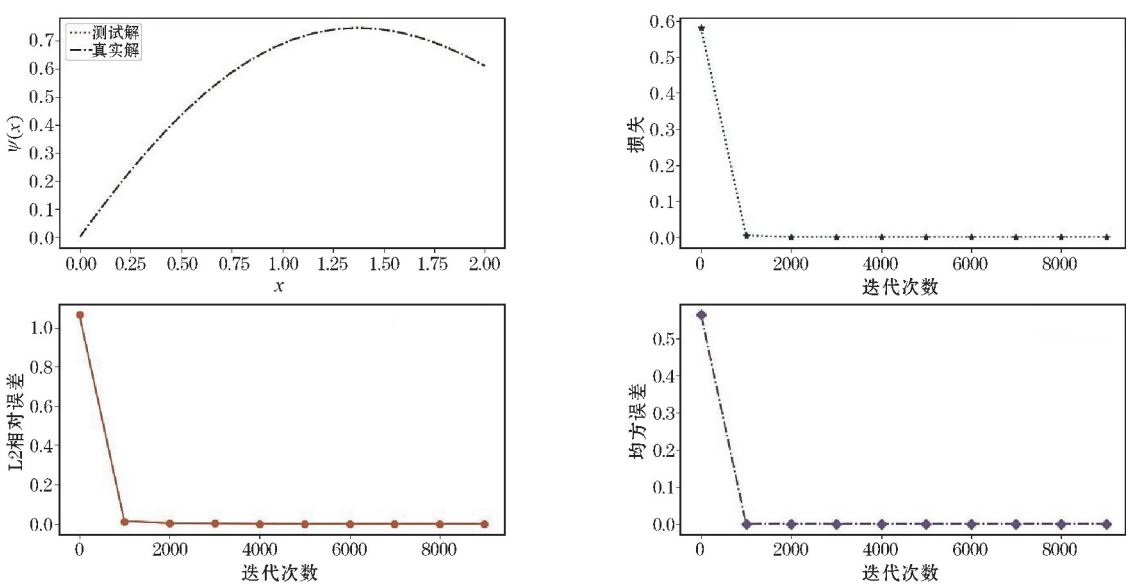

图 7 PCANN 求解算例 2

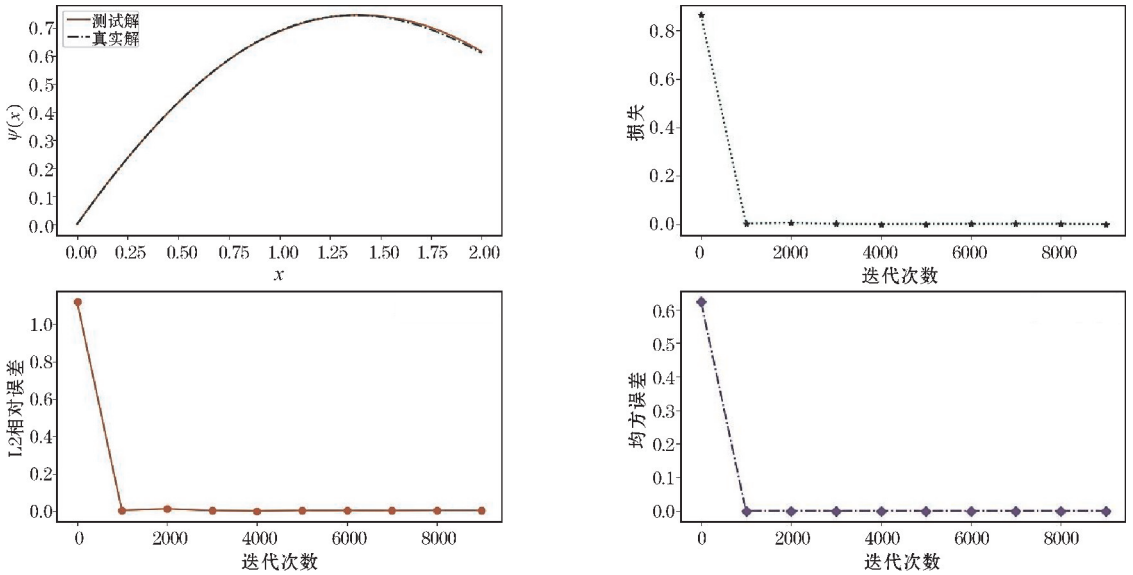

图 8　PCFNN 求解算例 2

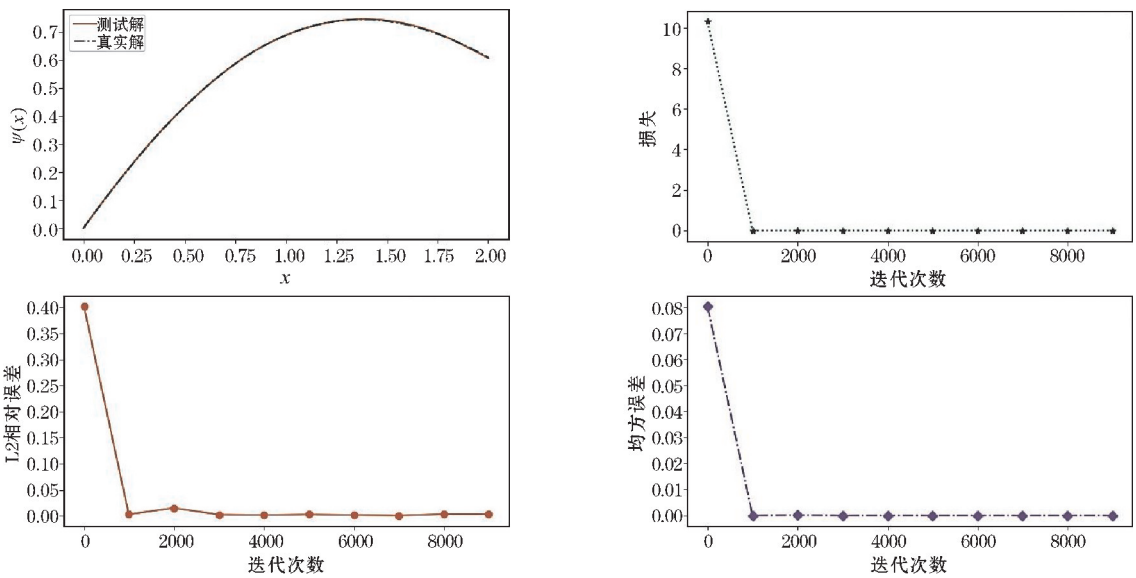

图 9　PCPFNN 求解算例 2

表 5　算例 2 用 PINNs 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.015357	0.000000	0.424932
1000	0.020350	0.000004	0.003593
2000	0.043189	0.000068	0.001395
3000	0.068338	0.000381	0.001156
4000	0.095611	0.001327	0.000793
5000	0.124816	0.003533	0.001058
6000	0.155755	0.007921	0.001121
7000	0.188229	0.015747	0.001610
8000	0.222036	0.028619	0.001645
9000	0.256974	0.048516	0.002195

表 6　算例 2 用 PCANN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.164478	0.023225	11.078766
1000	0.002202	0.000004	0.001688
2000	0.003177	0.000009	0.004424
3000	0.002334	0.000005	0.002028
4000	0.001671	0.000002	0.001605
5000	0.002886	0.000007	0.003942
6000	0.002719	0.000006	0.003718
7000	0.002249	0.000004	0.002321
8000	0.003253	0.000009	0.004946
9000	0.009206	0.000073	0.019844

表 7 算例 2 用 PCFNN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	1.119954	0.623303	0.863134
1000	0.005046	0.000013	0.004489
2000	0.012686	0.000080	0.007145
3000	0.003531	0.000006	0.002358
4000	0.001948	0.000002	0.001303
5000	0.003536	0.000006	0.002285
6000	0.003624	0.000007	0.002372
7000	0.003500	0.000006	0.002365
8000	0.004414	0.000010	0.002485
9000	0.004382	0.000010	0.000955

表 8 算例 2 用 PCPFNN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.401996	0.080305	10.303808
1000	0.003218	0.000005	0.001284
2000	0.015295	0.000116	0.001079
3000	0.002703	0.000004	0.003326
4000	0.001794	0.000002	0.001072
5000	0.002925	0.000004	0.001205
6000	0.001856	0.000002	0.000858
7000	0.001015	0.000001	0.000473
8000	0.003401	0.000006	0.001488
9000	0.003375	0.000006	0.001885

