单位代码：10293 密 级：

南京邮电大学

# 硕 士 学 位 论 文

论文题目：基于智能 PID 神经网络的多变量系统解耦控制研究

| | |
|---|---|
| 学　　号 | 1020051515 |
| 姓　　名 | 吴敏 |
| 导　　师 | 丁洁 |
| 学 科 专 业 | 控制科学与工程 |
| 研 究 方 向 | 控制理论与控制工程 |
| 申请学位类别 | 工学硕士 |
| 论文提交日期 | 2023 年 4 月 |

# Research on Decoupling Control of Multivariable System Based on Intelligent PID Neural Network

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Science in Engineering


By

Wu Min

Supervisor: Prof. Ding Jie

April 2023

# 摘要

随着各类产业不断发展升级，出现了大量多变量、非线性和强耦合的复杂系统，这类系统中的耦合特性会给系统控制器设计造成困难。比例积分微分（Proportional-Integral-Derivative，PID）神经网络可以进行有效的解耦控制，但其存在初始权值随机和精确度不高两个问题，适用范围有限。因此，本文针对 PID 神经网络存在的不足，提出了一系列智能 PID 神经网络进行多变量和强耦合复杂系统的解耦控制。本文的主要工作如下：

首先，针对 PID 神经网络随机初始权值带来的训练时间长，权值易陷入局部最优以及控制初期稳定性难以保证等问题，提出了改进的天牛群算法优化 PID 神经网络的初值。改进的算法降低了天牛群算法陷入早熟收敛的概率，对 PID 神经网络初始权值具有更好的优化能力。经过仿真实验对比，验证了所提出的控制器具有更快的收敛速度和更好的解耦控制效果。

其次，针对 PID 神经网络训练过程中忽略了历史信息影响带来的精确度不足的问题，提出了基于分数阶学习算法的 PID 神经网络，其采取分数阶学习算法来进行权值的修正更新，利用分数阶微积分的记忆特性考虑了所有历史时刻权值对于当前权值更新的影响，提高了神经网络控制的精确度和收敛速度。同时，引入麻雀搜索算法对神经网络初值进行优化，避免了随机初值带来的不利影响。经过仿真实验对比，验证了所提出的控制器进一步提升了基于智能优化算法的 PID 神经网络的解耦控制精度和响应速度。

最后，针对基于分数阶学习算法的 PID 神经网络解耦控制会出现少量超调以及参数选取不合适控制输出后期易发散的问题，提出了基于分数阶神经元的分数阶 PID 神经网络，其将分数阶 PID 算法引入 PID 神经网络隐含层神经元，定义了具有动态记忆功能的分数阶 PID 神经元，可以提高控制精度、收敛速度，减少超调。同时，利用天牛群算法优化分数阶 PID 神经网络初始权值，并利用李雅普诺夫理论建立了有关学习率的一个充分条件保证了控制器的稳定性。经过仿真实验对比，验证了所提出的控制器在收敛速度、控制精度、超调和稳定性上都有显著优势。

**关键词：解耦控制，比例积分微分神经网络，智能优化算法，分数阶学习算法，分数阶神经元**

# Abstract

With the continuous development and upgrading of various industries, a large number of multivariable, nonlinear systems with strong coupling have emerged, and the coupling characteristics in such systems makes it difficult to design the controller. Proportional-Integral-Derivative（PID）neural network can carry out effective decoupling control, but it has two problems of random initial weights and low precision, so its scope of application is limited. Therefore, in this paper, a series of intelligent PID neural networks are proposed for decoupling control of multivariable systems with strong coupling to address the shortcomings of PID neural network. The main work of this paper is as follows:

Firstly, for the problems of long training time, the weights are easy to fall into local optimum and the stability of the initial control is difficult to guarantee brought by the random initial weights of PID neural network, an improved beetle swarm optimization algorithm is proposed to optimize the initial weights of PID neural network. The improved algorithm reduces the probability of the beetle swarm optimization algorithm falling into premature convergence and has better optimization capability for the initial weights of PID neural network. After simulation experiment comparison, it is verified that the proposed controller has faster convergence speed and better decoupling control effect.

Secondly, to address the problem of insufficient accuracy caused by ignoring the influence of historical information in the training process of PID neural network, a PID neural network based on fractional-order learning algorithm is proposed, which adopts a fractional-order learning algorithm to perform the correction update of weights and takes into account the influence of all historical moment weights on the current weight update by using the memory property of fractional-order calculus to improve the accuracy and convergence speed of neural network control. At the same time, the sparrow search algorithm is introduced to optimize the initial weights of the neural network, which avoids the adverse effects caused by random initial weights. After simulation experiment comparison, it is verified that the proposed controller further improves the decoupling control accuracy and response speed of PID neural network based on intelligent optimization algorithm.

Finally, a fractional-order PID neural network based on fractional-order neurons is proposed for the problem that the decoupling control of PID neural network based on fractional-order learning algorithm will have a small amount of overshoot and the parameter selection is not appropriate for the control output to be easily scattered later. Fractional-order PID neural network introduces the

fractional-order PID algorithm into the hidden layer neurons of PID neural network, and defines fractional-order PID neurons with dynamic memory function, which can improve control accuracy, convergence speed and reduce overshoot. Meanwhile, the initial weights of fractional-order PID neural network are optimized by using the beetle swarm optimization algorithm, and a sufficient condition about the learning rate is established by using Lyapunov theory to ensure the stability of the controller. After simulation and experimental comparison, it is verified that the proposed controller has significant advantages in convergence speed, control accuracy, overshoot and stability.

**Key words:** Decoupling control, Proportional-Integral-Differential neural network, Intelligent optimization algorithm, Fractional-order learning algorithm, Fractional-order neurons

# 目 录

# 第一章 绪论

## 1.1 研究背景和意义

随着我国各类产业需求不断升级，推进了各类设备自动化、数字化和智能化的转型，但同时自动化设备在实际运行过程中由于工况复杂，需要控制的变量增多可能出现系统各个回路之间存在耦合关系的现象，例如下肢外骨骼康复机器人系统[1]，双反馈的风电机组功率控制系统[2]，四旋翼无人机控制系统[3]，水稻秧棚控制系统[4]等。耦合特性指对这类多变量系统进行控制时，一个回路输入变量的改变会引起所有回路输出控制变量一起变化，各个回路之间的输入变量和输出变量会相互影响，但在实际的耦合系统中，经常需要独立地控制某些特定的变量以实现期望的性能，系统变量之间的耦合效应会导致传统控制方法很难达到对系统某个变量的单独控制，从而给系统控制带来新的挑战。因此，需要提出可高效解决多变量、强耦合复杂系统解耦控制问题的方法，为进一步促进各类产业快速转型发展提供理论研究基础。

解耦控制的目的是设计适当的解耦器或控制器，消除各个回路间的耦合关系，使得多变量耦合系统中的每个输出变量仅受某个特定的输入变量所影响。随着解耦控制研究的不断进步，目前已经出现了很多有效的解耦控制方法，如自适应控制[5-6]，无模型控制[7] ，模糊控制[8-9]和神经网络控制[10-12]等。这些解耦控制方法在某些情况下可以获得良好的控制效果，但是也存在一些限制，如自适应控制器实际上需要在线辨识系统模型，对大规模系统在线辨识计算量很大且扰动较多，辨识难度较大，无模型控制器适用的系统范围有限，而模糊控制需要根据经验制定一套合适的模糊规则，这对于复杂的实际系统应用较为困难。神经网络控制器由于其神经元输入输出的静态特性不能单独作为控制器使用，需要附加另外的控制器，增加了复杂度。因此，如何设计提出模型简便、适用性广、计算量合适且可以快速获得高精度解耦控制效果的解耦控制器是多变量、强耦合系统解耦控制研究的一个重要课题。

比例积分微分（Proportional-Integral-Derivative，PID）控制由于结构简单、易于操作在工业控制中具有广泛的应用，但其面对复杂系统中的不确定性以及耦合特性时难以达到理想控制效果，而神经网络的自学习与无限逼近能力为控制具有耦合特性的非线性动态系统提供了有价值的研究方向[13-14]，PID 神经网络（PID Neural Network，PIDNN）很好的结合了二者的优点[15]。PIDNN 是由舒怀林教授提出的一种新型动态神经网络，它将 PID 算法引入到 PIDNN 的隐含层神经元，采用比例、积分和微分函数作为神经元的状态转换函数使得神经

元可以获得比例，积分和微分的动态效果，结合神经网络强大的非线性映射特性和任意函数逼近能力，不仅可以单独作为控制器使用，且可以根据反馈误差自适应的调整权值，从而可以实现多变量、强耦合复杂系统的解耦控制，具有广泛的应用前景。

经典 PIDNN 可以有效实现多变量、强耦合复杂系统的解耦控制，但为了获得更好的控制效果，需要研究 PIDNN 中的两个具有挑战性的问题。首先，PIDNN 的初始连接权值对于保证控制器的性能具有重要意义，PIDNN初始权值是随机数，会影响网络收敛速度和初期控制的稳定性，同时还会增大权值陷入局部最优的概率，如何优化初始权值是一个需要解决的关键问题。此外，如何提高神经网络训练的准确性是另一个需要深入研究的重要问题，PIDNN在训练过程中没有考虑历史信息对当前更新的影响，控制误差较大，响应速度较慢，如何提高控制的精确度是研究的一个重点方向。因此，本文针对 PIDNN 初值随机、精确度不高等问题，提出收敛速度更快、控制精度更高、稳定性更好，适应性更强的智能 PIDNN 解耦控制器，提高 PIDNN 在实际系统应用的广泛性，这对实现各类产业中多变量、强耦合系统的高效解耦控制具有重要的理论研究意义以及实际应用价值。

## 1.2 国内外研究现状及发展趋势

### 1.2.1 解耦控制研究现状

控制系统中的耦合问题是由 Boksenbom 和 Hood 在 1949 年提出的[16]，但直至工业农业等领域快速发展，耦合现象影响到系统控制问题，解耦控制问题才得到了广泛关注。20 世纪 80 年代，Waller 教授正式在报告中研究了多变量系统中的耦合问题[17]，我国的钱学森教授也在控制理论初期就开始研究耦合系统的解耦控制问题，随着解耦控制理论的不断完善以及各类产业控制系统逐渐变得复杂，对耦合系统的控制要求逐渐变高，解耦控制逐渐成为研究的热点问题。针对多变量、非线性系统中的复杂耦合关系，国内外学者提出了很多解耦控制方法，如自适应解耦控制、模糊解耦控制和神经网络解耦控制等方法，这些解耦控制方法在不同类型的耦合系统上具有良好表现，但适用范围有限且很难将解耦和控制融合处理，具有一定的复杂度。因此，继续研究复杂度低，可以同时完成解耦和控制两个目标且能广泛适用的解耦控制方法非常必要。

Boksenbom 和 Hood 率先提出的使被控系统的传递函数成为对角线的解耦思想对解耦控制具有指导性意义[16]，但其建立在系统输入、输出个数相同的条件下，适用范围有限。随后出现了一些更通用的解耦控制方法，文献[18]针对球磨煤粉碎系统提出了一种基于多模型和

神经网络的解耦控制方法，通过解耦补偿器处理耦合效应，并采用神经网络来处理非线性影响，获得了良好的解耦控制效果，并成功得到了实际应用。文献[19]将控制回路之间的耦合关系视为干扰，选择交互最少的控制进行配对，设计了多变量 PID 分数阶滤波器多回路控制器，控制效果具有良好的鲁棒性。这些解耦控制方法不要求系统输入、输出相同，可以对一些复杂的多变量、强耦合系统进行有效的解耦控制，但大都需要依赖被控对象的精确模型，阻碍了其应用的广泛性。

随着智能控制的兴起，解耦控制出现了一个新的分支，将模糊控制、神经网络等智能控制方法引入解耦控制，为高效的在线精确解耦控制提供了有力的理论支撑。文献[20]提出了一种由神经网络逆系统和鲁棒控制器组成的复合解耦控制方法，利用神经网络逼近系统的逆系统，避免了底盘系统中前转向和悬架子系统控制间的耦合关系，获得了良好的解耦效果，提高了底盘系统的性能。文献[21]利用模糊神经网络构建永磁同步发电机中发电电压和悬挂力的逆系统，提出了一种基于模糊神经网络逆系统的解耦控制方法，实现了发电电压和悬挂力之间的动态解耦。文献[22]提出了一种基于 PID 神经网络和滤波 $x$ 最小均方算法的噪声解耦控制方法，实现了多通道有源噪声系统的解耦控制，且收敛速度快，误差小，获得了良好的降噪效果。这些智能解耦控制方法在各种不同类型的系统上展现了更好的解耦控制性能，可以针对时变的系统进行精确地在线解耦控制，但对于类型不同的复杂耦合系统，智能解耦控制方法适应性也不同，面对日益复杂的实际系统，仍然需要继续研究适应性更广、控制效果更佳的解耦控制方法。

解耦控制一直是控制领域一个活跃的课题，目前国内外解耦技术大体可以分为以下几类：

（1） 传统解耦控制

传统解耦控制结构简单，易于实现，应用较为广泛，其核心思想是利用精确的数学模型，设计解耦补偿器，与被控对象组成广义系统，使该广义系统的传递函数形成对角矩阵，从而使多变量的耦合系统在外部解耦成为无耦合的单变量系统[23]。传统解耦控制方法一般需要精确的数学模型，适用于确定的线性多变量系统，难以实现自适应控制，且其是在固定模型基础上实现解耦，一旦系统在控制过程中产生变动，解耦补偿器就会失效，从而失去解耦效果。因此，对于状况复杂的非线性、强耦合的多变量系统，传统解耦控制方法很难达到良好的控制指标。

（2） 自适应解耦控制

自适应控制主要分为基于模型的方法和数据驱动方法，相较于传统控制方法对时变系统具有更好的容错性[24]。自适应解耦控制是在传统的前馈控制上进行了自适应控制的改进，来适应实际系统控制过程中工况的改变，其核心思想是将对系统的辨识和解耦控制同时在线进

行，将系统各变量间的耦合关系看作系统的一种可测干扰项，通过自校正的前馈控制来耦合的部分进行静态或动态的补偿，并根据工况对补偿器中的参数进行实时的优化[25]。自适应解耦控制在传统解耦控制的基础上向前迈进了一大步，通过辨识与解耦控制的结合达到了在变化的系统中实际可用的成果，但其需要对系统模型进行在线辨识，对于较为复杂的系统需要巨大的运算量，且针对复杂工况的系统，辨识出来的数学模型与实际控制过程存在很大误差，系统的鲁棒性也无法保证，从而难以得到理想的解耦控制性能。因此，对于自适应解耦控制在实际系统中的控制效果以及鲁棒性仍需进一步深入研究。

（3） 智能解耦控制

智能解耦控制是目前解耦控制研究中的一个热点方向，在非线性复杂系统的解耦控制上具有独特的优势，其通过将神经网络等人工智能方法引入解耦控制领域，实现多变量、非线性系统的高效在线精确解耦，开辟了一个解耦控制的新领域[26]。目前针对多变量、非线性和强耦合复杂系统的主要智能解耦控制方法有模糊解耦控制和神经网络解耦控制等方法。

模糊解耦控制分为直接解耦和间接解耦，直接解耦可以对被控对象解耦或对控制器解耦，间接解耦法是对模糊规则进行子空间分解，但两种模糊解耦方法的核心思想都是制定合适的模糊规则来描述系统的耦合部分，通过反馈补偿来完成解耦[27]。文献[28]针对非线性、强耦合的高炉最高压力控制问题，提出了一种模糊解耦控制方法，使用两个模糊 PID 控制器、一个模糊解耦控制器和两个校正调节器组合起来的控制系统进行解耦控制，将顶压误差保证在了合理范围内。文献[29]提出了一种混合的模糊解耦控制方法用于解决集成电路封装磁浮系统的解耦控制。通过线性解耦项实现磁浮系统中的主要耦合关系的解耦，再利用非线性的智能补偿项解耦非线性耦合部分，只需近似线性模型即可实现响应快速、误差小的解耦控制性能。

模糊解耦控制可以根据实际经验归纳出模糊规则进行解耦控制，无需精确的系统模型，且对系统实际运行过程中的工况、环境等条件的变化具有较强的容错能力，但模糊解耦控制的关键在于模糊规则的制定，由于实际系统的工况复杂，各个生产环节环环相扣，且随着被控系统耦合维数的增加，模糊规则设计的难度也在不断增加，这就导致模糊解耦控制运算量过大且解耦效率不高，阻碍了其大规模的推广使用。

神经网络解耦控制的核心思想是利用神经网络的自学习、自适应和非线性映射等能力来逆向训练出被控系统的模型，然后根据被控目标自适应得调整权值，实现系统的解耦控制。文献[30]采用反向传播神经网络的逆系统解耦控制实现了主动电磁轴承-转子系统的解耦控制，通过神经网络构建了被控系统的逆系统，再与原系统进行级联，使被控系统解耦成为多个独立的子系统，达到了高性能解耦控制效果。文献[31]针对混合储能系统的控制问题，采用

Hammerstein 型神经网络来辨识模型，根据模型信息制定了 PID 神经网络，对系统进行自适应控制，在四区互联电力系统上实现了有效的解耦控制。

神经网络解耦控制使用了神经网络的非线性映射以及自学习特性，在多变量、强耦合系统中具有优异表现，可以完成在线的精确解耦。虽然其目前还处于探索阶段，但它在多变量、强耦合复杂系统中优秀的解耦控制表现使得其成为研究的热点，且目前已有很多研究成果，在实际工业农业等领域已有初步应用[32-33]。神经网络展现了高效的解耦控制能力，但静态的神经网络通常不能单独作为控制器实现解耦控制，需要附加额外的控制器，这样就增加了控制的复杂度，研究发展具有动态特性的神经网络是一个充满前景的研究方向。

（4） 其他解耦控制方法

随着对解耦控制理论不断的研究，很多交叉融合的算法被引入到解耦控制中，且达到了良好的控制效果。如文献[34]在双独立电驱动履带车驱动系统中采用了反馈线性化实现车辆纵向速度和偏航角速率之间的解耦，再设计自适应广义预测控制对子系统进行跟踪控制，实现了车辆在各种不同情况下平稳行进。文献[35]提出了一种基于 Caputo 分数阶微分的 BP-PID 解耦控制方法，使用 Caputo 定义下的分数阶梯度下降法对 BP-PID 算法进行优化，在电极锅炉系统中实现了高效解耦控制，响应速度快且超调小。这些解耦控制方法多为各类算法的融合，结合各自的特点和优点获得了良好的解耦控制性能，在针对一些特定问题上具有不错的表现，但普适性还有待提高。

随着解耦控制方法研究不断深入，解耦控制领域不断涌现出新型的研究成果，但随着现代产业不断发展升级，耦合系统的复杂度不断提升，耦合的维度也不断扩大，解耦控制还有很多需要完善发展的地方，研究获得适应性更强，模型更简单，控制精确度更高的解耦技术是一个亟待解决的重大课题。

### 1.2.2 PID 神经网络研究现状

PIDNN 是由舒怀林教授于 1997 年提出的一种新型动态神经网络，为神经网络的发展提供了一种新的思路，它不是简单的使用神经网络整定 PID 参数，而是保留了神经网络的结构，根据 PID 算法定义出具有比例、积分和微分功能的神经元，将静态神经元扩充到动态神经元，这些神经元的输入输出具有动态特性，很好的完成了各种信息的融合处理，使神经网络可以单独作为控制器使用，在多变量系统控制中具有良好表现，实用性更强[36]。PIDNN 一经提出便获得了学术界以及工程应用界的关注，已经在被广泛应用于各类非线性系统和耦合系统的控制[37-39]。

PIDNN 在国内的研究起步较晚，但 PIDNN 保留了神经网络的非线性映射特性以及容错性，同时结合了 PID 控制结构简单、调节方便、适应性强等优良特点，可以实现辨识与控制的同步进行，且对于多变量系统可以通过调节子网络数量进行很好地适应，因此，在实际工程控制领域具有很多研究成果。文献[40]中，PIDNN 被应用于油田分层中的注水工艺控制，针对其中非线性以及时滞特性表现出良好的控制效果，控制精度高，鲁棒性好。文献[41] 使用 PIDNN 控制加热器的温度，在各种未知的天气变化和干扰下获得比 PID 控制更好的控制效果，实现了高精度的温度控制。文献[42]针对水下航行器的垂直平面控制问题，提出了一种改进的 PIDNN 控制器，适应获得了良好的全局控制性能。文献[43]使用 PIDNN 对晶体生长炉的温度进行控制，获得了更加精准的加热效果。文献[44]中，PIDNN 被用于节流阀的实时控制，以提高钻井的效率和安全性。在这些应用中，PIDNN可以适应各类不同实际系统中复杂的耦合关系，不确定性以及非线性，以简便的模型获得了良好的控制效果，证明了 PIDNN 在实际系统应用中的巨大潜力。目前国内外学者对 PIDNN 初始权值的确定以及网络连接权值的更新算法进行了深入的研究，提出了很多改进的 PIDNN，解决了其随机初始权值问题，并获得了更优的解耦控制效果。

PIDNN结合了PID控制和神经网络的优点，立足于实用性，在一些非线性、多变量、强耦合复杂系统的控制方面具有良好表现，但 PIDNN 初始权值的选取依赖于 PID 控制经验，如果没有 PID 控制经验，则初始权值为无规律的随机数，不仅会使网络收敛时间变长、权值易陷入局部最优，甚至可能导致控制初期不稳定，且其采用梯度下降法对权值进行反向修正，没有考虑网络训练过程中历史信息对当前参数更新的影响，精确度有待提升。因此，针对 PIDNN 的改进成为研究的热点。

针对 PIDNN 初始权值随机的问题，智能优化算法是一个非常有力的初值优化工具，目前已有很多算法被用于 PIDNN 初值的优化[45-47]。此外，很多研究者将各类不同的算法引入 PIDNN 来提高其精确度，如文献[48]将改进的梯度下降算法引入反馈神经网络更新网络权值，获得了更高的精确度，减少了超调。经过各类算法改进后的 PIDNN 收敛速度更快，控制效果得到了一定的提升，进一步扩大了 PIDNN 的适用范围，但随着科技的不断进步，一些设备对系统控制的要求也越来越高，所以需要对 PIDNN 进行更加深入的研究，使其获得更高的精确度，更快的收敛速度以及更加良好的稳定性。因此，从 PIDNN 内部的算法与神经元的结构入手，提出可以提高解耦控制性能，增强 PIDNN 普适性的智能 PIDNN 具有重要实际应用意义。

### 1.2.3 智能优化算法改进 PID 神经网络研究现状

PIDNN 的初始连接权值影响着网络收敛的起始点、初始收敛方向以及控制初期的稳定性，但传统 PIDNN 的初始权值不易确定，一部分有 PID 控制经验的系统可以根据已有的控制经验确定初始权值，而大部分系统无法获得可靠经验，只能选取随机数作为初始权值。随机初始权值会导致网络从随机的方向开始收敛，再通过不断地学习来找到正确的收敛方向，这个过程不仅会导致网络收敛速度变慢，权值易陷入局部最优且无法保证控制初期控制效果的稳定性。因此，研究 PIDNN 初始权值的优化问题非常必要。

近年来，已经有许多智能优化算法应用于优化神经网络。文献[49]使用量子免疫优化算法对反馈神经网络进行优化，提高了神经网络的准确性。文献[50]提出了一种反时限衰减算子的混沌郊狼优化算法对反馈神经网络的参数进行优化，使神经网络获得了更加高效的分类性能。在文献[51]中，将蚁群算法引入高速列车定位的机器学习算法中，解决了单 $k$ 均值算法的过拟合问题，提高了算法的收敛速度和精确度。这些应用展现了智能优化算法对于神经网络参数具有良好的优化效果，因此，很多学者将各类智能优化算法引入 PIDNN 对其最优初始权值进行搜索，例如遗传算法[52-53]、鱼群算法[54]、粒子群算法[55-56]和云遗传算法[57]等，这些算法在 PIDNN 的不同应用中展现了不俗的优化效果。智能优化算法具有很好的避免局部最优能力以及良好的稳定性和灵活性可以更好的处理复杂优化问题[58]，大量研究成果证明了智能优化算法是确定 PIDNN 最优初始权值的有力工具。但是没有一种算法可以完美解决所有优化问题，不同的算法适用的领域不同，且某些算法也存在一些缺陷，因此，新算法的提出以及针对不同算法的改进仍需进行深入研究。

### 1.2.4 分数阶理论在控制领域研究现状

分数阶微积分的概念和整数阶微积分是同时代出现的，但三个世纪中分数阶微积分的研究工作一直限制在纯数学理论方面，直至 1960 年，分数阶微积分在具有记忆和遗传效应的实际系统中的优良特性引起学者的关注[59]，计算机等科学技术的飞速发展也为分数阶微积分的研究提供了有力支撑，分数阶微积分的研究开始走向实际工程控制领域。法国学者 Oustaloup 教授所领导研究组将提出的分数阶鲁棒控制方法成功应用于汽车工业的悬挂控制[60]证明了分数阶微积分在实际应用中的可行性，极大地促进了分数阶微积分的发展与研究，在经典控制算法、神经网络等领域得到了很多应用[61-62]。

分数阶微积分经过不断地研究和发展，有了三种常见定义：Grünwald-Letnikov（G-L）、

Riemann-Liouville（R-L）和 Caputo 定义，其中 G-L，R-L 定义常用于离散系统，而 Caputo 定义更适用于连续系统[63]。分数阶微积分可以描述任意阶次的导数和积分，分数阶次增加了一个自由度，丰富了相关动力学行为[64]，且其良好的记忆特性和遗传特性使得分数阶算法可以获得比整数阶微积分更高的精确度[65]。正由于分数阶展露的优势，很多算法分数阶算法被引入控制领域并获得了广泛的应用，如分数阶 PID 算法、分数阶神经网络等。文献[66]中，Podlubny 教授提出了分数阶 PID 控制器的模型，相较于 PID 控制器多了两个可修改的参数有助于快速准确地达到所需的目标，因此比 PID 控制器更加稳健，在控制领域具有广泛的应用[67-68]。文献[69]中提出了一种有源元件模拟实现的分数阶控制器，并将其用于直流电机的控制，达到了无电阻、节能的效果。在文献[70]中，提出了一种改进的分数阶控制器对电动汽车的负载频率进行控制，达到了预期的控制效果。此外，许多类型的分数阶神经网络正在出现，在文献[71]中，提出了一种自适应分数阶反向传播神经网络，采用分数阶梯度下降法方法来更新神经网络训练中的连接权重，可以获得更好的准确性。文献[72]提出了一种分数阶卷积神经网络，利用 Caputo 型分数梯度方法来实现偏差和权重的有效动态更新。

分数阶微积分理论是整数阶微积分理论的拓展和延伸，具有良好的应用背景和完整的理论体系，其优异的记忆特性可以解决控制领域很多算法精确度不足的问题，在与一些应用广泛的控制算法的结合方面也具有了很多的研究成果以及实际应用案例，具有广阔的应用前景。

## 1.3 本文主要研究内容

随着各类产业智能化的发展，实际系统不断升级，呈现出输入、输出变量越来越多，变量间存在复杂耦合关系的变化，对系统控制提出了更大的挑战。在此背景下，本文主要研究多变量、强耦合复杂系统的解耦控制问题，利用 PIDNN 作为基本控制框架，考虑智能优化算法，分数阶学习算法以及分数阶 PID 算法的优势改善 PIDNN 在解耦控制中存在的收敛速度慢，控制误差较大以及稳定性难以保证等问题，提出了一系列具有更加优越的解耦控制性能的智能 PIDNN 控制器，以解决此类系统的高精度解耦控制问题。

本文的主要创新点如下：

（1）提出了一种改进的天牛群算法搜索 PIDNN 的最优初始权值，避免使用随机初始权值，减少了训练过程中权值陷入局部最优的概率，保证了 PIDNN 初始运行的稳定性，加快了收敛速度。

（2）在智能优化算法优化初始权值的基础上，将分数阶学习算法引入 PIDNN 权值修正步骤，提出了基于分数阶学习算法的 PIDNN，分数阶学习算法的长记忆特性提高了控制的

精确度。同时使用麻雀搜索算法获得其最优初始权值，可以获得更加快速精确的解耦控制效果。

（3）首次将分数阶神经元应用于 PIDNN，提出了基于分数阶神经元的分数阶 PID 神经网络（$PI^{\lambda}D^{\mu}$ neural network，$PI^{\lambda}D^{\mu}NN$），同时引入天牛群算法寻找最优的 $PI^{\lambda}D^{\mu}NN$ 初始权值，结合分数阶 PID 算法的具有记忆能力、稳定性好的特性和神经网络良好的解耦控制能力，所提控制器能够更精确地更新网络权值、加速收敛、减少超调、保证稳定性，获得更好的静态和动态控制性能。

本文的主要内容如下：

第一章首先介绍了多变量、强耦合系统解耦控制的发展历程，阐述了其研究的必要性以及重要性，分析了传统 PIDNN 在解耦控制领域的研究现状与需要研究解决的问题，随后介绍了智能优化算法优化 PIDNN 的发展历程，以及分数阶微积分理论在控制领域的发展与研究现状，最后总结了研究内容。

第二章首先对 PIDNN 的基本结构以及解耦控制原理进行了详细介绍，阐明了 PIDNN 的解耦优势，其次分析了 PIDNN 随机初始权值可能带来的不良影响。然后着重介绍了天牛群算法并对其不足进行改进，提出了一种改进的天牛群算法，利用改进后的算法优化 PIDNN 初始权值，最后给出并分析了其解耦控制仿真结果。

第三章首先介绍了分数阶微积分的概念，将分数阶学习算法引入 PIDNN 的权值更新过程，推导出分数阶的权值更新公式。其次介绍了麻雀搜索算法，并利用其对基于分数阶学习算法的 PIDNN 进行最优初始权值搜索。随后阐述了基于麻雀优化算法的分数阶学习策略的 PIDNN 的解耦控制方案，最后给出并分析了其解耦控制仿真结果。

第四章首先介绍了分数阶 PID 控制器，推导定义出了分数阶比例（P），积分（$I^{\lambda}$）和微分（$D^{\mu}$）神经元，将 PIDNN 隐含层神经元使用分数阶神经元进行替换，给出了 $PI^{\lambda}D^{\mu}NN$ 的结构并描述了 $PI^{\lambda}D^{\mu}NN$ 的解耦流程。其次对 PIDNN 进行了稳定性分析，推导出了一个有关学习率的稳定条件。最后采用天牛群算法对初始权值进行寻优，给出并分析了基于天牛群算法的 $PI^{\lambda}D^{\mu}NN$ 的解耦控制仿真结果。

第五章是总结与展望，总结了本文所完成的具体工作以及得到的研究成果，展望了进一步的研究方向。

# 第二章 基于智能优化算法的 PID 神经网络解耦控制

很多现代工业、农业系统中存在多个输入和输出变量，而且由于实际环境的复杂性，这类系统中的控制变量往往会互相影响，形成耦合关系，这会给系统控制带来不小的问题。很多传统的控制方法无法在耦合情况下实现对系统中某个单一变量的理想控制。因此，需要对复杂的多变量、强耦合系统进行有效的解耦控制。PIDNN是由舒怀林教授首次提出的一种结合了PID控制和神经网络优点的动态神经网络，神经网络强大的非线性映射能力赋予PIDNN实现耦合系统解耦控制的能力，而隐含层比例、积分和微分神经元的动态特性，使得PIDNN可以单独作为控制器实现解耦控制，但其连接权值的初始值是随机选取的，会很大程度的影响控制效果，采用智能优化算法可以很好地解决随机初值问题。

本章中，第一小节描述了 PIDNN 控制器结构，详细推导了 PIDNN 的输入输出、各层神经元的状态以及各层权值更新算法，分析了 PIDNN 解耦控制的优势所在以及其在初值选取方面的劣势。第二小节介绍了天牛群算法的优化原理，并针对其易陷入早熟收敛提出了一种改进的天牛群算法，采用改进的算法搜索 PIDNN 的最优初始权值。第三小节对提出的基于改进的天牛群算法的 PIDNN 控制器进行了实例仿真，其结果表明了所提出控制器可以加快收敛速度，更好的进行解耦控制。

## 2.1 PID 神经网络相关知识

### 2.1.1 PID 神经网络结构

PIDNN 是一种三层前馈神经网络，包含输入层、隐含层和输出层，其在结合了 PID 控制和神经网络的特点的基础上，通过反馈误差不断更新修正各层连接权值来实现控制目标，并输出合适的控制输入到被控对象来实现解耦控制。PIDNN按照输出值不同可以分为单变量PIDNN（SPIDNN）和多变量 PIDNN（MPIDNN）。SPIDNN 的结构为 2×3×1，如图 2.1 所示，其隐含层神经元个数是固定的，分别为比例（P）、 积分（I）和微分（D）神经元，能够完成类似 PID 控制器的功能，即比例，积分和微分功能，其输入信号由输入层进行输入，经过 P，I 和 D 神经元的运算，最终由输出层输出结果。

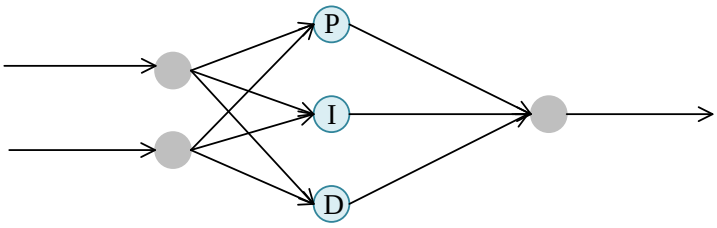


图 2.1　SPIDNN 结构图

MPIDNN 由 SPIDNN 扩展而来，其网络结构由被控对象输入和输出的个数决定。如图 2.2 所示，针对被控对象是具有 $m$ 个输入和 $n$ 个输出的耦合系统，MPIDNN 结构为 $2m\times3m\times1n$，由 $n$ 个并行的子网络组成，其中子网络的数目等于被控系统输出变量的数目。每个子网络由一个输入层、一个隐藏层和一个输出层组成，输入层有两个神经元，一个接收系统所需的目标信号，另一个连接系统实际输出，隐含层含有 P、I 和 D 神经元。在图 2.2 中，$r_p\left(p=1,2,\cdots,n\right)$ 表示理想信号，$y_p$ 表示被控对象的实际输出，$x_o\left(o=1,2,\cdots,m\right)$ 表示 PIDNN 的输出。

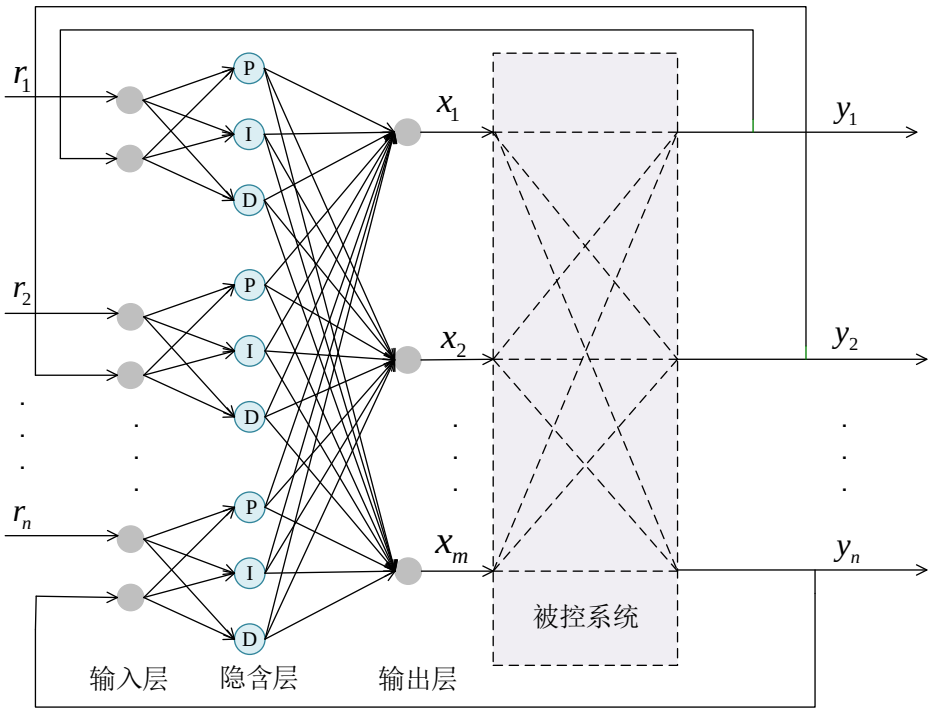


图 2.2　MPIDNN 解耦控制结构图[36]

PIDNN按照前向算法获得网络输出，即根据其输入，结合每层当前的连接权重以及状态函数得到输出。每一层输入和输出具体如下[73]：

（1） 输入层

输入层的输入为：

$$\begin{aligned} N_{s1}\left(k\right)&=r_s\left(k\right)\\ N_{s2}\left(k\right)&=y_s\left(k\right)\end{aligned} \tag{2.1}$$

式中，$s$ 为子网络的序列号，$k$ 为采样时刻，$s1$ 表示第 $s$ 个子网络输入层第一个神经元，$s2$ 表示第 $s$ 个子网络输入层第二个神经元。

输入层的状态函数是单位比例函数，所以输入层的状态为：

$$v_{si}(k)=N_{si}(k) \tag{2.2}$$

式中，$i$ 代表输入层的第 $i(i=1,2)$ 个神经元。

输入层的输出为：

$$x_{si}(k)=\begin{cases}1, & v_{si}(k)>1\\ v_{si}(k), & -1\le v_{si}(k)\le 1\\ -1, & v_{si}(k)<1\end{cases} \tag{2.3}$$

（2） 隐含层

MPIDNN 包含 $s$ 个子网络，每个子网络的隐含层都包含 P，I 和 D 神经元。

隐含层输入为：

$$N'_{sh}(k)=\sum_{i=1}^{2}\omega_{sih}(k)x_{si}(k) \tag{2.4}$$

式中，$h(h=1,2,3)$ 为隐含层神经元个数，$\omega_{sih}$ 为输入层到隐含层的连接权重，上标 “'” 表示隐含层变量。

隐含层中 P 神经元的状态函数为单位比例函数，其状态为：

$$v'_{s1}(k)=N'_{s1}(k) \tag{2.5}$$

隐含层中 I 神经元的状态函数为单位积分函数，其状态为：

$$v'_{s2}(k)=v'_{s2}(k-1)+N'_{s2}(k) \tag{2.6}$$

隐含层中 D 神经元的状态函数为单位微分函数，其状态为：

$$v'_{s3}(k)=N'_{s3}(k)-N'_{s3}(k-1) \tag{2.7}$$

隐含层神经元输出为：

$$x'_{sh}(k)=\begin{cases}1, & v'_{sh}(k)>1\\ v'_{sh}(k), & -1\le v'_{sh}(k)\le 1\\ -1, & v'_{sh}(k)<1\end{cases} \tag{2.8}$$

（3） 输出层

输出层每个神经元的输入为：

$$N_o^{''}(k)=\sum_{h=1}^{3}\omega_{sho}(k)x_{sh}^{'}(k) \tag{2.9}$$

式中，$o(1,2,\cdots,m)$为输出层神经元个数，$\omega_{sho}$为隐含层到输出层的连接权重，上标“''”表示输出层变量。

输出层中的状态函数是单位比例函数，所以神经元的状态为：

$$v_o^{''}(k)=N_o^{''}(k) \tag{2.10}$$

输出层的输出为：

$$x_o^{''}(k)=\begin{cases}1, & v_o^{''}(k)>1\\ v_o^{''}(k), & -1\le v_o^{''}(k)\le 1\\ -1, & v_o^{''}(k)<1\end{cases} \tag{2.11}$$

PIDNN 的输出$x_o(k)$与输出层输出相同，即为：

$$x_o(k)=x_o^{''}(k) \tag{2.12}$$

### 2.1.2 PID 神经网络反向算法

PIDNN各层间的连接权重按照梯度下降算法进行更新，神经网络根据系统的目标信号与系统实际输出之间的误差不断的进行训练学习，使得成本函数最小化，最终达到控制目标。

定义 PIDNN 的成本函数为[36]：

$$J=\sum_{p=1}^{n}E_p=\frac{1}{l}\sum_{p=1}^{n}\sum_{k=1}^{l}\left[r_p(k)-y_p(k)\right]^2 \tag{2.13}$$

式中，$E_p$为理想信号与系统实际输出的偏差平方均值，$l$为采样点数，$n$为被控变量个数。

定义学习率为$\eta$，则各层的权重由如下梯度下降法进行更新：

$$\boldsymbol{\omega}(k+1)=\boldsymbol{\omega}(k)-\eta\frac{\partial J}{\partial\boldsymbol{\omega}(k)} \tag{2.14}$$

式中，$\boldsymbol{\omega}$代表输入层到隐含层权重$\omega_{sih}$和隐含层到输出层权重$\omega_{sho}$。

根据式（2.14），PIDNN 各层权重的更新公式总结如下[73]：

（1） 隐含层到输出层

隐含层到输出层的连接权值可按式（2.15）更新：

$$\omega_{sho}(k+1)=\omega_{sho}(k)-\eta_1\frac{\partial J}{\partial\omega_{sho}(k)} \tag{2.15}$$

式中，$\eta_1$为隐含层到输出层的学习率，且

$$\frac{\partial J}{\partial\omega_{sho}(k)}=\sum_{p=1}^{n}\frac{\partial J}{\partial E_p}\frac{\partial E_p}{\partial x_o(k)}\frac{\partial x_o(k)}{\partial x_o''(k)}\frac{\partial x_o''(k)}{\partial v_o''(k)}\frac{\partial v_o''(k)}{\partial N_o''(k)}\frac{\partial N_o''(k)}{\partial\omega_{sho}(k)} \tag{2.16}$$

由式（2.13），可得：

$$\sum_{p=1}^{n}\frac{\partial J}{\partial E_p}\frac{\partial E_p}{\partial x_o(k)}=-\frac{2}{l}\sum_{p=1}^{n}\sum_{k=1}^{l}\left[r_p(k)-y_p(k)\right]\frac{\partial y_p(k)}{\partial x_o(k)} \tag{2.17}$$

由于被控对象参数未知，$y_p$对$x_o$偏微分的计算会产生困难，但此项的大小仅影响网络的收敛速度，但其正负会决定网络的收敛方向，因此，采用$y_p$和$x_o$的相对变化量的符号函数来近似$y_p$对$x_o$的偏微分，即：

$$\sum_{p=1}^{n}\frac{\partial J}{\partial E_p}\frac{\partial E_p}{\partial x_o(k)}=-\frac{2}{l}\sum_{p=1}^{n}\sum_{k=1}^{l}\left[r_p(k)-y_p(k)\right]\operatorname{sgn}\left[\frac{y_p(k)-y_p(k-1)}{x_o(k)-x_o(k-1)}\right] \tag{2.18}$$

由式（2.12），可得：

$$\frac{\partial x_o(k)}{\partial x_o''(k)}=\frac{\partial x_o''(k)}{\partial x_o''(k)}=1 \tag{2.19}$$

由式（2.11），可得：

$$\frac{\partial x_o''(k)}{\partial v_o''(k)}=\frac{\partial v_o''(k)}{\partial v_o''(k)}=1 \tag{2.20}$$

由式（2.10），可得：

$$\frac{\partial v_o''(k)}{\partial N_o''(k)}=\frac{\partial N_o''(k)}{\partial N_o''(k)}=1 \tag{2.21}$$

由式（2.9），可得：

$$\frac{\partial N_o''(k)}{\partial\omega_{sho}(k)}=\frac{\partial}{\partial\omega_{sho}(k)}\left[\sum_{h=1}^{3}\omega_{sho}(k)x_{sh}'(k)\right]=\sum_{h=1}^{3}x_{sh}'(k) \tag{2.22}$$

由式（2.18）~（2.22），可以推导出：

$$\frac{\partial J}{\partial\omega_{sho}(k)}=-\frac{2}{l}\sum_{h=1}^{3}\sum_{p=1}^{n}\sum_{k=1}^{l}\left[r_p(k)-y_p(k)\right]\operatorname{sgn}\left[\frac{y_p(k)-y_p(k-1)}{x_o(k)-x_o(k-1)}\right]x_{sh}'(k) \tag{2.23}$$

令：

$$\varphi(k)=\frac{2}{l}\sum_{p=1}^{n}\sum_{k=1}^{l}\left[r_p(k)-y_p(k)\right]\operatorname{sgn}\left[\frac{y_p(k)-y_p(k-1)}{x_o(k)-x_o(k-1)}\right] \tag{2.24}$$

则可得：

$$\frac{\partial J}{\partial \omega_{sho}(k)}=-\sum_{h=1}^{3}\varphi(k)x_{sh}^{'}(k) \tag{2.25}$$

（2） 输入层到隐含层

输入层到隐含层的连接权值可按式（2.26）更新：

$$\omega_{sih}(k+1)=\omega_{sih}(k)-\eta_2\frac{\partial J}{\partial \omega_{sih}(k)} \tag{2.26}$$

式中，$\eta_2$ 为输入层到隐含层的学习率，且

$$\frac{\partial J}{\partial \omega_{sih}(k)}=\sum_{p=1}^{n}\frac{\partial J}{\partial E_p}\frac{\partial E_p}{\partial x_o(k)}\frac{\partial x_o(k)}{\partial x_o^{''}(k)}\frac{\partial x_o^{''}(k)}{\partial v_o^{''}(k)}\frac{\partial v_o^{''}(k)}{\partial N_o^{''}(k)}\frac{\partial N_o^{''}(k)}{\partial x_{sh}^{'}(k)}\frac{\partial x_{sh}^{'}(k)}{\partial v_{sh}^{'}(k)}\frac{\partial v_{sh}^{'}(k)}{\partial N_{sh}^{'}(k)}\frac{\partial N_{sh}^{'}(k)}{\partial \omega_{sih}(k)} \tag{2.27}$$

由式（2.18）~（2.21）和式（2.24），可以得出：

$$\sum_{p=1}^{n}\frac{\partial J}{\partial E_p}\frac{\partial E_p}{\partial x_o(k)}\frac{\partial x_o(k)}{\partial x_o^{''}(k)}\frac{\partial x_o^{''}(k)}{\partial v_o^{''}(k)}\frac{\partial v_o^{''}(k)}{\partial N_o^{''}(k)}=-\varphi(k) \tag{2.28}$$