算例 3：

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}\Psi+\frac{1}{5}\frac{\mathrm{d}}{\mathrm{d}x}\Psi+\Psi=-\frac{1}{5}\mathrm{e}^{-\frac{x}{5}}\cos x \qquad (10)$$

考虑初值问题 $\Psi(0)=0,\frac{\mathrm{d}}{\mathrm{d}x}\Psi(0)=1,x\in[0,1]$。解析解为 $\Psi_a(x)=\mathrm{e}^{-\frac{x}{5}}\sin x$，测试解的形式为 $\Psi_t(x)=x+x^2N(x,p)$。

使用神经网络对方程(10)进行求解，网络训练使用网格[0,1]中 10 个等距点。接下来采用不同的方法和神经网络对其进行求解。使用 PINNs 对算例 3 进行求解，结果如图 10 所示。使用物理约束耦合简单的三层前馈神经网络(PCANN)求解，其结果由图 11 给出。接下来采用物理约束耦合全连接神经网络(PCFNN)以及物理约束耦合并联全连接神经网络(PCPFNN)求解，其结果由图 12～13 给出。误差和损失结果如表 9～12 所示，这里采用 L2 误差和 MSE 误差。

由表 9～12 可以看出，PINNs 在求解微分方程的时候效果偏差。在迭代 9000 次的时候，使用 PINNs 得到的 MSE 误差为0.023237，使用 PCANN 得到的 MSE 误差为0.000140，使用 PCFNN 得到的 MSE 误差为 0.000075，使用 PCPFNN 得到的 MSE 误差为 0.000176，由此可见构造的物理约束耦合神经网络的方法能够更有效地求解微分方程。