由式（2.9），可得：

$$\frac{\partial N_o^{''}(k)}{\partial x_{sh}^{'}(k)}=\frac{\partial}{\partial x_{sh}^{'}(k)}\left[\sum_{h=1}^{3}\omega_{sho}(k)x_{sh}^{'}(k)\right]=\sum_{h=1}^{3}\omega_{sho}(k) \tag{2.29}$$

由式（2.8），可得：

$$\frac{\partial x_{sh}^{'}(k)}{\partial v_{sh}^{'}(k)}=\frac{\partial v_{sh}^{'}(k)}{\partial v_{sh}^{'}(k)}=1 \tag{2.30}$$

式（2.27）中的 $\frac{\partial v_{sh}^{'}(k)}{\partial N_{sh}^{'}(k)}$ 做以下处理：

$$\frac{\partial v_{sh}^{'}(k)}{\partial N_{si}^{'}(k)}\approx\frac{\Delta v_{sh}^{'}(k)}{\Delta N_{sh}^{'}(k)}=\operatorname{sgn}\frac{v_{sh}^{'}(k)-v_{sh}^{'}(k-1)}{N_{sh}^{'}(k)-N_{sh}^{'}(k-1)} \tag{2.31}$$

由于 $\frac{v_{sh}^{'}(k)-v_{sh}^{'}(k-1)}{N_{sh}^{'}(k)-N_{sh}^{'}(k-1)}$ 的正负会影响连接权值的下一步的收敛方向，且其绝对值大小会影响连接权值的收敛速度，故式（2.31）采用 $v_{sh}^{'}(k)$ 和 $x_{sh}^{'}(k)$ 相对变化量的符号函数来对其进行代替，这样可以在不影响学习效果的前提下简化计算。

由式（2.4），可得：

$$\frac{\partial N_{sh}^{'}(k)}{\partial \omega_{sih}(k)}=\frac{\partial}{\partial \omega_{sih}(k)}\left[\sum_{i=1}^{2}\omega_{sih}(k)x_{si}(k)\right]=\sum_{i=1}^{2}x_{si}(k) \tag{2.32}$$

由式（2.28）~（2.32），推导可得：

$$\frac{\partial J}{\partial \omega_{sih}(k)}=-\sum_{i=1}^{2}\sum_{h=1}^{3}\omega_{sho}(k)\operatorname{sgn}\frac{v_{sh}^{'}(k)-v_{sh}^{'}(k-1)}{N_{sh}^{'}(k)-N_{sh}^{'}(k-1)}x_{si}(k)\varphi(k) \tag{2.33}$$

PIDNN根据以上规律实时调节各层间的连接权值，可以使得被控系统中的输出变量按照给定的理想信号变化，达到解耦控制的目的。

### 2.1.3 PID 神经网络解耦控制原理

PIDNN通过将 PID 控制算法融入神经网络的隐含层神经元，使得静态的神经元具有了动态特性，因此，PIDNN可以单独作为解耦控制器来实现解耦控制，且神经网络的自学习特性和强大的任意非线性映射能力让 PIDNN 可以无需系统模型就完成控制目标，将解耦与控制融合。PIDNN会根据理想信号与实际输出的误差不断的学习和修正权值以最小化目标函数，最终获得一个输出，将 PIDNN 的输出作为被控对象的输入，经过一系列执行器的动作后，该输入可以使得系统的被控变量达到目标信号，即被控系统获得理想输出，完成解耦控制的要求。

PIDNN 的完整控制方案如图 2.3 所示，其中$\boldsymbol{e}(k)=\boldsymbol{r}(k)-\boldsymbol{y}(k)$是系统理想输出与实际输出的差值，$\boldsymbol{x}(k)$是 PIDNN 的输出，同时也是被控系统的输入。

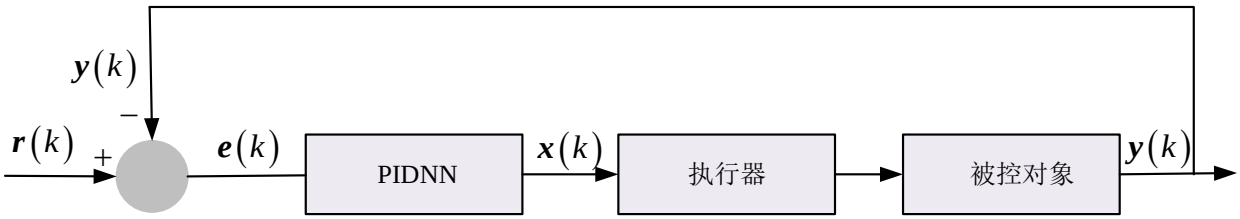


图 2.3　PIDNN 闭环解耦控制方案图

此方案中，被控系统的实际输出$\boldsymbol{y}$会实时反馈到 PIDNN 的输入层神经元，与输入层神经元获取的设定目标值$\boldsymbol{r}$比较，获得差值$\boldsymbol{e}$，PIDNN 根据差值不断的进行训练，调整权值，使得$\boldsymbol{e}$不断缩小直至近零，此时 PIDNN 的输出$\boldsymbol{x}$即为可以使被控系统达到理想信号的正确输入，PIDNN 的每一个子网络控制一个被控系统的输出变量，因此，只需将 PIDNN 的输出来输入给执行器，经过一系列的执行动作后被控对象所有的输出$\boldsymbol{y}$会达到目标值$\boldsymbol{r}$，被控系统即达到解耦控制的目的。

## 2.2 基于改进天牛群优化算法的 PID 神经网络解耦控制

PIDNN具有很多方面的应用，在多变量、强耦合复杂系统的解耦控制方面很具优势。然而，传统的 PIDNN 的初始权值的选取并无固定算法，一般是随机选取的。神经网络的初始权值代表着网络收敛的初始方向和学习的起点，因此，随机初值会带来以下几个问题：

（1）影响收敛速度

随机选取的初值会使得神经网络从一个随机的方向进行训练，如果想要获得正确的收敛方向，必须花费大量的时间进行反复的训练学习，从而导致网络收敛速度变慢。

（2）容易陷入局部最优

由于 PIDNN 的连接权值采取反传算法进行更新，随机的起始点会增加权值收敛陷入局部最优的概率。

（3）影响 PIDNN 解耦控制的稳定性

PIDNN作为控制器来实现系统的解耦控制，随机的初始权值会导致网络初始工作的不稳定，实际应用中，如果控制器初始状态不稳定，系统会无法进入正常工作状态。无法做到全过程稳定是影响 PIDNN 应用的重要因素。

想要提高 PIDNN 控制效果，必须选取合适的初始权值，智能优化算法已被证明是一种优化 PIDNN 初值的有效方法。通过采用智能优化算法搜索 PIDNN 的最优初始权重，可以加快神经网络的收敛速度，同时稳定神经网络的控制效果，提高控制性能。

### 2.2.1 智能优化算法

天牛群算法（Beetle Swarm Optimization Algorithm，BSO）由王甜甜等人[74]提出的一种群体智能优化算法，它结合了粒子群算法（Particle Swarm Optimization，PSO）和天牛须算法（Beetle Antennae Search，BAS）的特点，通过天牛之间的信息共享来更新搜索过程，从而获得全局最优解。BSO 算法兼具探索性和开发性，结合了天牛觅食的搜索机制和群体优化机制，提高了算法的快速性、准确性和稳定性。BAS 算法是通过模拟单个天牛觅食机制来进行寻优的，计算简单，只需 4 个步骤就可以完成对最优解的搜索，BSO 算法将粒子群中的群体机制引入到 BAS 算法中，其通过模拟天牛群的觅食机制来进行对最优解的搜素，具有运行速度快、鲁棒性强的优点，可以有效、稳定地处理多目标优化问题。

（1） BAS 算法

BAS 算法是 2017 年提出的一种仿天牛觅食机制的个体智能优化算法[75]。天牛觅食仅靠

一对触须感受食物的气味浓度来寻找食物，如果左边食物气味浓度强就往左边飞，反之亦然。食物的气味在三维空间中不同点具有不同的值，天牛的一对触须可以获得触须所在处两点的气味浓度值，天牛的目标是获得气味浓度最大的值，即食物的所在处。BAS 算法将天牛的搜索行为推广到任意维度的空间，实现函数的寻优。

天牛在寻优时搜寻的方向是随机的，即左右触须间的矢量方向也需要是随机的，采用如下随机向量对其进行标准化：

$$\vec{a}=\frac{rands(q,1)}{\left\|rands(q,1)\right\|} \tag{2.34}$$

式中，$rands(\cdot)$ 为随机函数，$q$ 代表了待优化问题的维度。

天牛左右触须的位置可以按照式（2.35）更新：

$$\begin{aligned} P_r^{t+1}&=P_r^t+\vec{a}/2*d^t \\ P_l^{t+1}&=P_l^t-\vec{a}/2*d^t \end{aligned} \tag{2.35}$$

式中，$P_r^t$ 代表天牛右须在第 $t$ 次迭代中的位置，$P_l^t$ 代表天牛左须在第 $t$ 次迭代中的位置，$d^t$ 代表着第 $t$ 次迭代中两须之间的距离。

天牛左右两须的气味浓度可以利用适应度函数值来表示，分别为 $f\left(P_r^t\right)$ 和 $f\left(P_l^t\right)$。然后根据式（2.36）的天牛迭代机制来更新天牛的位置：

$$P^{t+1}=P^t+\delta^t*\vec{a}*\mathrm{sign}\left(f\left(P_r^t\right)-f\left(P_l^t\right)\right) \tag{2.36}$$

式中，$\mathrm{sign}(\cdot)$ 为符号函数，$P^t$ 表示天牛在第 $t$ 次迭代中的位置，$\delta^t$ 表示天牛第 $t$ 次迭代中的步长，通常随着迭代次数的增加而减小，可以按照式（2.37）更新：

$$\delta^t=z_1\delta^{t-1}+\delta^0 \tag{2.37}$$

式中，$z_1$ 为正的常数，$\delta^0$ 为步长的初始值。

步长与两须间距离的关系如下[76]：

$$d^t=\delta^t/z_2 \tag{2.38}$$

式中，$z_2$ 是一个正的常数。

BAS 算法的流程如图 2.4 所示。

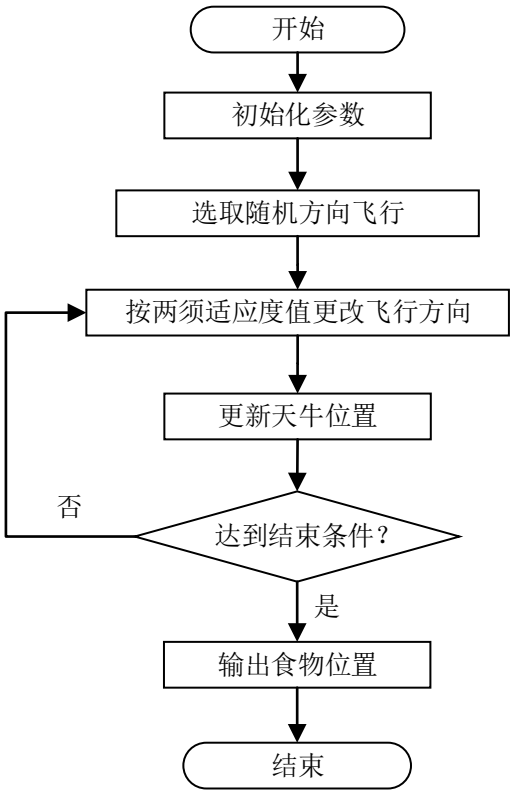


图 2.4　BAS 算法流程图

（2） PSO 算法

PSO 算法是由 Eberhart 和 Kennedy 受鸟群捕食行为激发而提出的一种群体智能优化算法[77]。群体中的个体会互相协作，共享信息，综合自己获得的最优位置和群体的最优位置来更新位置，最终使得群体获得正确的位置方向，最终寻找到食物。PSO 算法利用无质量的粒子模拟鸟群中的个体，粒子群就是一组搜索空间的有效解。

PSO 算法中的每个粒子都具有位置和速度两个重要属性，粒子的位置更新公式为：

$$P_{iq}^{t+1} = P_{iq}^{t} + V_{iq}^{t} \tag{2.39}$$

式中，$P_{iq}^{t}$ 表示第 $i$ 个粒子在第 $q$ 维度的第 $t$ 次迭代中的位置，$V_{iq}^{t}$ 表示第 $i$ 个粒子在第 $q$ 维度的第 $t$ 次迭代中的速度。

粒子的速度更新公式为：

$$V_{iq}^{t+1} = \vartheta^{t} V_{iq}^{t} + b_1 z_3 \left( L_{iq}^{t} - P_{iq}^{t} \right) + b_2 z_4 \left( L_{gq}^{t} - P_{iq}^{t} \right) \tag{2.40}$$

式中，$L_{iq}^{t}$ 代表第 $i$ 个粒子在第 $q$ 维度的第 $t$ 次迭代中的个体极值，而 $L_{gq}^{t}$ 代表群体在第 $q$ 维度的第 $t$ 次迭代中的群体极值，$b_1$ 和 $b_2$ 是两个学习因子，$z_3$ 和 $z_4$ 是位于 $[0,1]$ 的随机数，$\vartheta^{t}$ 是惯性权重，可以按式（2.41）更新：

$$\vartheta^{t} = \vartheta_{\max} - \frac{\vartheta_{\max} - \vartheta_{\min}}{T} t \tag{2.41}$$

式中，$T$ 是迭代次数的最大值，$\vartheta_{\max}$ 和 $\vartheta_{\min}$ 分别代表 $\vartheta$ 的最大值和最小值。

PSO 算法的流程如图 2.5 所示。

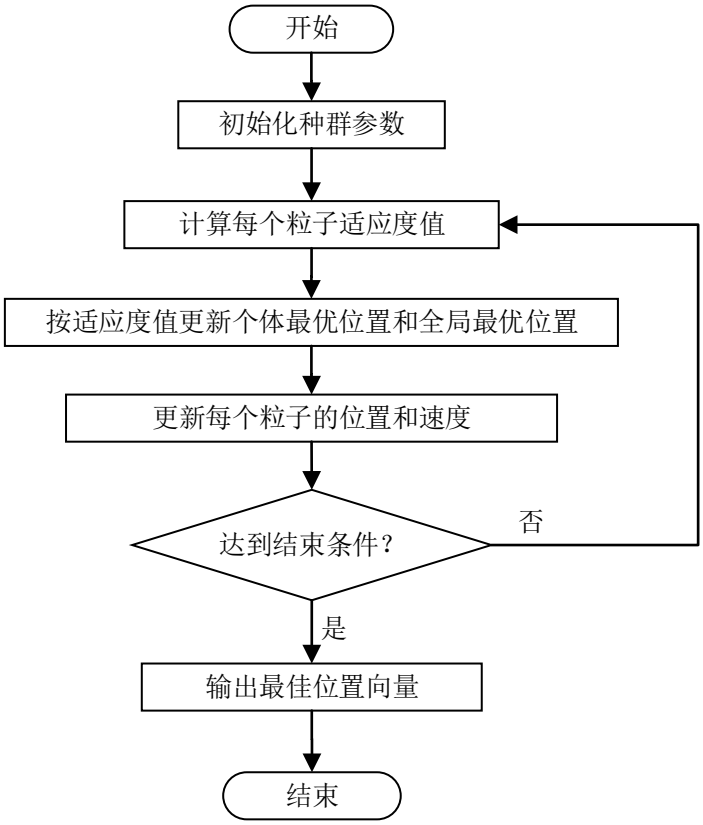


图 2.5　PSO 算法流程图

（3） BSO 算法

BAS 算法中仅有一个天牛个体，在处理高维函数的问题上性能并不理想，并且天牛的初始位置很大程度上决定了寻优的结果。因此，结合 PSO 算法中群体合作的思想，将 BAS 算法中的个体扩展成为群体，即 BSO 算法。在 BSO 算法中，每只天牛对应一个适应度值，这是确定最佳位置的基础，天牛之间也会分享信息，天牛的飞行方向取决于它们触须探测到的信息的强度，而飞行距离取决于飞行速度[76]。根据天牛群觅食原理，在 $q$ 维空间中，天牛位置和速度更新如下[74]：

天牛个体的位置更新公式为：

$$P_{iq}^{t+1} = P_{iq}^{t} + \beta V_{iq}^{t} + (1-\beta)\xi_{iq}^{t} \tag{2.42}$$

式中，$P_{iq}^{t}$ 表示第 $i$ 个天牛在第 $q$ 维度的第 $t$ 次迭代中的位置，$V_{iq}^{t}$ 表示第 $i$ 个天牛在第 $q$ 维度的第 $t$ 次迭代中的速度，$\beta$ 是正的常数，$\xi_{i}^{t}$ 表示增量函数，可由式（2.43）更新:

$$\xi_{iq}^{t+1} = \delta^{t} V_{iq}^{t} sign\left( f\left(P_{rq}^{t}\right) - f\left(P_{lq}^{t}\right)\right) \tag{2.43}$$

天牛左右触须的位置可按式（2.44）更新：

$$P_{rq}^{t+1} = P_{rq}^{t} + V_{iq}^{t} * d^{t} / 2$$
$$P_{lq}^{t+1} = P_{rq}^{t} - V_{iq}^{t} * d^{t} / 2 \tag{2.44}$$

天牛的速度可以更新为:

$$V_{iq}^{t+1} = \vartheta^{t} V_{iq}^{t} + b_1 z_3 \left( L_{iq}^{t} - P_{iq}^{t} \right) + b_2 z_4 \left( L_{gq}^{t} - P_{iq}^{t} \right) \tag{2.45}$$

式中，$L_{iq}^{t}$ 是个体的极值位置，$L_{gq}^{t}$ 是种群的极值位置，$\vartheta^{t}$ 为惯性权重，可以按式（2.46）更新：

$$\vartheta^{t} = \vartheta_{\max} - \frac{\vartheta_{\max} - \vartheta_{\min}}{T} t \tag{2.46}$$

BSO 算法的流程如图 2.6 所示。

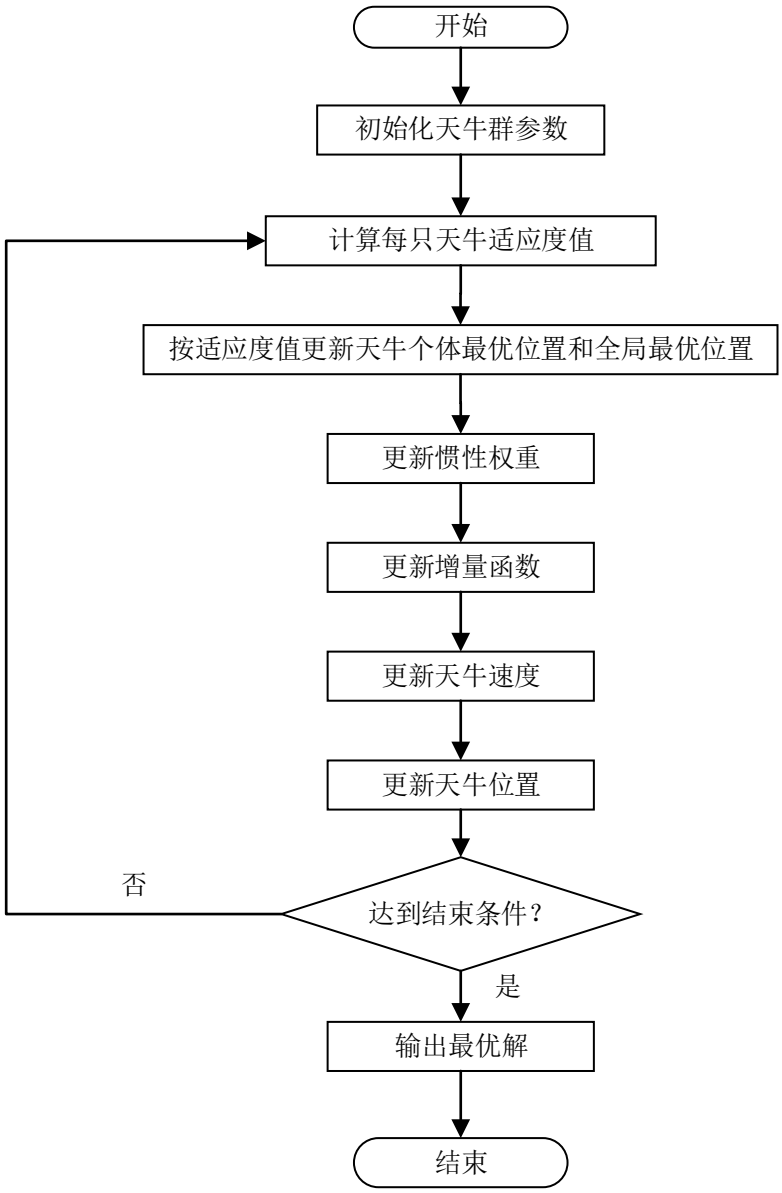


图 2.6　BSO 算法流程图

### 2.2.2 改进的天牛群算法

BSO 算法借鉴了 BAS 算法和 PSO 算法的优点，达到探索能力和开发能力的平衡发展，具有良好的全局优化特性。但 BSO 算法依旧存在早熟收敛的问题，因此，本节提出了一种改进的天牛群优化算法（Improved Beetle Swarm Optimization Algorithm，IBSO），该算法在天牛速度更新中采用非线性递减方式更新惯性权重，可以在搜索过程的后期控制天牛的速度，增强天牛的局部搜索能力，并在种群体的全局最优位置利用变异因子增加天牛种群的多样性，从而降低算法过早收敛的概率，可以提高算法的优化能力。

IBSO 算法中，天牛个体的位置更新公式同式（2.42），速度更新公式同式（2.45），但式（2.45）中的惯性权重修改为如下非线性方式更新[78]：

$$\vartheta^t = \gamma_1\left(1-\tanh\left(\gamma_2\left(\frac{t}{T}-\gamma_3\right)\right)\right)+\gamma_4 \tag{2.47}$$

式中，tanh 是双曲正切函数，$\gamma_1$ 和 $\gamma_4$ 用于调整 $\vartheta$ 的极值，$\gamma_2$ 用于控制极值过渡区的陡度，$\gamma_3$ 用于控制搜索范围，均为正数，$T$ 为最大迭代次数。

式（2.47）的非线性方式更新惯性权重可以在控制过程中根据不同阶段来更好地调节天牛的飞行速度，在前期提高全局搜索性能，后期提高局部搜索性能，更高效地完成最佳位置搜索。

针对 BSO 算法在全局最优解的位置具有陷入局部最优的风险，在其全局最优位置中引入一个突变因子，该突变因子可以使种群最优位置在每一次迭代时都会产生变异，这种变异会引导天牛种群改变飞行方向，通过飞到不同的区域寻找潜在的最优解，减少算法陷入早熟收敛的风险。

采用一个随机扰动量来代表突变因子，则每次迭代时全局最优位置都需要按式（2.48）进行更新[79]：

$$L_{gq}^t = L_{gq}^t\left(1+\chi rand\upsilon\right) \tag{2.48}$$

式中，$rand\upsilon$ 是一个服从标准正态分布，且均值和单位方差为零的随机变量，$\chi$ 是一个介于 $[0,1]$ 之间的突变因子。

IBSO 算法的流程如图 2.7 所示。

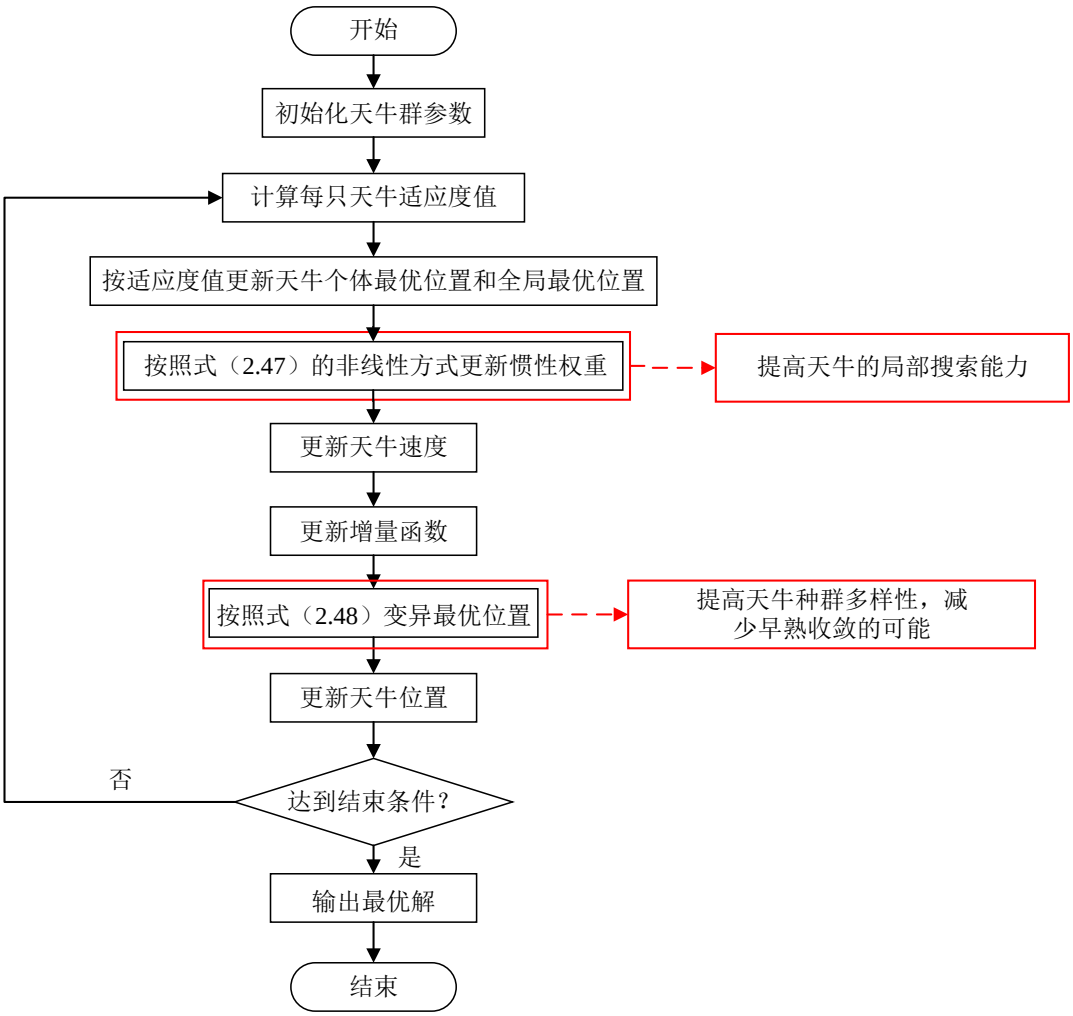


图 2. 7　IBSO 算法流程图

IBSO 算法以非线性方式更新惯性权重，并在最优位置加入一个变异因子，以帮助天牛平衡局部搜索和全局搜索能力，提高了种群的多样性，降低了算法陷入早熟收敛的概率，使算法具有更好的寻优性能。

### 2.2.3 基于改进的天牛群优化算法的 PID 神经网络

PIDNN初值的选取不仅关系到神经网络收敛速度，而且对于控制起始阶段的稳定性至关重要，传统 PIDNN 任意选取初值的方式会影响控制效果。本节中采用 IBSO 算法寻找 PIDNN 的最优初始权值，提出基于 IBSO 算法的 PIDNN（IBSO-PIDNN），IBSO 算法会根据适应度函数来确定最优解，得到最优的初始权值。

被控对象的输入和输出的个数决定了 PIDNN 中子网络的个数，PIDNN 的结构确定后，即可得到 IBSO 算法所需训练的连接权值。IBSO 算法可以根据适应度函数进行衡量，寻找到适应度最好的位置，即 PIDNN 的最优初始权值。将 PIDNN 的输出作为被控对象的输入，对被控系统进行控制调节，达到设定的目标值，即完成解耦控制的目的。

定义如下适应度函数：

$$f(t)=\sum_{t=1}^{T}e^{0.01\cdot t}\left(\sum_{k=1}^{length}\sum_{p=1}^{n}\left|r_p(k)-y_p(k)\right|\right) \tag{2.49}$$

式中，$T$ 为 IBSO 算法的最大迭代次数，*length* 为 PIDNN 训练次数。

算法 2.1 给出了 IBSO-PIDNN 的解耦控制算法的伪代码：

算法 2.1： IBSO-PIDNN 解耦控制算法

输入：$\boldsymbol{r},\boldsymbol{y}$
输出：$\boldsymbol{x}_o^{''}$

初始化 IBSO 算法参数，给定 PIDNN 的训练次数
IBSO 算法迭代次数：$T$
PIDNN 训练次数：*length*
*while* $(t<T)$ *do*
　计算并比较每个位置的适应度值
　*for* $t=1:T$ *do*
　　通过式（2.47）更新惯性权重 $\vartheta$
　　通过式（2.45）更新天牛速度 $V_{iq}$
　　通过式（2.37）更新步长 $\delta$
　　通过式（2.43）更新增量函数 $\xi$
　　通过式（2.48）变异全局最优位置 $L_{gq}$
　　通过式（2.42）更新当前搜索的位置 $P_{iq}$
　*if* $f(t)<f_{pbest}(t)$
　　$f(t)=f_{pbest}(t)$
　*if* $f(t)<f_{gbest}(t)$
　　$f(t)=f_{gbest}(t)$
　计算当前适应度值，如果优于个体极值或群体极值，更新它
　$t=t+1$
　*end for*
*end while*
输出当前最优位置向量
PIDNN 以 IBSO 算法输出的最优解作为初始权值
*for* $k=1:length$ *do*
　　通过式（2.3）计算 $x_{si}(k)$
　　通过式（2.26）更新输入层到隐含层权值
　　通过式（2.8）计算 $x_{sh}^{'}(k)$
　　通过式（2.15）更新隐含层到输出层权值
　　通过式（2.11）计算 $x_o^{''}(k)$
*end for*
*end*

IBSO 算法中，非线性递减惯性权重可以在最优区域确定时控制收敛速度减慢，增强局部搜索能力，变异因子可以提高种群的多样性，因此，降低了早熟收敛的概率。由于对 IBSO 算法进行了优化，可以更加快速准确的搜索到 PIDNN 的最优初始权值，这样网络可以在初始阶段就获得正确的收敛和学习方向，不仅可以加快收敛速度，还可以保证网络初期控制的稳定性，使得 PIDNN 能够获得更加理想的解耦控制性能。

## 2.3 仿真实例

考虑一个三输入两输出的温室系统，该系统输入为加热、加湿和通风，输出为温度和湿度。温室系统中的控制变量是相互耦合，例如，对温室进行加湿不仅会增加湿度，还会导致温度降低，同理其他变量的改变都会同时影响到温度和湿度，导致温度和湿度难以同时达到设定的理想值，其简化模型如下[80]：

$$\begin{cases} y_1(k)=0.897y_1(k-1)+0.202x_1(k)-0.371x_2(k)+0.250x_3(k) \\ y_2(k)=0.400y_2(k-1)+0.250x_1(k)+0.500x_2(k)-0.250x_3(k)[y_2(k-1)-0.450] \end{cases} \tag{2.50}$$

其中，$\boldsymbol{y}=[y_1,y_2]^{\mathrm{T}}$ 和 $\boldsymbol{x}=[x_1,x_2,x_3]^{\mathrm{T}}$ 分别是系统输出和输入向量。目标信号设置为：$\boldsymbol{r}=[0.5,0.7]^{\mathrm{T}}$。

针对此多变量、强耦合温室系统，使用 PIDNN、基于 BSO 算法的 PIDNN（BSO-PIDNN）和 IBSO-PIDNN 控制器对其进行解耦控制，并对比其仿真结果。学习率设置为 $\eta_1=0.0006$，$\eta_2=0.0001$，训练次数设置为 200，天牛个体设置为 50 只。IBSO 算法参数设置为：$b_1=b_2=1.5$，$\gamma_1=0.25$，$\gamma_2=\gamma_4=0.5$，$\gamma_3=2$，$\chi=0.1$，$T=45$。

图 2.8 和图 2.9 分别显示了温室系统在传统 PIDNN、BSO-PIDNN 和 IBSO-PIDNN 三种解耦控制器下的输出响应和输出误差收敛曲线，图 2.10 显示了 IBSO-PIDNN 的输出。BSO 算法和 IBSO 算法的演化过程分别如图 2.11 和图 2.12 所示，表 2.1 列出了温室系统的输出变量在三种不同解耦控制器下的最小追踪时间，$T_s$ 代表系统输出变量达到理想信号所需的最短时间。

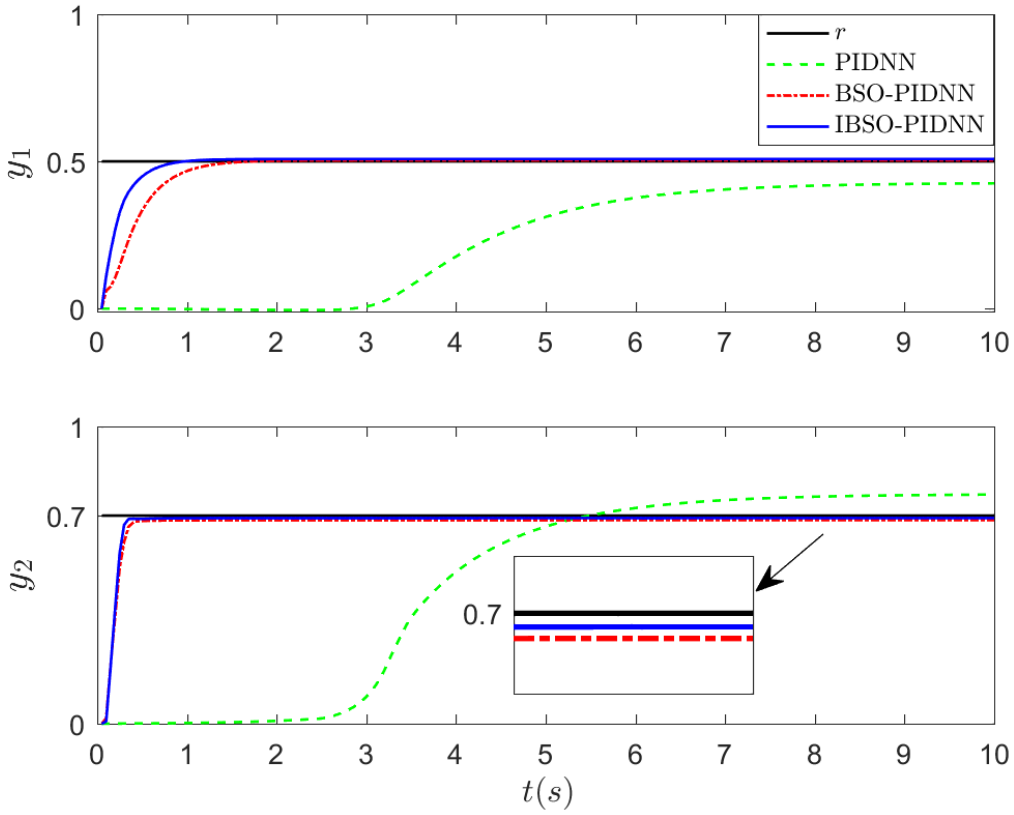


图 2.8　不同解耦控制器下温室系统的输出响应

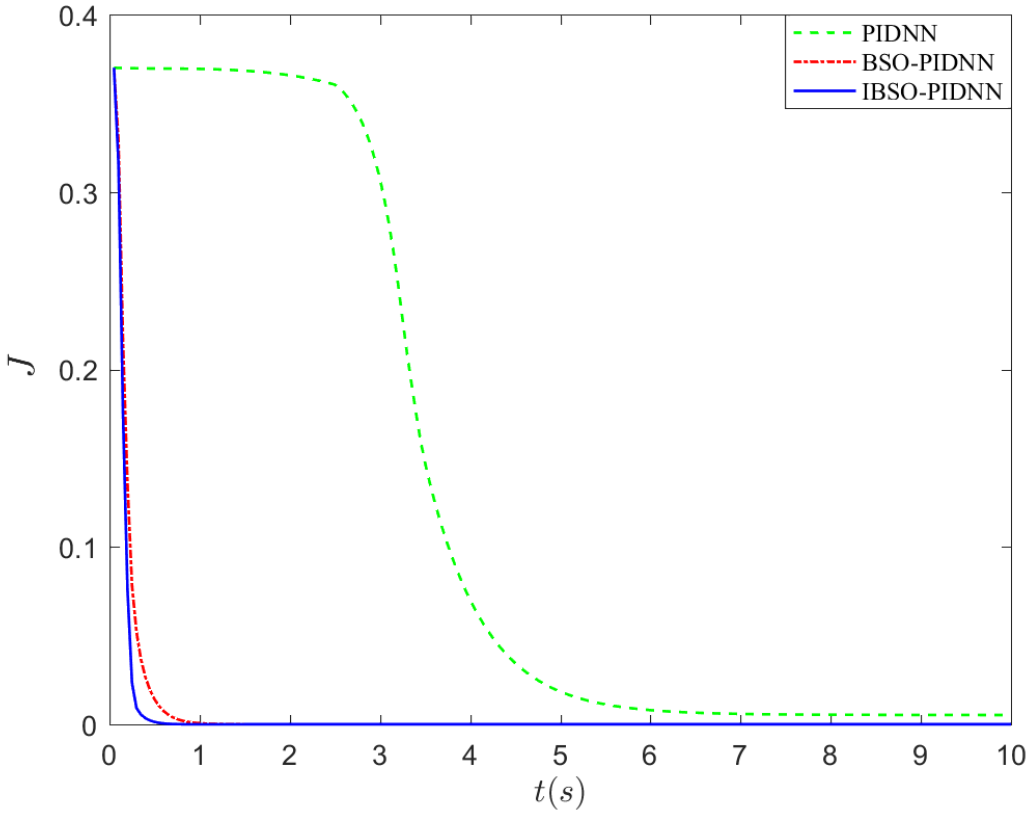


图 2.9　不同解耦控制器下温室系统的输出误差收敛曲线

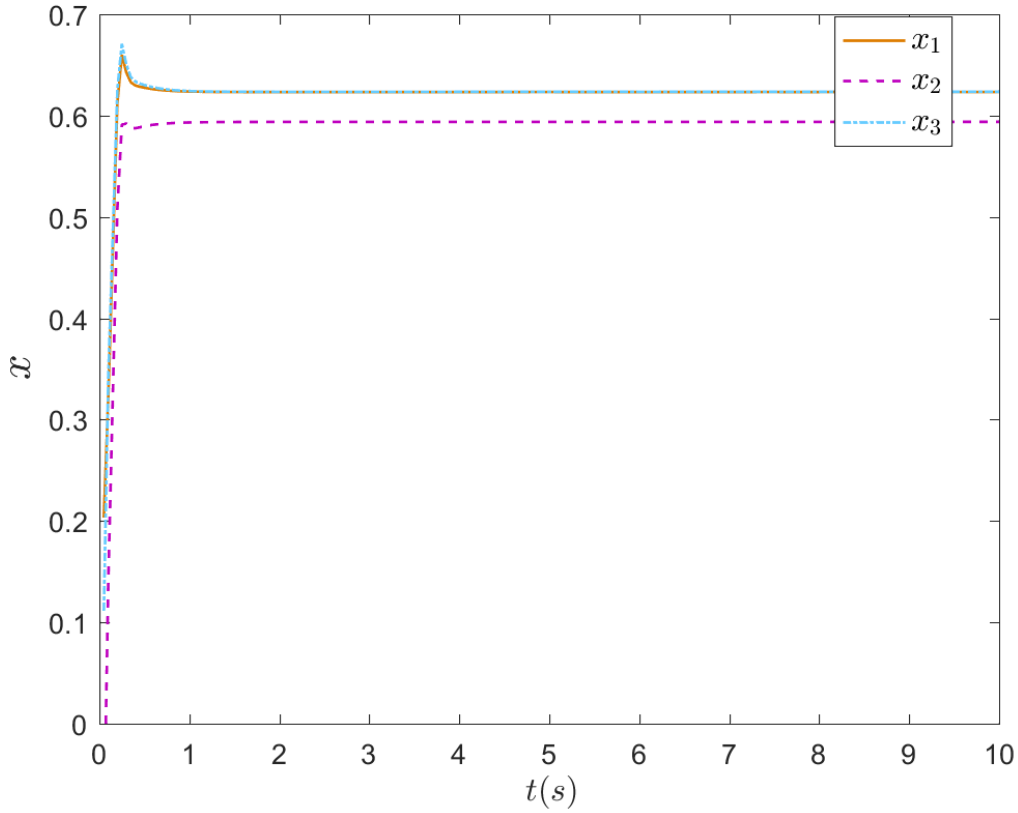


图 2. 10 IBSO-PIDNN 的输出

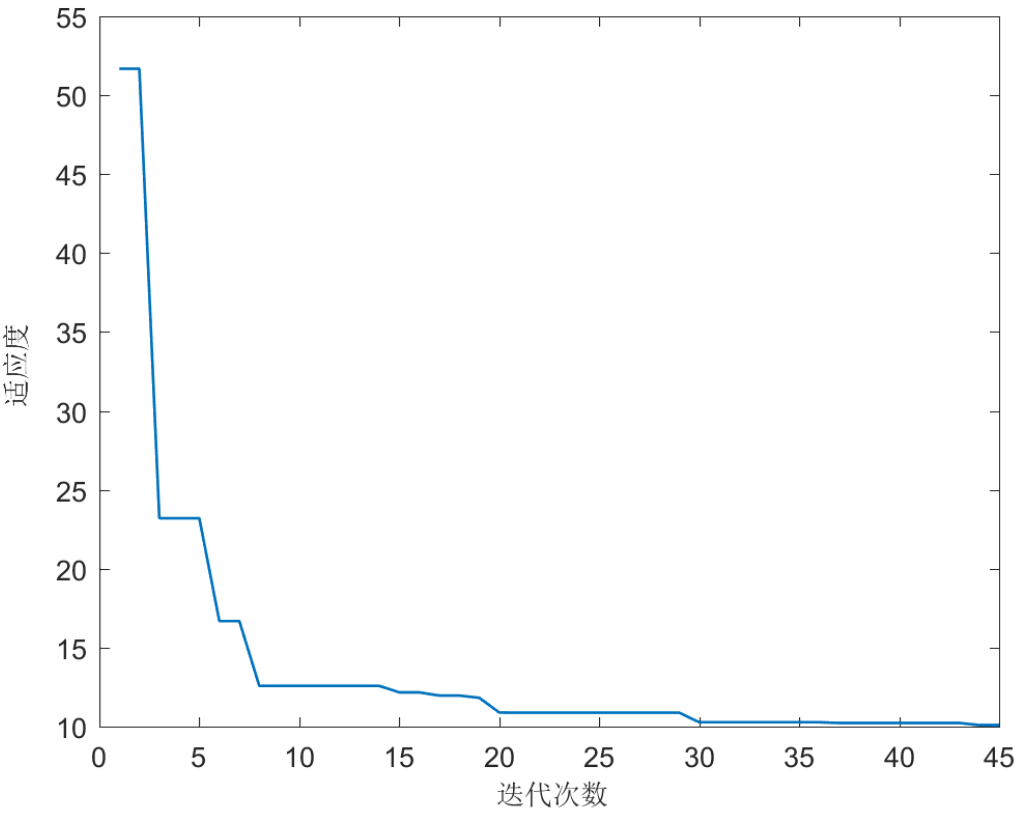


图 2. 11 BSO 算法的演化过程

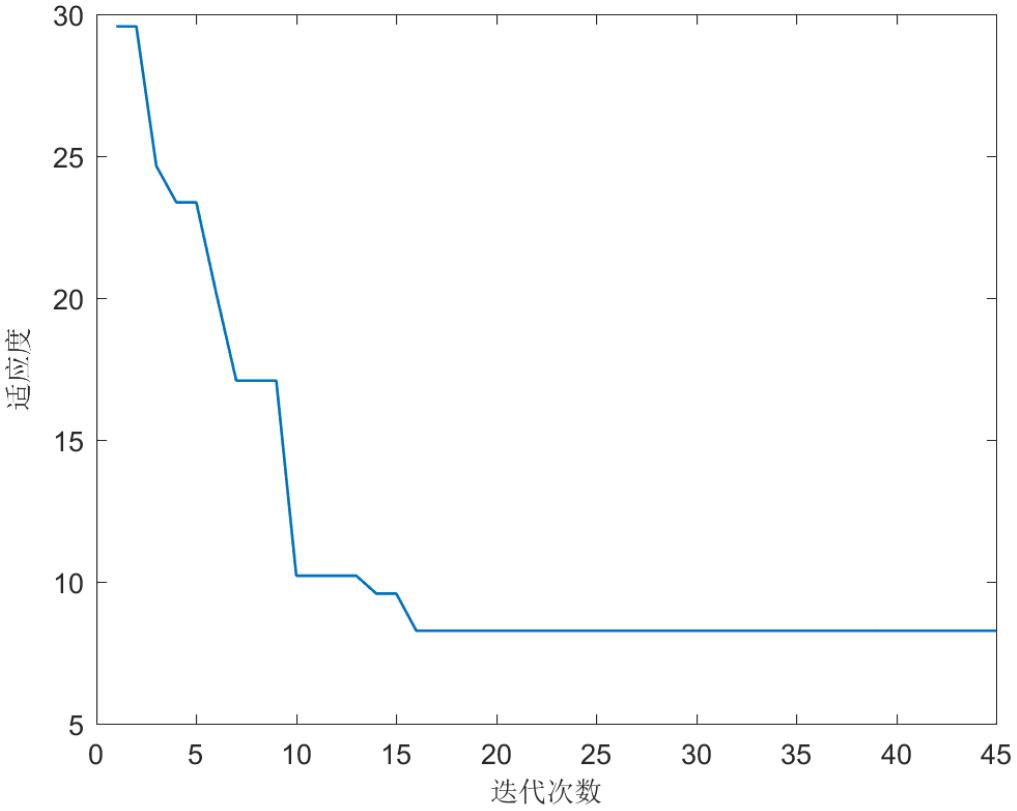


图 2.12　IBSO 算法的演化过程

表 2.1　不同解耦控制器下温室系统输出变量的最小追踪时间

| 控制器 / 追踪时间 $T_s(s)$ | PIDNN | BSO-PIDNN | IBSO-PIDNN |
|---|---|---|---|
| $y_1$ | 8.70 | 1.60 | 0. 90 |
| $y_2$ | 5.30 | 0.45 | 0.35 |

从图 2.8～2.12 和表 2.1 中可以看出：

（1）PIDNN、BSO-PIDNN 和 IBSO-PIDNN 都实现了温度和湿度的单独控制，能够进行有效的解耦控制，但本章所提出的 IBSO-PIDNN 显示了更好的控制指标。