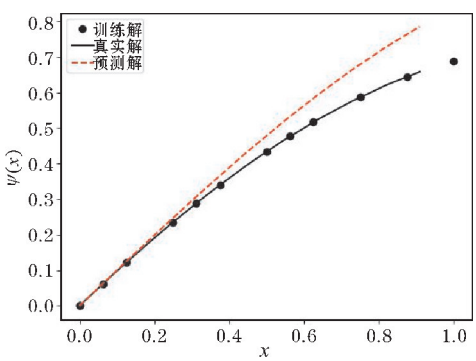

图 10 PINNs 求解算例 3

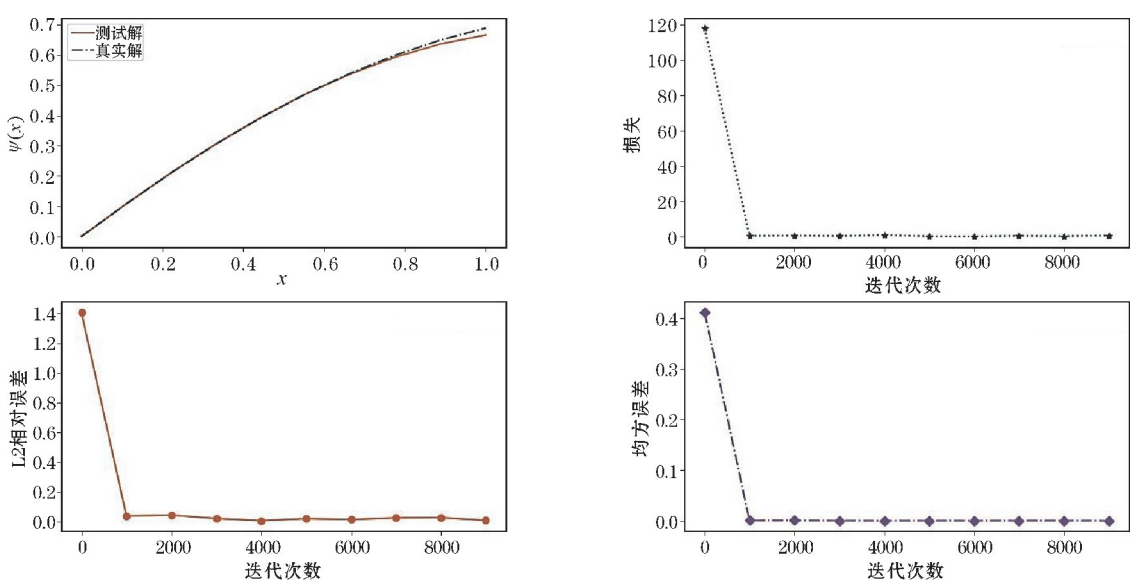

图 11 PCANN 求解算例 3

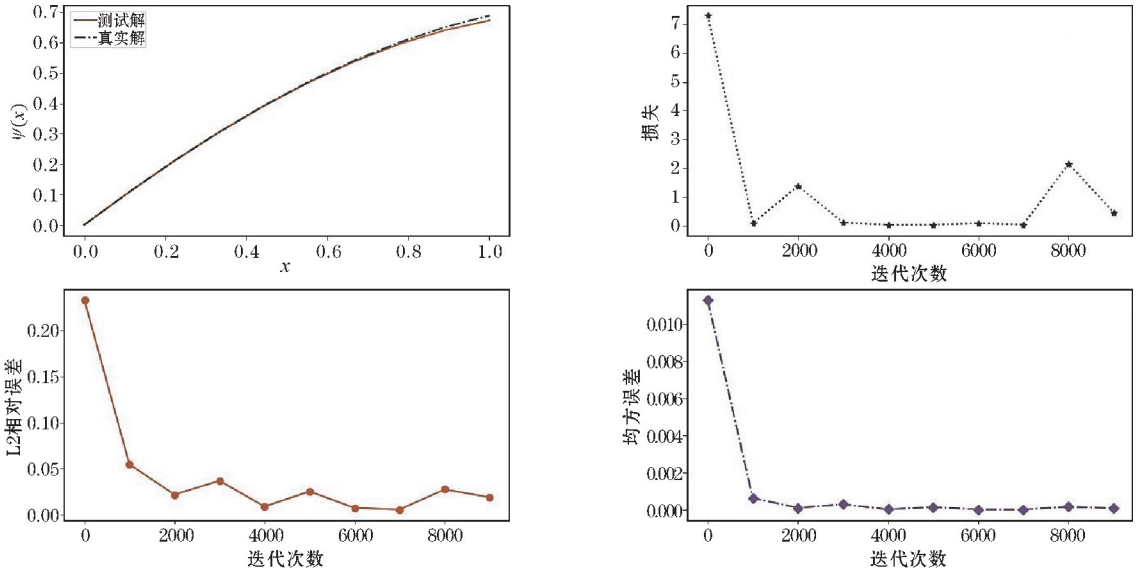

图 12　PCFNN 求解算例 3

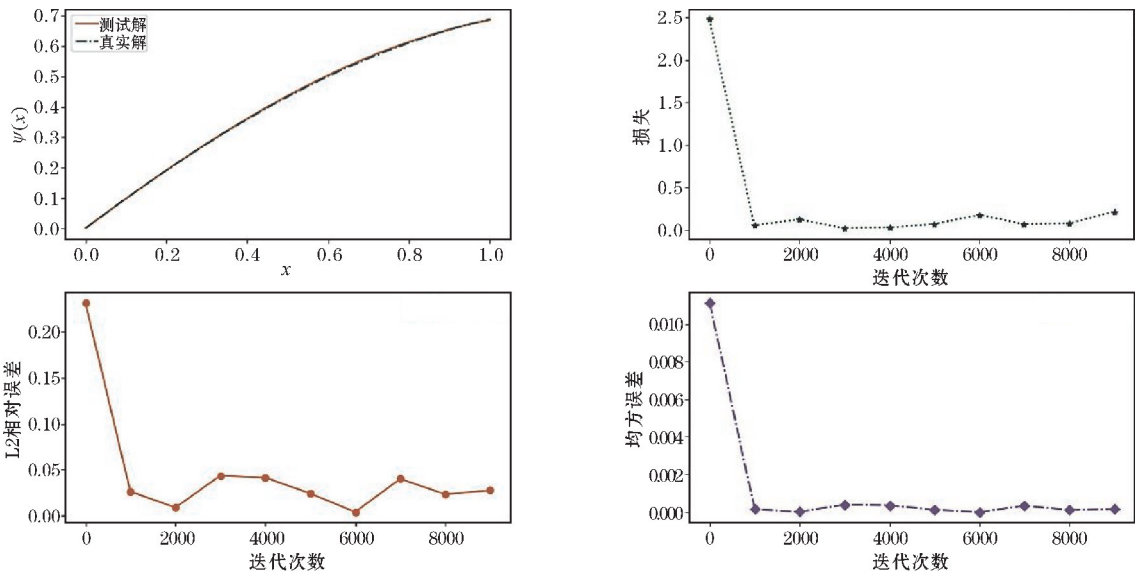

图 13　PCPFNN 求解算例 3

表 9　算例 3 用 PINNs 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.006594	0.000000	0.018510
1000	0.009565	0.000000	0.000026
2000	0.019370	0.000003	0.000006
3000	0.041014	0.000056	0.000004
4000	0.064780	0.000312	0.000006
5000	0.090503	0.001083	0.000024
6000	0.118019	0.002878	0.000001
7000	0.147156	0.006443	0.000026
8000	0.177742	0.012794	0.000020
9000	0.209601	0.023237	0.000001

表 10　算例 3 用 PCANN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.457030	0.043405	27.015709
1000	0.027783	0.000160	0.661004
2000	0.018600	0.000072	0.026958
3000	0.026587	0.000147	0.001473
4000	0.005574	0.000006	0.094560
5000	0.042025	0.000367	0.085438
6000	0.036018	0.000270	0.657996
7000	0.026075	0.000141	0.237428
8000	0.040791	0.000346	0.009819
9000	0.025962	0.000140	0.002632

表 11 算例 3 用 PCFNN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.233206	0.011301	7.306309
1000	0.054626	0.000620	0.079114
2000	0.021834	0.000099	1.372849
3000	0.036774	0.000281	0.104971