（2）在图 2.8 和图 2.9 中， PIDNN 在温度和湿度控制上的误差都很大，不能完成理想的解耦控制，优化后的 PIDNN 控制器的性能得到了显著的改善，但 BSO-PIDNN 在湿度的控制中仍然存在不小的误差， IBSO-PIDNN 进一步缩小了误差，将温度和湿度的输出误差均缩小至合理范围，使其能更快、更准确地追踪到理想信号。

（3）图 2.10 以及表 2.1 的数据清楚地表明，PIDNN 很长时间无法将误差收敛至合理范围，而 IBSO-PIDNN 的收敛速度最快，且给温室系统提供了稳定的输入，在最短的时间内使输出误差达到稳定的最小状态，将温度和湿度的追踪时间都控制在了 1s 以内，且湿度的追踪时间比 BSO-PIDNN 快了近一倍，展现了 IBSO 更好的优化能力。

（4）图 2.11 和图 2.12 的结果表明，IBSO 算法具有更好的优化能力且收敛速度快，在第 16 次迭代时找到了最优值，而 BSO 算法需要 43 次迭代才获得最优值。

## 2.4 本章小结

本章描述了 PIDNN 的基本结构以及解耦原理，分析了 PIDNN 在解耦控制方面的优势以及存在的一些问题。针对 PIDNN 随机初值带来的问题，使用智能优化算法对其最优初始权值进行搜索，提出了一种改进的天牛群算法。改进的天牛群算法针对天牛群算法易陷入早熟收敛的问题在天牛种群的全局最优位置中加入了一个突变因子，提高天牛种群搜索的多样性，同时采取非线性递减的方式更新惯性权重，改善了天牛群的早熟收敛状况。然后进一步提出了基于 IBSO 算法的 PIDNN 控制器。最后采用温室系统进行了仿真实验，验证了所提出控制器的有效性，经过对仿真结果的分析，表明了所提出的控制器能有效地解耦变量，并且使被控变量能更加快速精确地收敛到目标信号。

# 第三章 基于分数阶学习算法的 PID 神经网络解耦控制

多变量、强耦合复杂系统的解耦控制一直是研究的热点问题。第二章中，采用了智能优化算法对 PIDNN 初始权值进行优化，改进后的 PIDNN 具有更好的解耦控制性能，但其在响应速度以及控制精度等控制指标上仍然具有提升的空间且不同智能优化算法对各种不同系统的优化效果也不同。因此，本章将分数阶学习算法引入 PIDNN，将其权值更新算法修改为分数阶学习算法，提出一种基于分数阶学习算法的 PIDNN（Fractional-order learning algorithm based PIDNN，FPIDNN），利用分数阶算法的长记忆特性提高 PIDNN 的解耦控制性能，同时使用麻雀搜索算法（Sparrow Search Algorithm，SSA）优化 FPIDNN 初始权值。

本章中，第一小节介绍了分数阶微积分的相关理论，推导了基于分数阶学习算法的 PIDNN 的权值更新算法。第二小节介绍了麻雀搜索算法，第三小节使用了 SSA 算法来对 FPIDNN 进行优化，描述了 SSA 算法的优化原理以及优化 FPIDNN 的步骤。第四小节对提出的基于麻雀优化算法的分数阶学习策略的 PIDNN 控制器进行了实例仿真，其结果表明了分数阶学习算法的有效性，提出的控制器可以获得更高的精确度和收敛速度。

## 3.1 基于分数阶学习算法的 PID 神经网络

### 3.1.1 分数阶微积分理论

分数阶微积分可以描述任意阶次的积分、导数，具有记忆和遗传特性，在一些具有历史依赖的系统中，可以对系统进行更加精确的描述和建模，且分数阶微积分考虑了变量历史时刻信息对变量当前时刻值的影响，因此，分数阶算法可以获得比整数阶算法更加精确的结果。分数阶微积分发展至今有三种使用最广泛的定义：Riemann-Liouville（R-L）、 Grünwald-Letnikov （G-L）和 Caputo 定义，其中 R-L 和 G-L 定义已被证明在工程应用中是等效的，且其更适用于离散系统[65]。

函数 $f(t)$ 的 $\alpha$ 阶导数的 G-L 定义为[65]：

$$\begin{aligned} D^{\alpha} f(t) &= \lim_{h_1 \to 0} \frac{1}{h_1^{\alpha}} \sum_{j=0}^{\infty} (-1)^j \binom{\alpha}{j} f(t - jh_1) \\ &\approx \frac{1}{h_1^{\alpha}} \sum_{j=0}^{[t/h_1]} (-1)^j \binom{\alpha}{j} f(t - jh_1) \end{aligned} \tag{3.1}$$

式中，$D$是一个分数阶微分算子，$h_1$为步长，$[\cdot]$表示取最近的整数，$\alpha$是分数阶次，$f(\cdot)$是关于时间$t$的函数，$\binom{\alpha}{j}$是牛顿二项式，其定义为：

$$\binom{\alpha}{j}=\frac{\Gamma(\alpha+1)}{\Gamma(j+1)\Gamma(\alpha-j+1)} \tag{3.2}$$

式中，$\Gamma(\cdot)$代表伽玛函数，其定义为：

$$\Gamma(\alpha)=\int_0^{\infty}e^{-\tau}\tau^{\alpha-1}\mathrm{d}\tau \tag{3.3}$$

$(-1)^j\binom{\alpha}{j}$为二项式系数，代表了过去不同时刻信息对未来时刻信息的影响程度，离当前时刻越远，影响程度越低，是对分数阶遗忘特性的一个描述。对其做出如下定义：

$$c_j^{\alpha}=(-1)^j\binom{\alpha}{j} \tag{3.4}$$

其计算方式如下：

$$c_j^{\alpha}=\begin{cases}0 & j<0\\ 1 & j=0\\ (-1)^j\dfrac{\alpha(\alpha-1)\cdots(\alpha-j+1)}{j!} & j=1,2,...\end{cases} \tag{3.5}$$

式（3.5）使用递推公式可表示为：

$$c_j^{\alpha}=\begin{cases}1 & j=0\\ \left(1-\dfrac{\alpha+1}{j}\right)c_{j-1}^{\alpha} & j=1,2,...\end{cases} \tag{3.6}$$

由分数阶微积分的定义可以看出，分数阶微积分按照时间刻度量化了过去不同时刻信息对于当前计算的影响，更好的展现了函数的发展依赖过程，可以利用这些历史信息获得更加正确的计算方向和计算结果，这也是分数阶算法能够获得更高精度的一个原因。

### 3.1.2 分数阶学习算法

梯度下降法是最优化理论中求解无约束最优化问题的经典方法，在深度学习中有很广泛的应用。其原理为在迭代过程中，通过梯度的相反方向找到使得损失函数获得最小值的点，以此获得优化问题的解。

对于 $R^n$ 上的具有一阶连续偏导的损失函数 $J(\theta),\theta=(\theta_1,\theta_2,\cdots,\theta_n)$，用梯度下降法更新参数可表示为[81]：

$$\begin{aligned}\theta_{t+1}&=\theta_t-\eta\frac{\partial J(\theta)}{\partial\theta}\\ \triangle\theta&=\theta_{t+1}-\theta_t=-\eta\frac{\partial J(\theta)}{\partial\theta}\end{aligned}\tag{3.7}$$

式中，$\theta_t$ 为 $t$ 时刻的 $\theta$ 值，$\theta_{t+1}$ 为 $t+1$ 时刻 $\theta$ 的值，$\eta$ 为学习率。

分数阶学习算法是梯度下降法的分数阶形式，式（3.1）给出了 G-L 分数阶微积分算子的定义，根据该定义的数值解，可以得到如下分数阶学习算法[82]：

$$\triangle\theta=\theta_{t+1}-\theta_t=-\eta\frac{\partial J}{\partial\theta_t}h_1^{\alpha}-h_1^{\alpha}D^{\alpha}\theta(t)\tag{3.8}$$

FPIDNN 利用分数阶学习算法代替整数阶梯度下降法来更新网络的连接权值，通过分数阶微积分记忆特性更快地获得最优的网络模型，即通过分数阶算法获得各层的最佳权重值使得成本函数最小化。

FPIDNN 的成本函数可定义为[36]：

$$J=\sum_{p=1}^{n}E_p=\frac{1}{l}\sum_{p=1}^{n}\sum_{k=1}^{l}\left[r_p(k)-y_p(k)\right]^2\tag{3.9}$$

式中，$E_p$ 为理想信号与系统实际输出的偏差平方均值，$l$ 为采样点数，$n$ 为被控变量个数。

按照式（3.8）所示分数阶学习算法更新权值，则权值的分数阶更新修正可表示为：

$$\boldsymbol{\omega}(k+1)=\boldsymbol{\omega}(k)-\eta\frac{\partial J}{\partial\boldsymbol{\omega}(k)}h_1^{\alpha}-h_1^{\alpha}D^{\alpha}\boldsymbol{\omega}(k)\tag{3.10}$$

式中，$\boldsymbol{\omega}$ 代表输入层到隐含层权重 $\omega_{sih}$ 和隐含层到输出层权重 $\omega_{sho}$。

根据式（3.10），FPIDNN 各层权重的更新公式总结如下：

（1） 隐含层到输出层的连接权重

由式（3.10）可以得到：

$$\omega_{sho}(k+1)=\omega_{sho}(k)-\eta_1\frac{\partial J}{\partial\omega_{sho}(k)}h_1^{\alpha}-h_1^{\alpha}D^{\alpha}\omega_{sho}(k)\tag{3.11}$$

由式（3.1）可得：

$$D^{\alpha}\omega_{sho}(k)=\frac{1}{h_1^{\alpha}}\sum_{j=0}^{[k/h_1]}(-1)^j\binom{\alpha}{j}\omega_{sho}(k-jh_1)\tag{3.12}$$

由式（3.4），式（3.12）可以简化为：

$$D^{\alpha}\omega_{sho}(k)=\frac{1}{h_1^{\alpha}}\sum_{j=0}^{[k/h_1]}c_j^{\alpha}\omega_{sho}(k-jh_1) \tag{3.13}$$

将式（3.13）带入式（3.11），可得以下隐含层至输出层权重更新公式：

$$\omega_{sho}(k+1)=\omega_{sho}(k)-\eta_1\frac{\partial J}{\partial\omega_{sho}(k)}h_1^{\alpha}-\sum_{j=0}^{[k/h_1]}c_j^{\alpha}\omega_{sho}(k-jh_1) \tag{3.14}$$

式中，$\eta_1$为隐含层到输出层的学习率，且

$$\frac{\partial J}{\partial\omega_{sho}(k)}=-\sum_{h=1}^{3}\varphi(k)x_{sh}^{'}(k) \tag{3.15}$$

（2） 输入层到隐含层的连接权重

同理，根据式（3.10），输入层至隐含层权值更新公式为：

$$\omega_{sih}(k+1)=\omega_{sih}(k)-\eta_2\frac{\partial J}{\partial\omega_{sih}(k)}h_1^{\alpha}-\sum_{j=0}^{[k/h_1]}c_j^{\alpha}\omega_{sih}(k-jh_1) \tag{3.16}$$

式中，$\eta_2$为输入层到隐含层的学习率，且

$$\frac{\partial J}{\partial\omega_{sih}(k)}=-\sum_{i=1}^{2}\sum_{h=1}^{3}\omega_{sho}(k)\operatorname{sgn}\frac{v_{sh}^{'}(k)-v_{sh}^{'}(k-1)}{N_{sh}^{'}(k)-N_{sh}^{'}(k-1)}x_{si}(k)\varphi(k) \tag{3.17}$$

分数阶学习算法考虑了神经网络权值从第一次迭代的值到现在时刻整个过去时间段的权值信息的影响来对此刻权值进行精确计算，使得 FPIDNN 在训练修正过程中可以更快地获得更加精确的结果，从而实现快速地高精度跟踪响应。

## 3.2 麻雀搜索算法

麻雀搜索算法是由薛建凯于 2020 年提出的一种群体智能优化算法[83]。SSA 算法模拟了麻雀的觅食行为以及反捕食行为来解决全局优化问题，具有收敛速度快、鲁棒性强、全局搜索能力强等优点，引起了广泛关注并获得了很多应用[84-85]。使用麻雀搜索算法对 FPIDNN 的初始权值进行最优搜索可以加快收敛，避免局部最优，使其获得更好的解耦控制效果。

在 SSA 算法中，麻雀可以分为发现者、加入者和侦察者三种类型。生产者主要承担的任务是为种群中其他成员提供觅食的方向，因此，发现者具有很大的搜素范围以及较高的适应度值。加入者通过跟随发现者来进行食物的获取，但在搜索范围内加入者的适应度值高于发现者，它的身份就可以转换为发现者。侦察者在种群中起到侦察周围环境安全的作用，一旦侦察者发觉危险，就会提醒种群中其他成员逃离危险区域，发生反捕食行为。

根据麻雀觅食原理，麻雀种群中各类麻雀在 $q$ 维空间中位置更新总结如下[83]：

发现者的位置可按式（3.18）进行更新：

$$P_{iq}^{t+1}=\begin{cases}P_{iq}^{t}\cdot\exp\left(\dfrac{-i}{a_1\cdot T}\right) & if\ R_2<ST\\ P_{iq}^{t}+a_2\cdot\boldsymbol{Q} & if\ R_2\geq ST\end{cases} \tag{3.18}$$

式中，$P_{iq}^{t}$ 是第 $i$ 个麻雀在第 $t$ 次迭代、第 $q$ 维的位置信息，$a_1\in(0,1]$ 为随机数，$a_2$ 是服从正态分布的随机数，$T$ 代表最大迭代次数，$\boldsymbol{Q}$ 是每个元素均为 1 的矩阵，$R_2\in[0,1]$ 代表预警值，$ST\in[0.5,1]$ 代表安全临界值，如果 $R_2<ST$，则表示周围环境是安全的，发现者可以进行搜索行为，否则代表周围觅食环境可能存在危险，已有成员发出报警，种群中所有麻雀需移动到安全区域觅食。

加入者的位置可按式（3.19）进行更新：

$$P_{iq}^{t+1}=\begin{cases}a_2\cdot\exp\left(\dfrac{P_{worst}^{t}-P_{iq}^{t}}{i^2}\right) & if\ i>\rho/2\\ P_{B}^{t+1}+\left|P_{iq}^{t}-P_{B}^{t+1}\right|\cdot\boldsymbol{Q}\cdot\left(\boldsymbol{C}^{\mathrm{T}}\left(\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}}\right)^{-1}\right) & otherwise\end{cases} \tag{3.19}$$

式中，$P_{B}^{t+1}$ 表示由生产者在第 $t+1$ 次迭代搜索到的最佳位置，$P_{worst}^{t}$ 表示第 $t$ 次迭代的全局最差位置，$\boldsymbol{C}$ 表示一个矩阵，矩阵中每个元素被随机分配为 1 或 –1，$\rho$ 为种群中麻雀数量。$i>\rho/2$ 说明第 $i$ 个加入者可能未能寻到食物，需要飞往其他区域搜寻食物。

侦察者的位置可按式（3.20）进行更新：

$$P_{iq}^{t+1}=\begin{cases}P_{best}^{t}+a_3\cdot\left|P_{iq}^{t}-P_{best}^{t}\right| & if\ f_i>f_g\\ P_{iq}^{t}+a_4\cdot\left(\dfrac{\left|P_{iq}^{t}-P_{worst}^{t}\right|}{(f_i-f_w)+\iota}\right) & if\ f_i=f_g\end{cases} \tag{3.20}$$

式中，$a_3$ 表示步长，为遵从正态分布的随机数，$P_{best}^{t}$ 是第 $t$ 次迭代的全局最佳位置，$a_4\in[-1,1]$ 是一个随机数，$f_i$ 表示当前麻雀的适应度值，$f_g$ 表示第 $t$ 次迭代的全局最佳适应度值，$f_w$ 表示第 $t$ 次迭代的全局最差适应度值，$\iota$ 表示一个非常小的常数，可以避免分母为零的情况出现。$P_{best}$ 周围都是安全的，当 $f_i=f_g$ 时，麻雀会意识到危险，向种群靠拢。

SSA 算法的流程如图 3.1 所示。

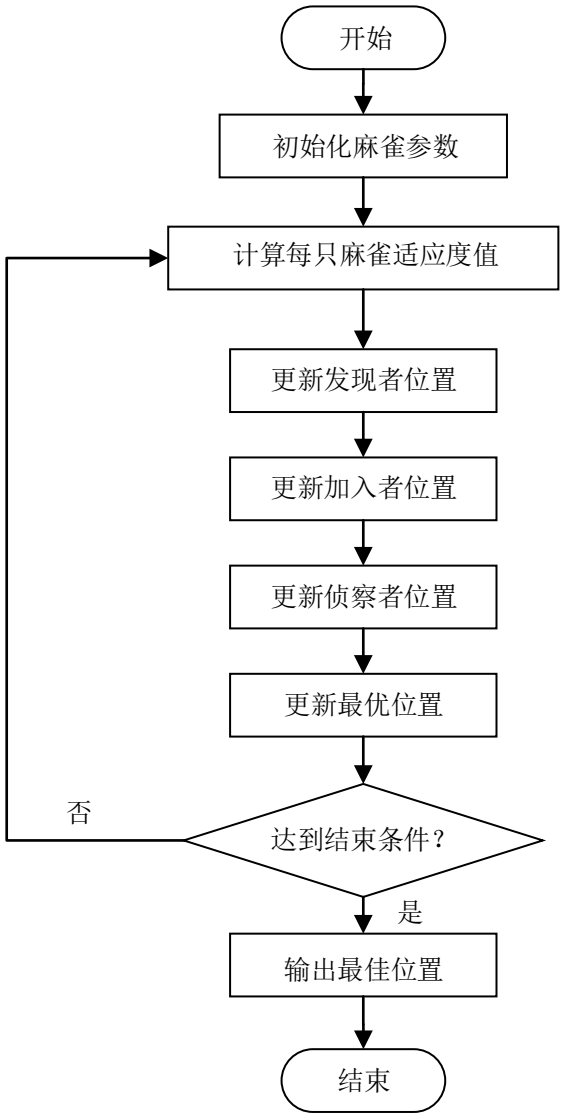


图 3.1 SSA 算法流程图

SSA 算法是一种群体优化算法，麻雀种群中的麻雀分为三个角色，各司其职，发现者具有带领作用，决定着飞行方向，加入者跟随发现者来获取食物，当加入者的适应度值达到发现者标准，二者身份即可互换，侦察者负责侦察环境是否存在危险，必要时对种群成员发出报警。正是由于麻雀种群这种严谨的觅食机制，使得 SSA 算法在复杂优化问题中具有良好的性能，因此，SSA 算法能够很好的应用于 FPIDNN 初始权值的优化。

## 3.3 基于麻雀优化算法的分数阶学习策略

本小节采用基于 SSA 算法的分数阶学习策略优化 PIDNN，提出了基于 SSA 算法的 FPIDNN（SSA-FPIDNN）。在 SSA 算法中，种群中每只麻雀的位置都代表了被优化问题的一个候选解，根据每个位置的适应度值，选择最优的位置即为优化问题的最优解。使用 SSA 算法优化 FPIDNN 的初始权值，则 SSA 算法最终搜索出的全局最佳位置就是 FPIDNN 的最优初

始权值。如图 3.2 所示，SSA-FPIDNN 使用 SSA 算法获取 FPIDNN 的最优初始权值，然后利用分数阶学习算法进行权值的更新修正，$\omega_{initial}$ 是 SSA 算法确定的最优初始权值，$\omega(k+1)$ 是分数阶学习算法确定的下一时刻权值。

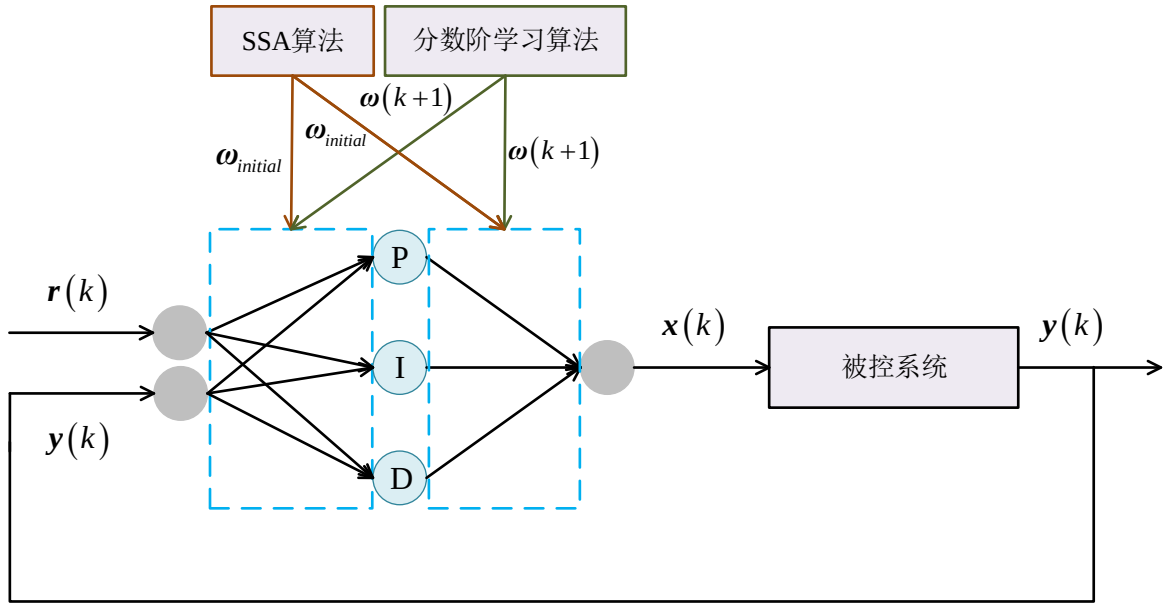


图 3.2　SSA-FPDNN 解耦控制图

基于麻雀优化算法的分数阶学习策略可以使 PIDNN 在学习的初始阶段就以正确的起始点和方向进行收敛过程，减少了反复训练学习的过程，加快了收敛速度，并在学习过程中借助历史权值信息更加精准的获得下一时刻权值，可以更快的使神经网络成本函数尽可能达到理想化值，从而获得更加理想的解耦控制效果。

图 3.3 描述了 SSA-FPIDNN 控制器的完整解耦控制方案，其中 $e(k)=r(k)-y(k)$ 是 FPIDNN 的输入，$x(k)$ 是被控系统的输入也是 FPIDNN 的输出。

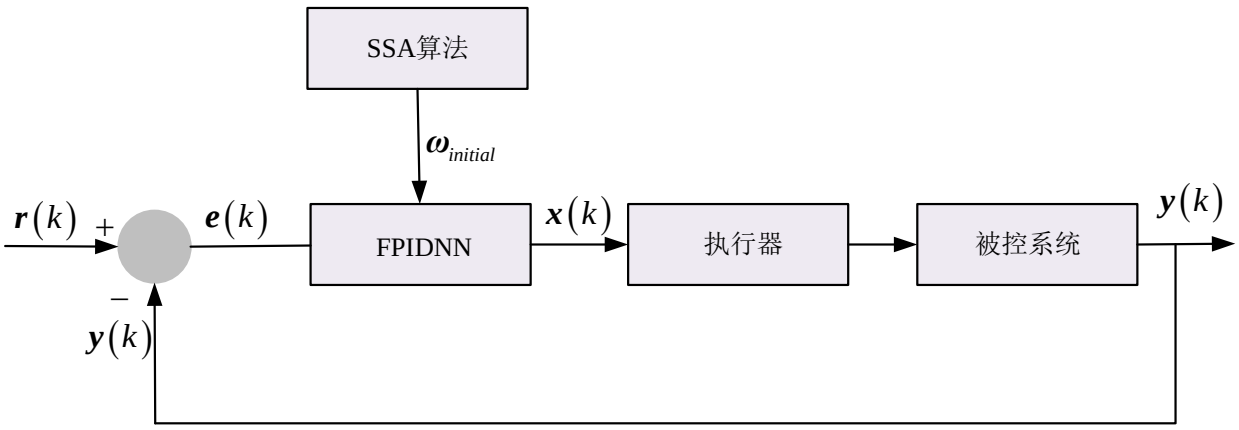


图 3.3　SSA-FPIDNN 解耦控制方案图

此闭环控制方案中，首先，由 SSA 算法搜索 FPIDNN 的最优初始权值 $\omega_{initial}$，输入给 FPIDNN，然后，FPIDNN 以 $\omega_{initial}$ 为初始权值，根据理想信号 $r$ 和被控系统的实际输出 $y$ 之

间的差值 $\boldsymbol{e}$ 不断地训练学习，更新权值，使 $\boldsymbol{r}$ 与 $\boldsymbol{y}$ 之间的差值不断趋近于零。FPIDNN 训练完成后得到的输出 $\boldsymbol{x}$ 会作为执行器的输入，此输入经 SSA-FPIDNN 训练后保证了可以使输出 $\boldsymbol{y}$ 达到理想信号 $\boldsymbol{r}$，执行器完成执行动作后，被控系统输出即可到达理想信号并保持稳定，完成了解耦控制。

定义如下适应度函数：

$$f(t)=\sum_{t=1}^{T}e^{0.01\cdot t}\left(\sum_{k=1}^{length}\sum_{p=1}^{n}\left|r_p(k)-y_p(k)\right|\right) \tag{3.21}$$

式中，$T$ 为 SSA 算法的最大迭代次数，$length$ 为 FPIDNN 训练次数。

SSA-FPIDNN 的具体解耦控制流程由算法 3.1 描述，其中 $PD$ 和 $SD$ 分别是麻雀种群中发现者和侦察者的数量：

算法 3.1： SSA-FPIDNN 的解耦控制算法

输入：$\boldsymbol{r},\boldsymbol{y}$
输出：$\boldsymbol{x}_o^{''}$

1：初始化 SSA 算法参数，给定 FPIDNN 的训练次数
2：IBSO 算法迭代次数：$T$
3：PIDNN 训练次数：$length$
4：*while* $(t<T)$ *do*
5：　计算并比较每个位置的适应度值，更新当前的最优和最差位置
6：　*for* $i=1:PD$ *do*
7：　　通过式（3.18）更新发现者位置
8：　*end for*
9：　*for* $i=(PD+1):\rho$ *do*
10：　　通过式（3.19）更新加入者位置
11：　*end for*
12：　*for* $i=1:SD$ *do*
13：　　通过式（3.20）更新侦察者位置
14：　*end for*
15：　计算并比较当前位置与之前位置，如果优于之前，更新位置
16：　$t=t+1$
17：*end while*
18：输出当前最优解
19：FPIDNN 以 SSA 算法输出的最优解作为初始权值
20：*for* $k=1:length$ *do*
21：　通过式（2.3）计算 $x_{si}(k)$
22：　通过式（3.16）的分数阶算法更新输入层到隐含层权值
23：　通过式（2.8）计算 $x_{sh}^{'}(k)$
24：　通过式（3.11）的分数阶算法更新隐含层到输出层权值

25：　通过式（2.11）计算 $x_o^{''}(k)$

26：*end for*

27：*end*

SSA-FPIDNN 采用 SSA 算法克服了随机初始权重引起的不利影响，并通过分数阶学习算法对权值进行更新，由于分数阶学习算法具有长记忆功能，考虑了过去权值对当前计算的影响，可以使神经网络在训练过程中得到更快更精确的结果，两种算法的引入提高了控制器的跟踪响应速度和控制精度。

## 3.4 仿真实例

**例 3.1**：考虑一个如下的 3 输入 3 输出的非线性耦合系统[86]：

$$\begin{cases} y_1(k)=0.4y_1(k-1)+x_1(k-1)/\left[1+x_1^{\,2}(k-1)\right]+0.2x_1^{\,3}(k-1) \\ \qquad +0.5x_2(k-1)+0.3y_2(k-1) \\ y_2(k)=0.2y_2(k-1)+x_2(k-1)/\left[1+x_2^{\,2}(k-1)\right]+0.4x_2^{\,3}(k-1) \\ \qquad +0.2x_1(k-1)+0.3y_3(k-1) \\ y_3(k)=0.3y_3(k-1)+x_3(k-1)/\left[1+x_3^{\,2}(k-1)\right]+0.4x_3^{\,3}(k-1) \\ \qquad +0.4x_2(k-1)+0.3y_1(k-1) \end{cases} \tag{3.22}$$

其中 $\boldsymbol{y}=[y_1,y_2,y_3]^{\mathrm{T}}$ 和 $\boldsymbol{x}=[x_1,x_2,x_3]^{\mathrm{T}}$ 分别是系统的输出向量和输入向量。目标信号设置为：$\boldsymbol{r}=[0.7,0.4,0.6]^{\mathrm{T}}$。

针对此多变量、强耦合数值系统，使用 PIDNN、基于 SSA 算法的 PIDNN（SSA-PIDNN）和 SSA-FPIDNN 控制器对其进行解耦控制，并对比其仿真结果。学习率设置为 $\eta_1=0.03$，$\eta_2=0.006$，分数阶次设置为 $\alpha=0.998$，训练次数设置为 200，麻雀个体设置为 50 只。SSA 算法参数设置为：$PD=10$，$SD=10$，$ST=0.7$，$T=200$。

图 3.4 和图 3.5 分别显示了数值系统在 PIDNN、SSA-PIDNN 和 SSA-FPIDNN 三种解耦控制器下的输出响应和输出误差收敛曲线，图 3.6 显示了 SSA-FPIDNN 的输出。表 3.1 列出了数值系统的输出变量在三种不同解耦控制器下的最小跟踪时间，$T_s$ 代表输出变量达到理想信号所需的最短时间。

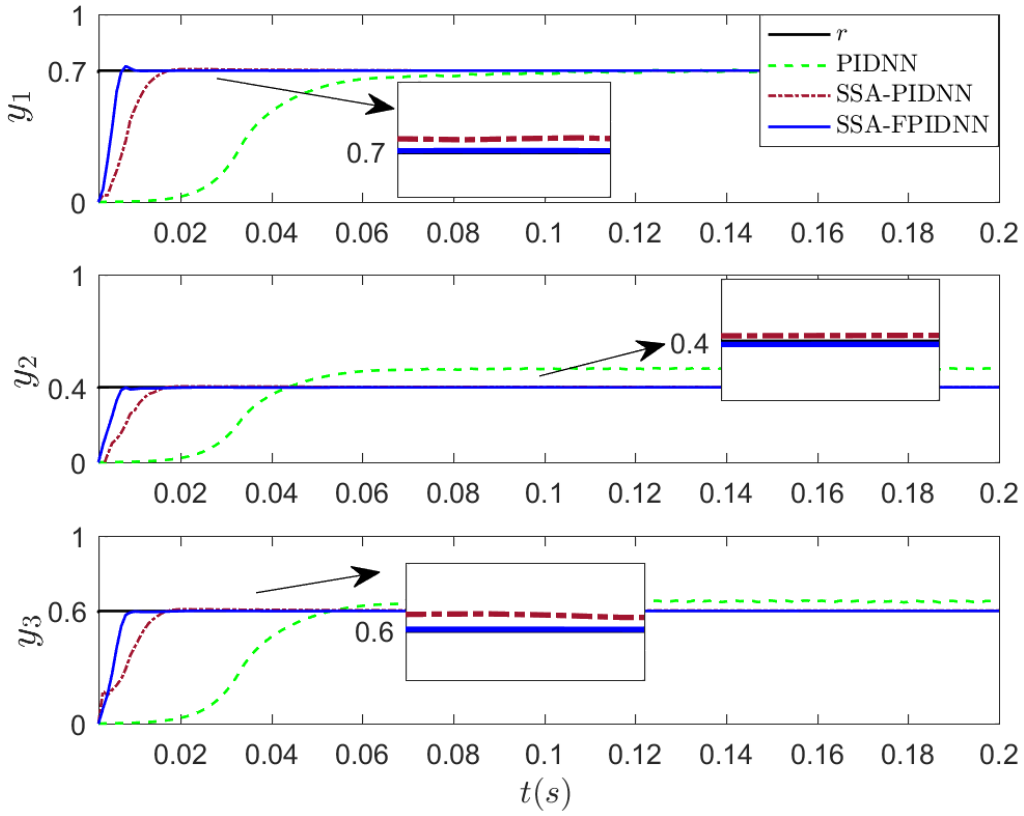


图 3.4　不同解耦控制器下数值系统的输出响应

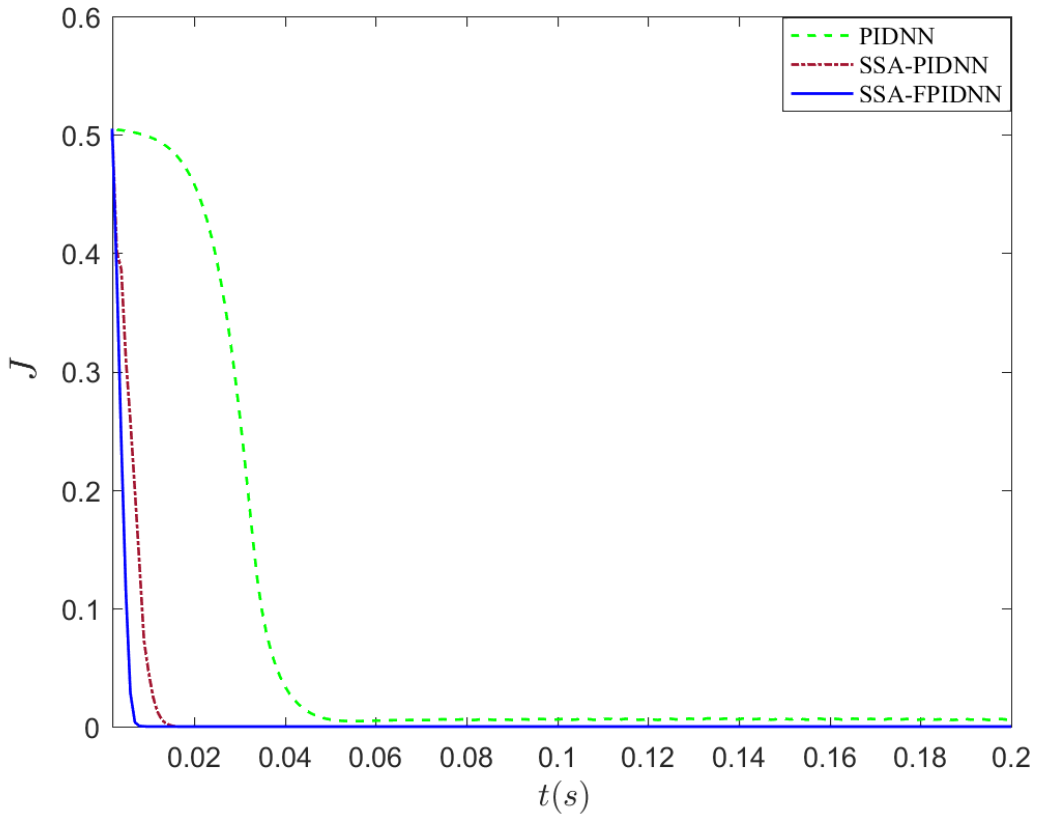


图 3.5　不同解耦控制器下数值系统的输出误差收敛曲线

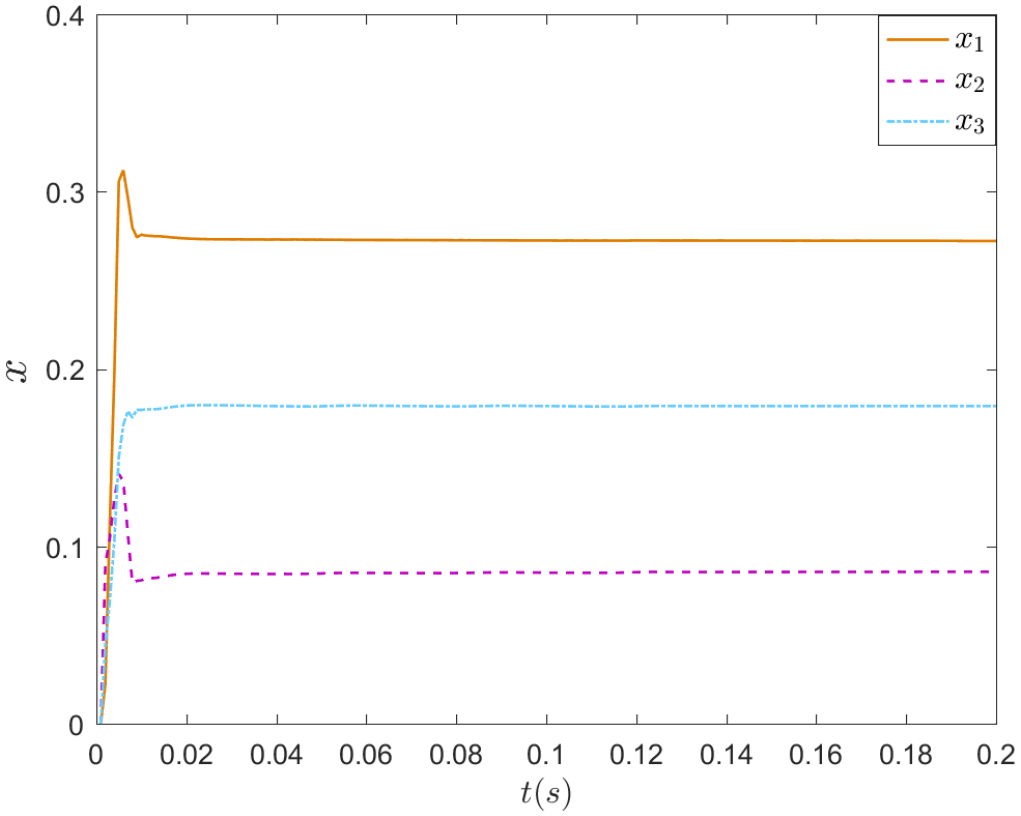


图 3.6　SSA-FPIDNN 的输出

表 3.1　不同解耦控制器下数值系统输出变量的最小追踪时间

| 控制器 / 追踪时间 $T_s(s)$ | PIDNN | SSA-PIDNN | SSA-FPIDNN |
|---|---|---|---|
| $y_1$ | 0.097 | 0.017 | 0.010 |
| $y_2$ | 0.044 | 0.016 | 0.013 |
| $y_3$ | 0.053 | 0.016 | 0.009 |

由图 3.4 ~ 3.6 和表 3.1 可以看出，

（1）PIDNN、SSA-PIDNN 和 SSA-FPIDNN 都能够实现耦合变量的单独控制，但 PIDNN 在 $y_2$ 和 $y_3$ 的控制上具有较大的误差，SSA-PIDNN 在三个输出变量中具有微小误差，而 SSA-FPIDNN 具有非常高的精确度，三个输出变量均快速精准地跟踪上了理想信号。

（2）在图 3.4 和图 3.5 中，PIDNN 需要较长时间才能追踪到理想信号，而改进后的 PIDNN 无论是在收敛速度上还是精确度上都有明显的优势，表明 SSA 算法具有良好的优化能力，虽然 SSA-FPIDNN 在 $y_1$ 的控制上具有少量的超调，但其控制精度优于 SSA-PIDNN，验证了分数阶学习算法的有效性。

（3）图 3.6 和表 3.1 都表明了 SSA-FPIDNN 具有更好的性能，可以给数值系统提供稳定的输入，且其在三个输出变量的追踪速度都更优于 SSA-PIDNN，使 $y_1$ 和 $y_3$ 减少了近一倍的

追踪时间，大大改善了控制性能。

**例 3.2**：考虑如 2.3 节中的温室系统仿真实例，即如下的简化三输入二输出模型[80]：

$$\begin{cases} y_1(k)=0.897y_1(k-1)+0.202x_1(k)-0.371x_2(k)+0.250x_3(k) \\ y_2(k)=0.400y_2(k-1)+0.250x_1(k)+0.500x_2(k)-0.250x_3(k)[y_2(k-1)-0.450] \end{cases} \tag{3.23}$$

其中，$\boldsymbol{y}=[y_1,y_2]^{\mathrm{T}}$ 和 $\boldsymbol{x}=[x_1,x_2,x_3]^{\mathrm{T}}$ 分别是系统输出和输入向量。目标信号设置为：$\boldsymbol{r}=[0.5,0.7]^{\mathrm{T}}$。

针对此多变量、强耦合系统，使用 PIDNN、SSA-PIDNN 和 SSA-FPIDNN 控制器对其进行解耦控制，并对比其仿真结果。学习率设置为 $\eta_1=0.006$，$\eta_2=0.002$，分数阶次设置为 $\alpha$ = 0.9899，训练次数设置为 200，麻雀个体设置为 50 只。SSA 算法参数设置为：$PD=10$，$SD=10$，$ST=0.7$，$T=200$。

图 3.7 和图 3.8 分别显示了温室系统在 PIDNN、SSA-PIDNN 和 SSA-FPIDNN 三种解耦控制器下的输出响应和输出误差收敛曲线，图 3.9 显示了 SSA-FPIDNN 的输出。

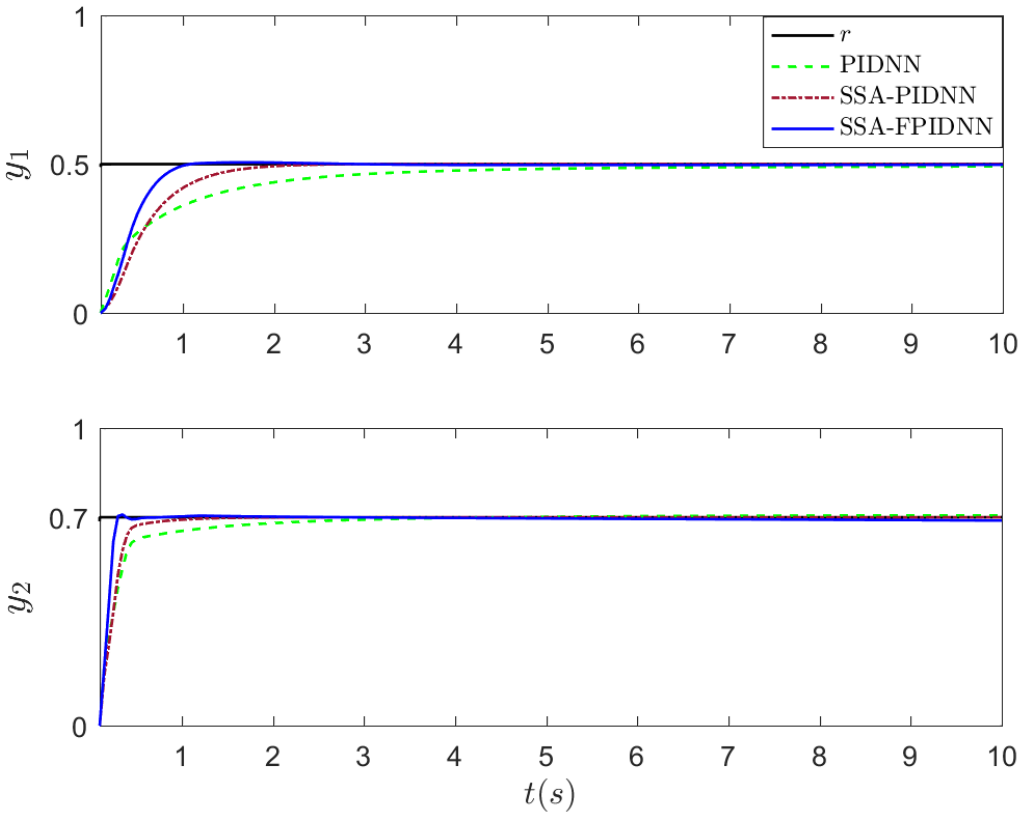


图 3.7　不同解耦控制器下温室系统的输出响应

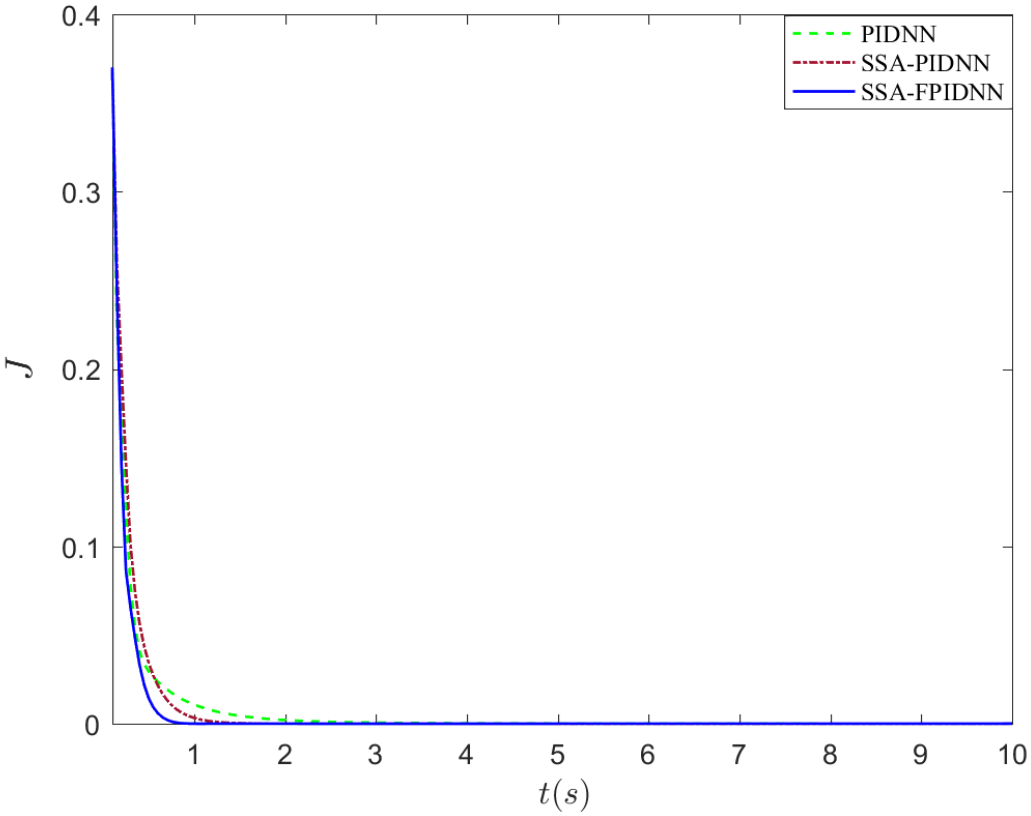


图 3.8　不同解耦控制器下温室系统的输出误差收敛曲线

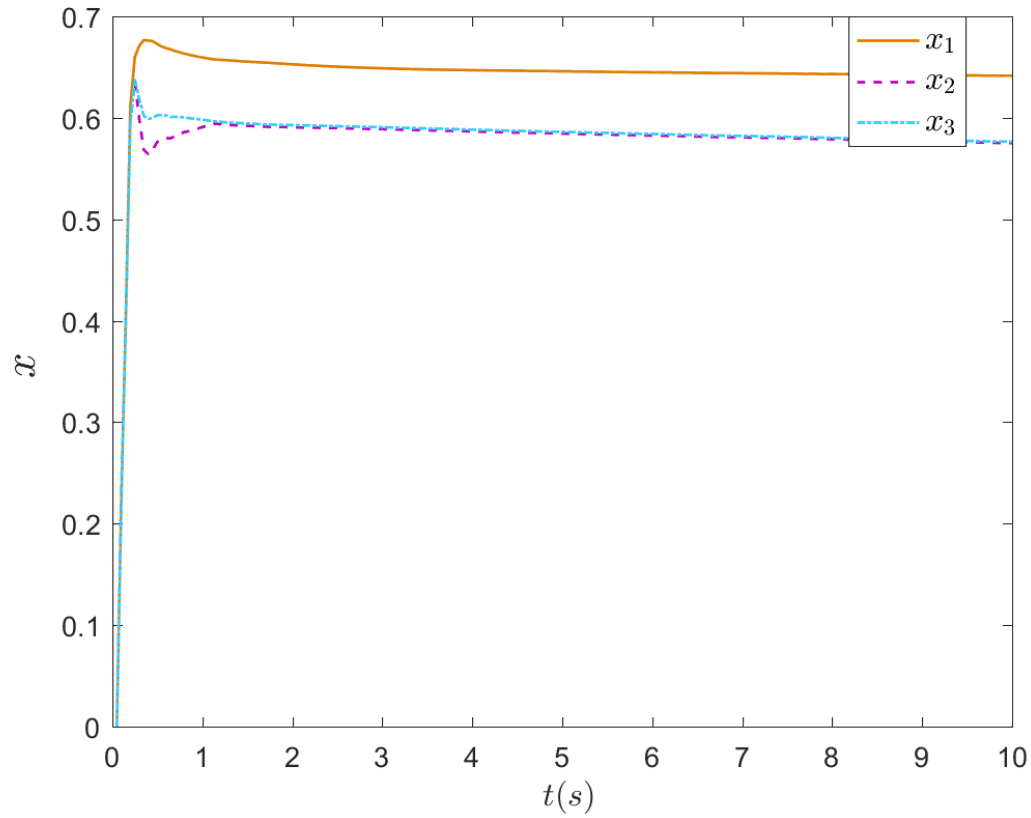


图 3.9　SSA-FPIDNN 的输出

为更好的验证本章所提出的解耦控制器的有效性，引用文献[80]中的基于粒子群算法的

PIDNN（PSO-PIDNN）的结果进行加强对比，各类不同控制器下系统输出变量的最小追踪时间如表 3.2 所示，$T_s$ 代表输出变量达到理想信号所需的最短时间。