4000	0.008913	0.000017	0.017461
5000	0.025278	0.000133	0.021248
6000	0.007362	0.000011	0.082388
7000	0.005681	0.000007	0.024941
8000	0.027365	0.000156	2.146689
9000	0.018950	0.000075	0.456067

表 12 算例 3 用 PCPFNN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.305969	0.019454	8.697351
1000	0.036002	0.000269	0.352201
2000	0.050293	0.000526	0.106360
3000	0.016982	0.000060	0.234487
4000	0.040780	0.000346	0.036627
5000	0.004643	0.000004	0.195004
6000	0.040463	0.000340	0.460185
7000	0.021739	0.000098	0.195434
8000	0.004862	0.000005	1.618045
9000	0.029098	0.000176	0.115681

算例 4：

考虑边界问题 $\Psi(0)=0,\Psi(1)=\sin(1)\mathrm{e}^{-1/5},x\in[0,1]$，解析解为 $\Psi_a(x)=\mathrm{e}^{-\frac{x}{5}}\sin x$，并且测试解的形式为 $\Psi_t(x)=\sin(1)\mathrm{e}^{-1/5}x+x(1-x)N(x,p)$。

使用神经网络对方程(10)进行求解，网络训练使用网格[0,1]中 10 个等距点。接下来采用不同的方法和神经网络对其进行求解。使用 PINNs 对算例 4 进行求解，结果如图 14 所示。使用物理约束耦合简单的三层前馈神经网络(PCANN)求解，其结果如图 15 所示。接下来采用物理约束耦合全连接神经网络(PCFNN)以及物理约束耦合并联全连接神经网络(PCPFNN)求解，其结果如图 16～17 所示。其误差和损失结果如表 13～16 所示。这里采用 L2 误差和 MSE 误差。

由表 13～16 可以看出，PINNs 在求解微分方程的时候效果偏差。在迭代 9000 次的时候，使用 PINNs 得到的 MSE 误差为0.506955，使用 PCANN 得到的 MSE 误差为0.000045，使用 PCFNN 得到的 MSE 误差为 0.000088，使用 PCPFNN 得到的 MSE 误差为 0.000031，由此可见构造的物理约束耦合神经网络的方法能够更有效地求解微分方程。