表 3.2　不同解耦控制器下温室系统输出变量的最小追踪时间

| 控制器 / 追踪时间 $T_s(s)$ | PIDNN | SSA-PIDNN | PSO-PIDNN | SSA-FPIDNN |
|---|---|---|---|---|
| $y_1$ | 5.85 | 1.95 | 2.20 | 1.05 |
| $y_2$ | 2.50 | 0.90 | 1.70 | 0.40 |

从图 3.7～3.9 和表 3.2 的结果可以看出：

（1）PIDNN、SSA-PIDNN 和 SSA-FPIDNN 都展现了有效的解耦控制能力，但 PIDNN 在温度和湿度的控制上仍然存在少量误差，且在湿度控制后期误差变大，SSA-PIDNN 和 SSA-FPIDNN 都展现了良好的精确度。

（2）在图 3.7 和图 3.8 中，SSA-FPIDNN 结合了 SSA 算法和分数阶学习算法的优点，达到了快速、高精度的解耦控制效果，与经典的 PIDNN 控制器以及 SSA-PIDNN 控制器相比，虽然在湿度的控制上出现了微量的超调，但可以在稳态误差近乎为零的情况下，使温度和湿度都能迅速精确地跟踪到理想信号，表明其更适用于实际系统。

（3）表 3.2 中的数据以及图 3.9，展示了 SSA-FPIDNN 跟踪时间最短，动态控制指标更加优越，给温室系统提供了稳定的输入，使其温度和湿度的收敛时间均在 1s 以内，相较于 SSA-PIDNN 温度收敛时间减少近 1 倍，湿度收敛时间减少近 3 倍，表明了分数阶算法的优越性，也验证了 SSA 算法适用于优化 PIDNN 初始权值，可以获得比粒子群算法更快的收敛速度。

（4）本节中两个系统都属于多变量、强耦合系统，且两个系统的输入和输出个数不相同，但 SSA-FPIDNN 均实现了良好地解耦控制，表明了其对多变量系统的适用性。

## 3.5 本章小结

本章介绍了分数阶学习算法，并将其引入 PIDNN 各层连接权值的更新算法中，推导了分数阶的 PIDNN 权值修正算法，提出了一种基于分数阶学习算法的 PIDNN。随后介绍了麻雀搜索算法的优化原理，并使用该算法优化 FPIDNN 进行初始权值，解决了随机初始权重带来的困难，给出了基于 SSA 算法的 FPIDNN 的解耦控制流程。最后通过仿真实验结果验证了提出的 SSA-FPIDNN 可以获得更好的精确度和响应速度，对传统 PIDNN 实行了良好的改进，为多变量、强耦合系统的解耦控制提供了一种可靠的方法。

# 第四章 分数阶 PID 神经网络解耦控制

第二章中通过智能优化算法搜索 PIDNN 最优初始权值一定程度上改善了权值收敛速度慢，易陷入局部最优的问题，第三章中提出的 SSA-FPIDNN 通过分数阶学习算法进一步提高了控制的响应速度和精确度，但其存在控制中易出现少量超调以及对神经网络参数选取不当控制后期易发散的问题。因此，本章提出了基于分数阶神经元的分数阶 PID 控制器（$PI^{\lambda}D^{\mu}$ neural network，$PI^{\lambda}D^{\mu}NN$），将分数阶 PID 算法引入 PIDNN 的隐含层神经元，定义了分数阶神经元，分数阶 PID 算法的记忆性和稳定性可以使 $PI^{\lambda}D^{\mu}NN$ 精确更新当前权值，提高控制精度、收敛速度和稳定性，减少超调，同时使用 BSO 算法进行初值优化，获得了优越的解耦控制效果。

本章中，第一小节由分数阶 PID 算法详细推导了分数阶的 P、$I^{\lambda}$、$D^{\mu}$ 神经元的状态转换函数，描述了 $PI^{\lambda}D^{\mu}NN$ 控制器结构。第二小节分析了 PIDNN 的稳定性，根据李雅普诺夫稳定性定理，推导出一个有关于学习率的稳定性充分条件。第三小节利用天牛群算法获得 $PI^{\lambda}D^{\mu}NN$ 的最优初始权值，并给出了基于天牛群算法的 $PI^{\lambda}D^{\mu}NN$ 解耦控制的步骤。第四小节对提出的基于天牛群算法的 $PI^{\lambda}D^{\mu}NN$ 控制器进行了实例仿真，其结果表明了所提出控制器在解耦控制方面的有效性以及在控制的收敛速度、控制精度和稳定性等性能上的优越性。

## 4.1 分数阶神经元

（1） $PI^{\lambda}D^{\mu}$ 神经元

分数阶 PID 算法相对于整数阶 PID 算法，其微分项和积分项中分别多出一个可以调节的参数，算法更加灵活，可以根据不同系统选择合理的参数，提供了更多控制方案。因此，分数阶 PID 算法相对于整数阶 PID 算法在一些高阶、非线性复杂系统中的控制性能更好，控制效果也更加稳定。

分数阶 PID 控制器传递函数定义如下[66]：

$$G(s)=\frac{U(s)}{E(s)}=k_P+k_I s^{-\lambda}+k_D s^{\mu} \quad (\lambda,\mu>0) \tag{4.1}$$

式中，$G(s)$为控制器的传递函数，$E(s)$为误差，$U(s)$为控制器输出，$\lambda$ 和 $\mu$ 分别表示积分和微分调节器的分数阶阶次，$k_P$、$k_I$、$k_D$ 分别为比例、积分和微分参数。而离散的分数阶 PID 控制器即 $PI^{\lambda}D^{\mu}$ 控制器可以由分数阶微积分的 Grünwald-Letnikov 定义描述为[87]：

$$u(k)=\left(k_P+k_I\sum\nolimits_k^{\lambda}+k_D\Delta_k^{\mu}\right)e(k) \tag{4.2}$$

式中， $e(k)$ 为 $k$ 时刻控制器的输入， $u(k)$ 为 $k$ 时刻控制器的输出， $\sum_k^{\lambda}$ 和 $\Delta_k^{\mu}$ 分别代表离散时间内的分数阶的积分和微分调节器。

离散的 PI$^{\lambda}$D$^{\mu}$ 控制器结构图如图 4.1 所示，控制器输入经过分数阶积分器和微分器的处理后得到控制器的输出。

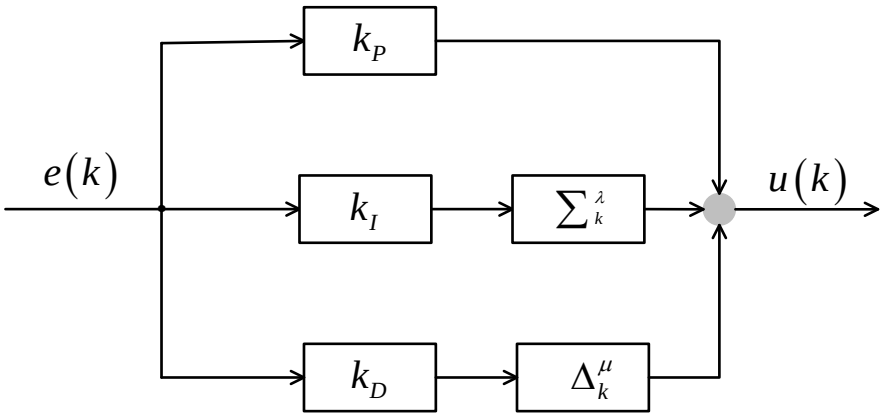


图 4.1 离散的 PI$^{\lambda}$D$^{\mu}$ 控制器结构图

式（4.2）中的 $\sum_k^{\lambda}e(k)$ 和 $\Delta_k^{\mu}e(k)$ 具体的表达式如下：

$$\begin{aligned}\sum\nolimits_k^{\lambda}e(k)&=\Delta_k^{-\lambda}e(k)=e(k)+\sum_{j=0}^{k-1}c_j^{-\lambda}e(k-j)\\ \Delta_k^{\mu}e(k)&=e(k)+\sum_{j=0}^{k-1}c_j^{\mu}e(k-j)\end{aligned} \tag{4.3}$$

式中， $\lambda>0,\mu\le 1$。式中 $c_j^{\varepsilon}=(-1)^j\binom{\varepsilon}{j}(\varepsilon=\lambda,\mu)$ 为遗忘因子，当 $j=1,2,\cdots$ 时其递推式为：

$$\begin{aligned}c_j^{\varepsilon}&=(-1)^j\frac{\varepsilon(\varepsilon-1)\cdots(\varepsilon-j+1)}{j!}\\&=(-1)^{j-1}\frac{\varepsilon(\varepsilon-1)\cdots(\varepsilon-j+2)}{(j-1)!}(-1)\frac{(\varepsilon-j+1)}{j}\\&=c_{j-1}^{\varepsilon}(-1)\frac{(\varepsilon-j+1)}{j}\\&=c_{j-1}^{\varepsilon}\left(1-\frac{\varepsilon+1}{j}\right)\end{aligned} \tag{4.4}$$

故遗忘因子更新式为：

$$c_j^{\varepsilon}=\begin{cases}1, & j=0\\ \left(1-\dfrac{\varepsilon+1}{j}\right)c_{j-1}^{\varepsilon}, & j=1,2,\cdots\end{cases} \tag{4.5}$$

由遗忘因子的表达式可知，在离散的分数阶 PID 控制器中，过去时刻的误差会影响当前时刻的运算，且 $j$ 的值与 $\left|c_j^{\varepsilon}\right|$ 值成反比，表明随着过去时刻越往前，其信息对当前时刻的运算的影响越小，而遗忘因子很好的分配了过去各个时刻信息的权重。

由式（4.3）~（4.5）可以定义出分数阶比例（P），积分（$I^{\lambda}$）和微分（$D^{\mu}$）神经元，其输入、输出分别如下:

（a） P 神经元

比例控制器表达式为：

$$u_P(k)=k_P e(k) \tag{4.6}$$

假设每个 P 神经元都有 $n-1$ 个输入分支连接到它，由 $i$ 表示，那么在任意时刻 $k$，比例神经元的输入为：

$$N_P(k)=\sum_{i=1}^{n-1}\omega_{ihP}(k)x_{iP}(k) \tag{4.7}$$

式中，$x_{iP}(k)$ 是 $k$ 时刻连接到 P 神经元的第 $i$ 个神经元的输出，$\omega_{ihP}(k)$ 是 P 神经元与第 $i$ 个神经元 $k$ 时刻的连接权值。

将比例函数定义为 P 神经元的状态函数，则其状态为：

$$v_P(k)=k_P N_P(k) \tag{4.8}$$

（b） $I^{\lambda}$ 神经元

分数阶积分控制器表达式为：

$$u_I(k)=k_I\sum{}_k^{\lambda}e(k)=k_I\left[e(k)+\sum_{j=0}^{k-1}c_j^{-\lambda}e(k-j)\right] \tag{4.9}$$

假设每个 $I^{\lambda}$ 神经元都有 $n-1$ 个输入分支连接到它，由 $i$ 表示，那么在任意时刻 $k$，分数阶积分神经元的输入为：

$$N_I(k)=\sum_{i=1}^{n-1}\omega_{ihI}(k)x_{iI}(k) \tag{4.10}$$

式中，$x_{iI}(k)$ 是 $k$ 时刻连接到 $I^{\lambda}$ 神经元的第 $i$ 个神经元的输出，$\omega_{ihI}(k)$ 是 $I^{\lambda}$ 神经元与第 $i$ 个神经元 $k$ 时刻的连接权值。

将分数阶积分函数定义为 $I^{\lambda}$ 神经元的状态函数，则其状态为：

$$v_I(k)=k_I\left[N_I(k)+\sum_{j=0}^{k-1}c_j^{-\lambda}N_I(k-j)\right] \tag{4.11}$$

（c） $D^{\mu}$ 神经元

分数阶微分控制器表达式为：

$$u_D(k)=k_D\Delta_k^{\mu}e(k)=k_D\left[e(k)+\sum_{j=0}^{k-1}c_j^{\mu}e(k-j)\right] \tag{4.12}$$

假设每个 $D^{\mu}$ 神经元都有 $n-1$ 个输入分支连接到它，由 $i$ 表示，那么在任意时刻 $k$，分数阶积分神经元的输入为：

$$N_D(k)=\sum_{i=1}^{n-1}\omega_{ihD}(k)x_{iD}(k) \tag{4.13}$$

式中，$x_{iD}(k)$ 是 $k$ 时刻连接到 $D^{\mu}$ 神经元的第 $i$ 个神经元的输出，$\omega_{ihD}(k)$ 是 $D^{\mu}$ 神经元与第 $i$ 个神经元 $k$ 时刻的连接权值。

将分数阶积分函数定义为 $D^{\mu}$ 神经元的状态函数，则其状态为：

$$v_D(k)=k_D\left[N_D(k)+\sum_{j=0}^{k-1}c_j^{\mu}N_D(k-j)\right] \tag{4.14}$$

在所提出的 $PI^{\lambda}D^{\mu}NN$ 中，连接权值起到 $k_P$、$k_I$ 和 $k_D$ 参数的作用，可以自适应地更新，因此可以忽略 $k_P$、$k_I$ 和 $k_D$ 的影响。为了简化计算，定义 $k_P=k_I=k_D=1$。因此，$PI^{\lambda}D^{\mu}NN$ 隐含层分数阶比例、积分和微分神经元状态可以分别按照式（4.15）~（4.17）进行更新：

$$v'_{sP}(k)=N'_{sP}(k) \tag{4.15}$$

$$v'_{sI}(k)=N'_{sI}(k)+\sum_{j=0}^{k-1}c_j^{-\lambda}N'_{sI}(k-j) \tag{4.16}$$

$$v'_{sD}(k)=N'_{sD}(k)+\sum_{j=0}^{k-1}c_j^{\mu}N'_{sD}(k-j) \tag{4.17}$$

此方法将分数阶 PID 算法融入到 PIDNN 的隐含层的神经元，充分利用了分数阶算法的长记忆和遗传特性，通过从开始时刻到当前时刻的所有时刻的值来进行权重更新，提高了控制的响应速度和精确度，改善了 PIDNN 的控制效果。

（2） 基于 $PI^{\lambda}D^{\mu}$ 神经元的 $PI^{\lambda}D^{\mu}NN$

$PI^{\lambda}D^{\mu}NN$ 是将分数阶 PID 算法引入到 PIDNN 的隐含层，即将 PIDNN 中 P，I 和 D 神经元采用分数阶算法的 P，$I^{\lambda}$ 和 $D^{\mu}$ 神经元替代，神经元的状态函数也分别转换为分数阶的比例，积分和微分函数，如图 4.2 所示，隐含层神经元已经被 P，$I^{\lambda}$ 和 $D^{\mu}$ 神经元替代。

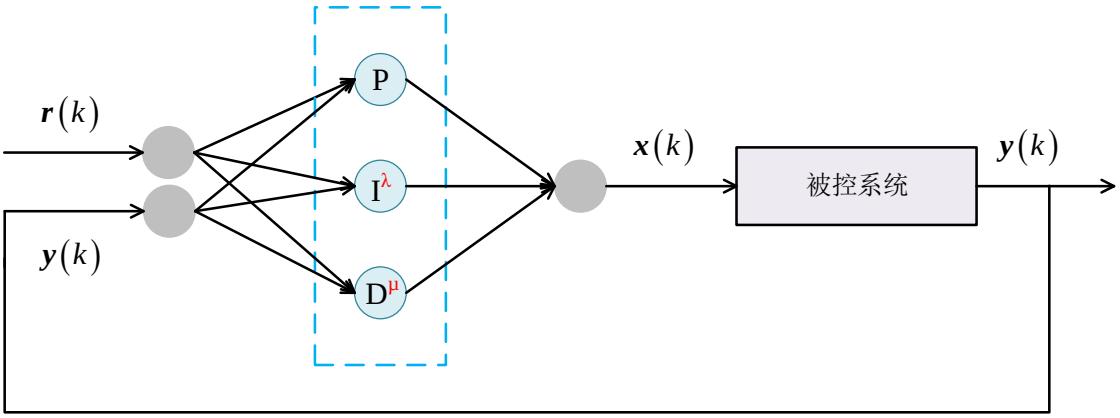


图 4.2 PI$^{\lambda}$D$^{\mu}$NN 解耦控制图

PI$^{\lambda}$D$^{\mu}$NN 的成本函数可定义为[36]：

$$J=\sum_{p=1}^{n}E_p=\frac{1}{l}\sum_{p=1}^{n}\sum_{k=1}^{l}\left[r_p(k)-y_p(k)\right]^2=\frac{1}{l}\sum_{p=1}^{n}\sum_{k=1}^{l}e_p^2(k) \tag{4.18}$$

式中，$e_p$ 为理想信号与系统实际输出之间的差值，$E_p$ 为理想信号与系统实际输出的偏差平方均值，$l$ 为采样点数，$n$ 为被控变量个数。

PI$^{\lambda}$D$^{\mu}$NN 各层权值按照如下梯度下降法进行更新[73]：

$$\boldsymbol{\omega}(k+1)=\boldsymbol{\omega}(k)-\eta\frac{\partial J}{\partial\boldsymbol{\omega}(k)} \tag{4.19}$$

（a） 隐含层到输出层

隐含层到输出层的连接权值可按式（4.20）更新：

$$\omega_{sho}(k+1)=\omega_{sho}(k)-\eta_1\frac{\partial J}{\partial\omega_{sho}(k)} \tag{4.20}$$

式中，$\eta_1$ 为隐含层到输出层的学习率，且

$$\frac{\partial J}{\partial\omega_{sho}(k)}=-\sum_{h=1}^{3}\varphi(k)x_{sh}^{'}(k) \tag{4.21}$$

（b） 输入层到隐含层

输入层到隐含层的连接权值可按式（4.22）更新：

$$\omega_{sih}(k+1)=\omega_{sih}(k)-\eta_2\frac{\partial J}{\partial\omega_{sih}(k)} \tag{4.22}$$

式中，$\eta_2$ 为输入层到隐含层的学习率，且

$$\frac{\partial J}{\partial\omega_{sih}(k)}=-\sum_{i=1}^{2}\sum_{h=1}^{3}\omega_{sho}(k)\operatorname{sgn}\frac{v_{sh}^{'}(k)-v_{sh}^{'}(k-1)}{N_{sh}^{'}(k)-N_{sh}^{'}(k-1)}x_{si}(k)\varphi(k) \tag{4.23}$$

$PI^{\lambda}D^{\mu}$NN 利用分数阶 PID 算法定义了 $PI^{\lambda}D^{\mu}$ 神经元，从 PIDNN 的结构上改进了 PIDNN，获得了分数阶 PID 控制器精确度高、稳定性好和超调小等优势，能更好地完成解耦控制任务。

## 4.2 PID 神经网络的稳定性分析

控制器的稳定性是决定着控制器是否具有实际应用意义的重要条件，它也代表着控制器对被控对象的控制效果。因此，本章根据 Lyapunov 稳定性定理对 PIDNN 控制器的稳定性进行分析。

定义 Lyapunov 函数为：

$$W(k)=\frac{1}{2}\sum_{p=1}^{n}e_p^2(k) \tag{4.24}$$

Lyapunov 函数的差值为：

$$\begin{aligned} W(k)&=W(k+1)-W(k)\\ &=\frac{1}{2}\sum_{p=1}^{n}\left[e_p^2(k+1)-e_p^2(k)\right] \end{aligned} \tag{4.25}$$

误差函数的差值可表达为：

$$\Delta e_p(k)=\left(\frac{\partial e_p(k)}{\partial \boldsymbol{\omega}(k)}\right)^{\mathrm{T}}\Delta\boldsymbol{\omega}(k) \tag{4.26}$$

式（4.26）中误差函数的具体表达式为：

$$e_p(k)=r_p(k)-y_p(k) \tag{4.27}$$

式中，$r_p(k)$为被控系统的理想信号，$y_p(k)$为被控系统的实际输出。

根据 PIDNN 的原理，可以得到：

$$\sum_{p=1}^{n}\frac{\partial e_p(k)}{\partial \boldsymbol{\omega}(k)}=\sum_{p=1}^{n}\frac{\partial e_p(k)}{\partial y_p(k)}\frac{\partial y_p(k)}{\partial x_o(k)}\frac{\partial x_o(k)}{\partial \boldsymbol{\omega}(k)}=-\sum_{p=1}^{n}\frac{\partial y_p(k)}{\partial x_o(k)}\frac{\partial x_o(k)}{\partial \boldsymbol{\omega}(k)} \tag{4.28}$$