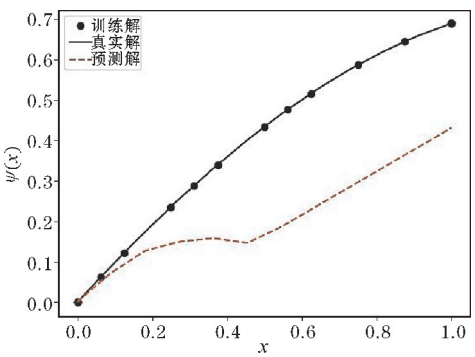

图 14 PINNs 求解算例 4

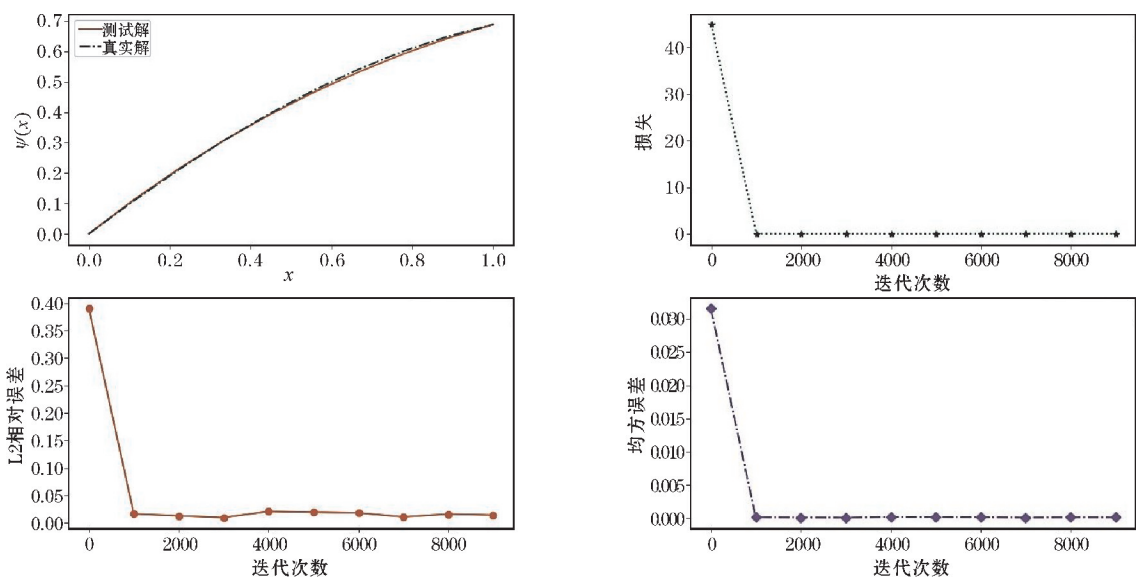

图 15 PCANN 求解算例 4

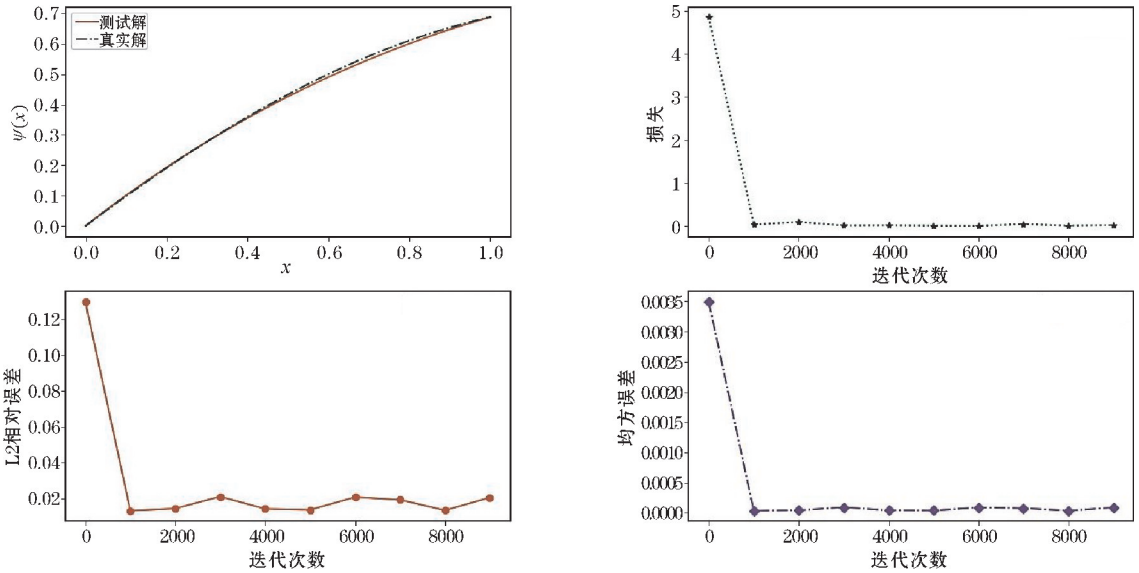

图 16　PCFNN 求解算例 4

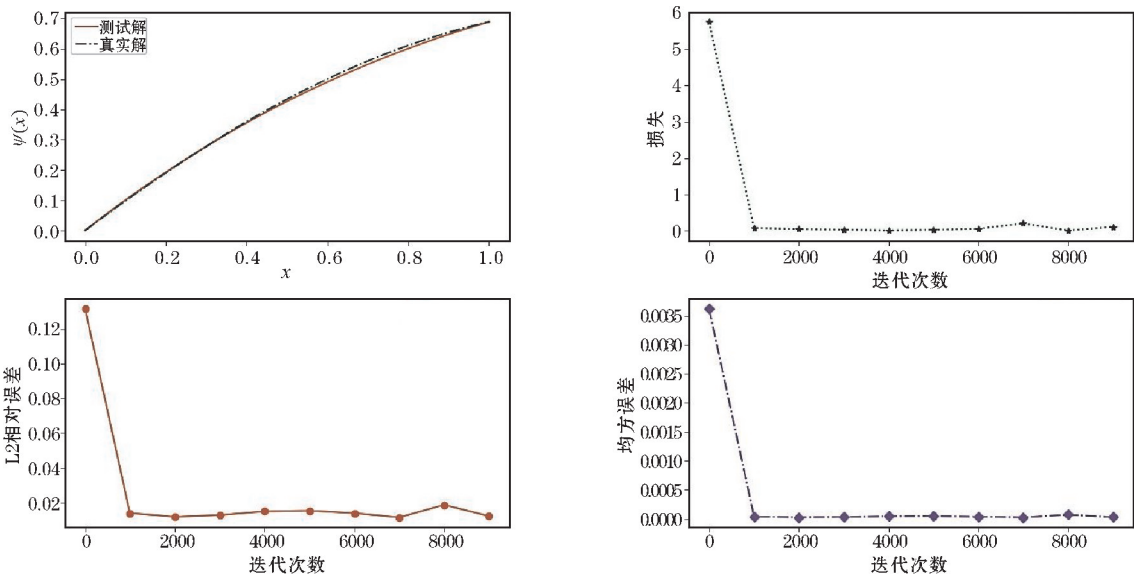

图 17　PCPFNN 求解算例 4

表 13　算例 4 用 PINNs 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.79615	0.000000	0.029777
1000	0.695240	1.933434	0.042552
2000	0.041014	0.000056	0.044771
3000	0.090503	0.001083	0.045886
4000	0.147156	0.006443	0.045460
5000	0.209601	0.023237	0.045539
6000	0.276436	0.063154	0.044060
7000	0.346263	0.142689	0.044323
8000	0.417720	0.282645	0.044935
9000	0.489505	0.506955	0.042697

表 14　算例 4 用 PCANN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.390325	0.031660	44.958839
1000	0.016664	0.000058	0.034772
2000	0.013298	0.000037	0.001599
3000	0.010190	0.000022	0.025460
4000	0.021085	0.000092	0.006878
5000	0.019745	0.000081	0.013968
6000	0.018484	0.000071	0.022794
7000	0.011333	0.000027	0.029697
8000	0.016338	0.000055	0.048034
9000	0.014668	0.000045	0.011110

表 15 算例 4 用 PCFNN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.129632	0.003492	4.862484
1000	0.013271	0.000037	0.029399
2000	0.014460	0.000043	0.091044
3000	0.021120	0.000093	0.009179
4000	0.014297	0.000042	0.014784
5000	0.013782	0.000039	0.005359
6000	0.020822	0.000090	0.000480
7000	0.019336	0.000078	0.042164
8000	0.013468	0.000038	0.006139
9000	0.020609	0.000088	0.019851

表 16 算例 4 用 PCPFNN 求解误差和损失

迭代次数	L2 误差	MSE 误差	损失
0	0.131994	0.003620	5.746288
1000	0.013823	0.000040	0.072640
2000	0.011737	0.000029	0.041471
3000	0.012690	0.000033	0.024461
4000	0.014828	0.000046	0.003996
5000	0.015243	0.000048	0.021663
6000	0.013794	0.000040	0.051606
7000	0.011402	0.000027	0.194353
8000	0.018588	0.000072	0.002897
9000	0.012116	0.000031	0.102561

5 结束语

随着数据量的增加,计算能力的提高,以及新的机器学习的出现算法(神经网络),已经成为人工智能一个有许多实际应用和积极研究的领域的话题。神经网络是实现人工智能的途径之一。这是一种机器学习,一种技术使计算机系统从经验中得到改进和数据。本文简要地描述了算法理论和应用,经过求解一阶和二阶常微分方程算例,在同等步长的情况下,对于初始条件和边界条件的情况,提出的物理约束耦合神经网络求解常微分方程方法在计算时间、计算效率以及计算精度上都有极大的优势。并且在该框架下运用不同的神经网络构造,都体现了该算法框架的明显优势。该方法的可行性、准确性以及高效性,为求解微分方程提供了有力的研究手段。

参考文献:

[1] Ujjwal K, Deep, R, Tanmay S. Multilevel Monte Carlo Finite Difference Methods For Fractional Conservation Laws With Random Data[J]. SIAM/ASA Journal on Uncertainty Quantification,2021,9(1):64-105.

[2] Lei L,Junqi Z,Chongmin S,et al. Automatic scaled boundary finite element method for three-dimensional elastoplastic analysis[J]. International Journal of Mechanical Sciences,2020,171.

[3] Pankaj K. Solution of Fokker-Planck equation by finite element and finite difference methods for nonlinear systems[J]. Sadhana-academy Proceedings in Engineering Sciences,2006,31(4):445-461.

[4] Chuanjun C,Xiaoyan Z,Guodong Z,et al. A Two-Grid Finite Element Method For Nonlinear Parabolic Integro-Differential Equations[J]. International journal of computer mathematics,2019,96(10):2010-2023.

[5] Junjiang L,Fawang L,Vo V,et al. A Space-Time Finite Element Method For Solving Linear Riesz Space Fractional Partial Differential Equations[J]. Numerical algorithms,2021,88(1):499-520.

[6] LeCun Y,Bengio Y,Hinton G. Deep learning[J]. Nature,2015,521:436.

[7] Raissi M,Perdikaris P,Karniadakis G E. Physics-informed neural networks:A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[J]. Journal of Computational Physics,2019,378:686-707.

[8] Poggio T,Mhaskar H,Rosasco L,et al. Why and when can deep-but not shallow-networks avoid the curse of dimensionality:a review[J]. International Journal of Automation and Computing,2017,14:503-519.

[9] Grohs P,Hornung F,Jentzen A,et al. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations[J]. arXiv preprint arXiv:2018,1809:02362.

[10] Raissi M,Perdikaris P,Karniadakis G E. Physics-informed neural networks:A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[J]. J. Comput. Phys. 2019,378:686-707.

[11] George Cybenko. Approximation by superpositions of a sigmoidal function[C]. In:Mathematics of

control, signals and systems 2.4, 1989:303-314.
[12] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators [C]. In: Neural Networks 2.5, 1989:359-366.

Numerical Solutions and Methods for Ordinary Differential Equations

HEI Yafang, HU Jiancheng
(College of Applied Mathematics, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: For the solution of various numerical methods of differential equations, such as finite difference and finite element methods, there exists a problem that the calculation storage and calculation time increase dramatically with the dimension of the differential equation, which seriously restricts the solution of high-dimensional problems. Neural network provides a new approach to solving differential equations, as it can infinite approximation to any nonlinear function. Through neural network training, the approximate solution of a differential equation becomes a continuous function, with enough precision to obtain any order derivative of the solution. The advantage of this method is that when the dimension of the problem increases, the increase of computation and storage is relatively small, and it can overcome the disaster of dimension and solve the high-dimension problem. At the same time, it has good generalization and the ability to solve complex regional problems. In this paper, a neural network method for solving differential equations is proposed, which is coupled by physical constraints. The numerical examples demonstrate that this method has high efficiency and good generalization, ensuring the convergence of the optimization algorithm, and the approximate solution has enough precision, which provides an effective way of solving differential equations.

Keywords: ordinary differential equation; numerical calculation; neural network