由式（4.19）得：

$$\Delta\boldsymbol{\omega}(k)=-\eta\frac{\partial J}{\partial \boldsymbol{\omega}(k)} \tag{4.29}$$

将式（4.18）代入式（4.29）得：

$$\Delta\boldsymbol{\omega}(k)=-2\eta\sum_{p=1}^{n}e_p(k)\frac{\partial e_p(k)}{\partial \boldsymbol{\omega}(k)} \tag{4.30}$$

由式（4.28）和式（4.30），式（4.26）可以改写为：

$$\begin{aligned}\Delta\sum_{p=1}^{n}e_p(k)&=-2\eta\sum_{p=1}^{n}e_p(k)\frac{\partial e_p(k)}{\partial\boldsymbol{\omega}(k)}\left(\frac{\partial e_p(k)}{\partial\boldsymbol{\omega}(k)}\right)^{\mathrm{T}}\\&=-2\eta\sum_{p=1}^{n}e_p(k)\left(-\frac{\partial y_p(k)}{\partial x_o(k)}\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right)\left(-\frac{\partial y_p(k)}{\partial x_o(k)}\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right)^{\mathrm{T}}\\&=-2\eta\sum_{p=1}^{n}e_p(k)\left(-\frac{\partial y_p(k)}{\partial x_o(k)}\right)^2\left\|\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right\|^2\end{aligned}\tag{4.31}$$

即：

$$\Delta\sum_{p=1}^{n}e_p(k)=-2\eta\sum_{p=1}^{n}e_p(k)\left(\frac{\partial y_p(k)}{\partial x_o(k)}\right)^2\left\|\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right\|^2\tag{4.32}$$

又有：

$$e_p(k+1)=e_p(k)+\Delta e_p(k)\tag{4.33}$$

则式（4.25）可以改写为：

$$\begin{aligned}\Delta W(k)&=\frac{1}{2}\sum_{p=1}^{n}\big(e_p(k+1)-e_p(k)\big)\big(e_p(k+1)+e_p(k)\big)\\&=\frac{1}{2}\sum_{p=1}^{n}\Delta e_p(k)\big(\Delta e_p(k)+2e_p(k)\big)\\&=\sum_{p=1}^{n}\left[e_p(k)\Delta e_p(k)+\frac{1}{2}\big(\Delta e_p(k)\big)^2\right]\end{aligned}\tag{4.34}$$

根据 Lyapunov 稳定性理论，当$W(k)\le 0$时，系统是稳定的，即：

$$e_p(k)\Delta e_p(k)\le-\frac{1}{2}\big(\Delta e_p(k)\big)^2\tag{4.35}$$

将式（4.32）代入式（4.35）得：

$$-2\eta\sum_{p=1}^{n}\big(e_p(k)\big)^2\left(\frac{\partial y_p(k)}{\partial x_o(k)}\right)^2\left\|\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right\|^2\le-\frac{1}{2}\left(-2\eta\sum_{p=1}^{n}e_p(k)\left(\frac{\partial y_p(k)}{\partial x_o(k)}\right)^2\left\|\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right\|^2\right)^2\tag{4.36}$$

所以：

$$1\ge\eta\sum_{p=1}^{n}\left(\frac{\partial y_p(k)}{\partial x_o(k)}\right)^2\left\|\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right\|^2\tag{4.37}$$

即：

$$0<\eta\le\frac{1}{\sum_{p=1}^{n}\left(\frac{\partial y_p(k)}{\partial x_o(k)}\right)^2\left\|\frac{\partial x_o(k)}{\partial\boldsymbol{\omega}(k)}\right\|^2}\tag{4.38}$$

由上述推导可得，当学习率满足式（4.38），PIDNN 满足稳定性要求。

## 4.3 基于天牛群优化算法的分数阶 PID 神经网络

分数阶的 P、$I^{\lambda}$、$D^{\mu}$ 神经元使分数阶 PID 神经网络具有了动态记忆特性，可以获得更好的解耦控制性能，但其初始权值仍然是随机选取的，会影响其解耦控制效果，需要进一步的改进。

第二章中介绍了天牛群算法的优化原理，天牛的搜索机制加上种群合作机制提高了算法的准确性和稳定性，能够快速得到最优解。对于 BSO 算法，假设乘法和加法所需的时间分别为$T_m$和$T_a$，则每只天牛执行一次更新的计算复杂度列于表 4.1 中。将天牛个数设为$\rho$，神经网络训练次数设为$length$，则 BSO 算法的计算复杂度为$length*\rho*(13T_a+15T_m)$。

表 4.1 BSO 算法的计算复杂度

| 变量 | 加运算数量 | 乘运算数量 |
|---|---|---|
| $P_{iq}$ | 3 | 2 |
| $\xi_{iq}$ | 1 | 2 |
| $\delta$ | 1 | 1 |
| $V_{iq}$ | 4 | 5 |
| $\vartheta$ | 2 | 2 |
| $P_{rq}$ | 1 | 1 |
| $P_{lq}$ | 1 | 1 |
| $d$ | 0 | 1 |
| 总和 | 13 | 15 |

由于不需要计算空间距离，避免了平方水平的计算复杂性，因此 BSO 算法的计算复杂度不高，且文献[74]中对 BSO 算法进行了非参数统计检验，比较了 BSO 算法和许多其他进化算法，如遗传算法，蚱蜢优化算法等，验证了其在单模态、多模态、固定维度多模态函数和多目标问题中依然具有很强的竞争力，展现出良好的鲁棒性和运行速度。

经比较，BSO 算法具有不俗的优化能力，且计算量合适，具有平衡的优化能力和计算复杂度，因此，本节采用 BSO 算法获取 $PI^{\lambda}D^{\mu}NN$ 的最优初始权值，进一步加快收敛速度，提高解耦控制质量，提出基于 BSO 算法的 $PI^{\lambda}D^{\mu}NN$（BSO-$PI^{\lambda}D^{\mu}NN$）。

图 4.3 描述了 BSO-$PI^{\lambda}D^{\mu}NN$ 的解耦控制方案，其中 $\boldsymbol{e}(k)=\boldsymbol{r}(k)-\boldsymbol{y}(k)$ 是 $PI^{\lambda}D^{\mu}NN$ 的输入，$\boldsymbol{\omega}_{initial}$ 是 BSO 算法确定的最优初始权值，$\boldsymbol{x}(k)$ 是被控系统的输入即 $PI^{\lambda}D^{\mu}NN$ 的输出。

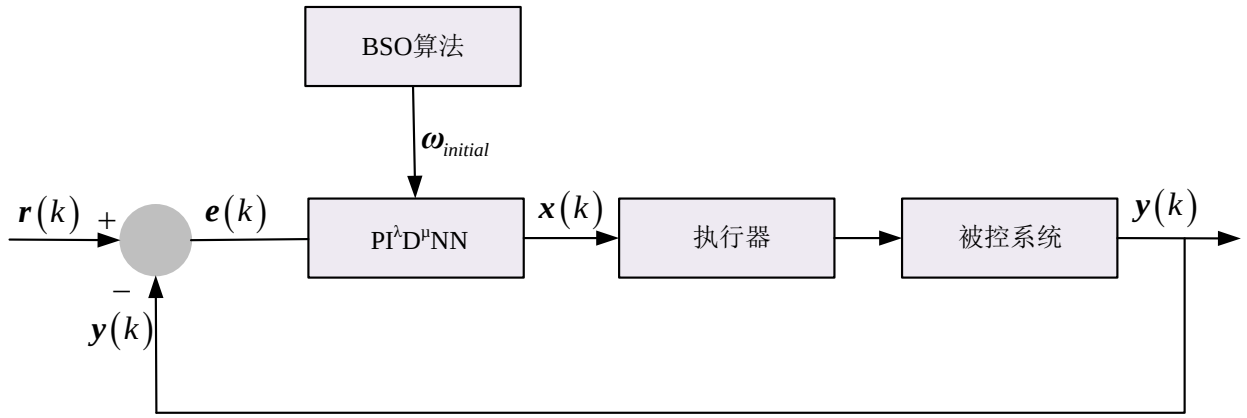


图 4.3 BSO-PI$^{\lambda}$D$^{\mu}$NN 解耦控制方案图

此闭环控制方案中，首先由 BSO 算法根据适应度函数搜索确定 PI$^{\lambda}$D$^{\mu}$NN 的最优初始权值 $\boldsymbol{\omega}_{initial}$，然后 PI$^{\lambda}$D$^{\mu}$NN 根据设定的理想信号 $\boldsymbol{r}$ 和被控系统的实际输出 $\boldsymbol{y}$ 之间的差值 $\boldsymbol{e}$ 来更新权值。在此过程中，系统实际输出 $\boldsymbol{y}$ 一直朝着理想信号 $\boldsymbol{r}$ 追踪，最终当目标函数最小化时，输出 $\boldsymbol{y}$ 可以达到理想信号 $\boldsymbol{r}$ 并保持稳定。执行器只需根据 BSO-PI$^{\lambda}$D$^{\mu}$NN 输出来设定系统输入，待执行动作完成后，被控系统所有输出均可达到控制目标，同时完成解耦和控制。

定义如下适应度函数：

$$f(t)=\sum_{t=1}^{T}e^{0.01\cdot t}\left(\sum_{k=1}^{length}\sum_{p=1}^{n}\left|r_p(k)-y_p(k)\right|\right) \tag{4.39}$$

式中，$T$ 为 BSO 算法的最大迭代次数，$length$ 为 PI$^{\lambda}$D$^{\mu}$NN 训练次数。

BSO-PI$^{\lambda}$D$^{\mu}$NN 的解耦控制流程由算法 4.1 描述：

算法 4.1： BSO-PI$^{\lambda}$D$^{\mu}$NN 解耦控制算法

输入： $\boldsymbol{r},\boldsymbol{y}$

输出： $\boldsymbol{x}_o^{''}$

1：初始化 IBSO 算法参数，给定 PI$^{\lambda}$D$^{\mu}$NN 的训练次数

2：IBSO 算法迭代次数： $T$

3：PI$^{\lambda}$D$^{\mu}$NN 训练次数： $length$

4： *while* $(t<T)$ *do*

5： 计算每个位置的适应度值并对其进行排序，确定当前最优位置

6： *for* $t=1:T$ *do*

7： 通过式（2.45）更新天牛速度 $V_{iq}$

8： 通过式（2.42）更新天牛位置 $P_{iq}$

9： *if* $f(t)<f_{pbest}(t)$

10： $f(t)=f_{pbest}(t)$

11： *if* $f(t)<f_{gbest}(t)$

12： $f(t)=f_{gbest}(t)$

13： 计算当前适应度值，如果优于个体极值或群体极值，更新它

14： $t=t+1$

15： *end for*

16： *end while*

17：输出当前最优位置向量

18：PI$^{\lambda}$D$^{\mu}$NN 以 BSO 算法输出的最优解作为初始权值

19： *for* $k=1:length$ *do*

20： 通过式（2.3）计算 $x_{si}(k)$

21： 通过式（4.22）更新输入层到隐含层权值

22： 通过式（4.15）~（4.17）计算隐含层神经元状态，再由式（2.8）计算 $x_{sh}^{'}(k)$

23： 通过式（4.20）更新隐含层到输出层权值

24： 通过式（2.11）计算 $x_{o}^{''}(k)$

25： *end for*

26： *end*

BSO 算法确定的最优初始权值可以使 PI$^{\lambda}$D$^{\mu}$NN 加快收敛速度，而分数阶神经元使其能够更精确地调整权值，同时减少超调，提高稳定性，因此，系统输出变量能够快速、准确地跟踪理想信号，达到理想的解耦控制效果。

## 4.4 仿真实例

**例 4.1**：考虑一个如下的 3 输入 3 输出的非线性耦合系统[88]：

$$\begin{cases} y_1(k)=0.3y_1(k-1)+0.4y_2(k-1)+x_1(k-1)/\left[1+x_1^2(k-1)\right] \\ \qquad +0.4x_1^3(k-1)+0.4x_2(k-1) \\ y_2(k)=0.4y_2(k-1)+0.4y_3(k-1)+x_2(k-1)/\left[1+x_2^2(k-1)\right] \\ \qquad +0.3x_2^3(k-1)+0.2x_1(k-1) \\ y_3(k)=0.2y_3(k-1)+0.3y_1(k-1)+x_3(k-1)/\left[1+x_3^2(k-1)\right] \\ \qquad +0.2x_3^3(k-1)+0.3x_2(k-1) \end{cases} \tag{4.40}$$

其中 $\boldsymbol{y}=[y_1,y_2,y_3]^{\mathrm{T}}$ 和 $\boldsymbol{x}=[x_1,x_2,x_3]^{\mathrm{T}}$ 分别是系统的输出向量和输入向量。所需的信号设置为：$\boldsymbol{r}=[0.7,0.5,0.4]^{\mathrm{T}}$。

针对此多变量、强耦合数值系统，使用 PIDNN、BSO-PIDNN 和 BSO-PI$^{\lambda}$D$^{\mu}$NN 控制器对其进行解耦控制，并对比其仿真结果。学习率设置为 $\eta_1=0.006$，$\eta_2=0.001$，分数阶次设置为 $\lambda=0.98,\ \mu=0.9$，训练次数设置为 200，天牛个体设置为 50 只。BSO 算法参数设置为：$b_1=b_2=1.5$，$\vartheta_{\max}=0.9$，$\vartheta_{\min}=0.1$，$T=30$。

图 4.4 和图 4.5 分别显示了数值系统在 PIDNN、BSO-PIDNN 和 BSO-PI$^{\lambda}$D$^{\mu}$NN 三种解耦控制器下的输出响应和输出误差收敛曲线，图 4.6 显示了 BSO-PI$^{\lambda}$D$^{\mu}$NN 的输出。

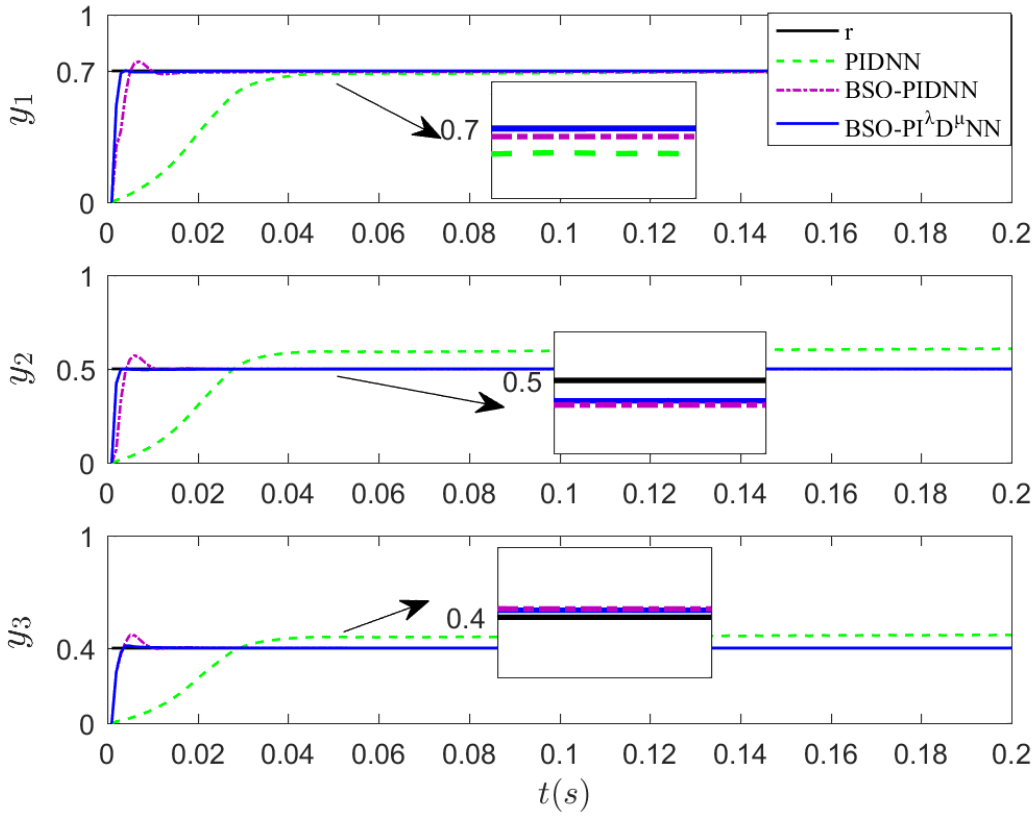


图 4.4　不同解耦控制器下数值系统的输出响应

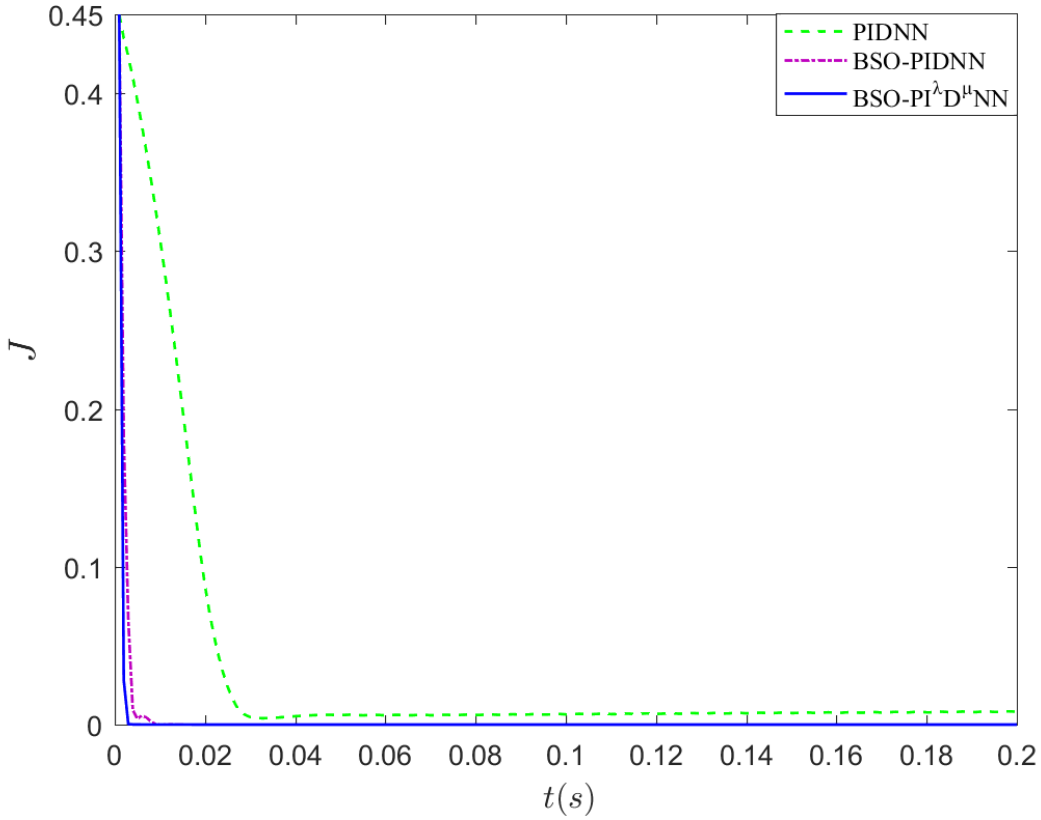


图 4.5　不同解耦控制器下数值系统的输出误差收敛曲线

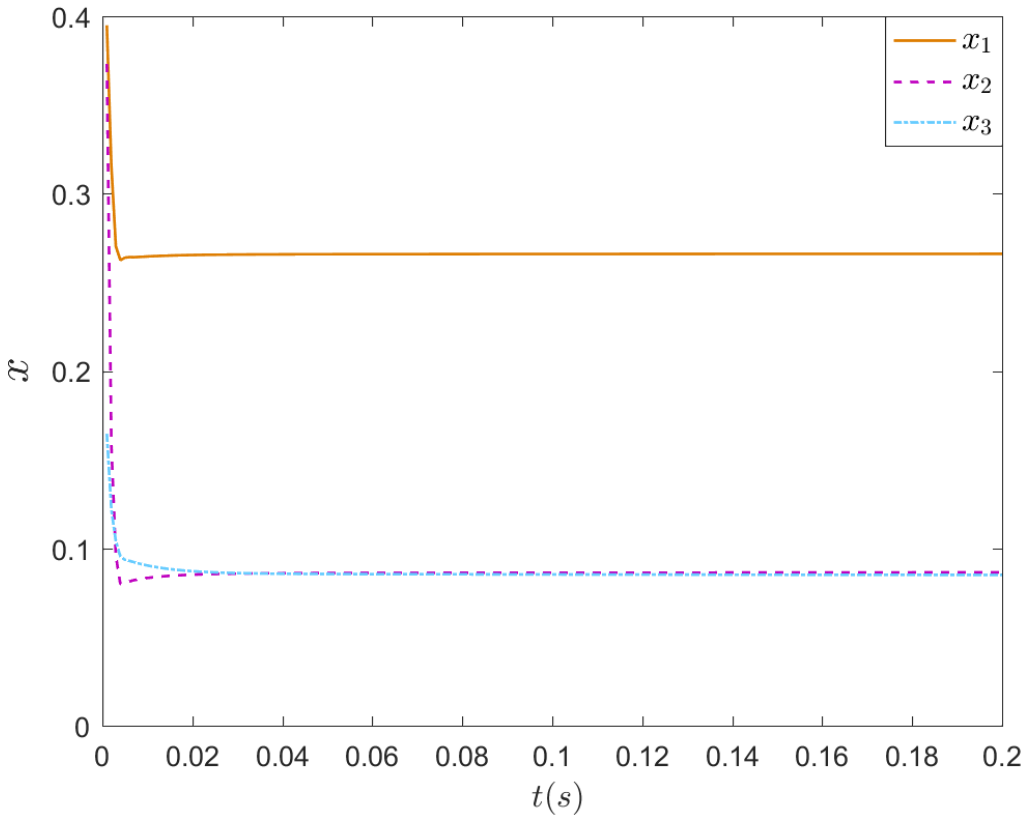


图 4.6 BSO-PI$^{\lambda}$D$^{\mu}$NN 的输出

为更好的验证本章所提出的解耦控制器的有效性，引用文献[88]中的基于改进的粒子群算法优化的 PIDNN（PSO-DE-PIDNN）的结果来加强对比，各类不同控制器下系统输出变量的最小追踪时间如表 4.2 所示，$T_s$ 代表输出变量达到理想信号所需的最短时间。

表 4.2 不同解耦控制器下数值系统输出变量最小追踪时间

| 控制器 / 追踪时间 $T_s(s)$ | PIDNN | PSO-DE-PIDNN | BSO-PIDNN | BSO-PI$^{\lambda}$D$^{\mu}$NN |
|---|---|---|---|---|
| $y_1$ | 0.047 | 0.0125 | 0.010 | 0.004 |
| $y_2$ | 0.028 | 0.0225 | 0.011 | 0.003 |
| $y_3$ | 0.029 | 0.0185 | 0.011 | 0.004 |

从图 4.4 ~ 4.6 和表 4.2 可以看出：

（1）图 4.4 中，PIDNN 在 $y_2$ 和 $y_3$ 的控制上具有非常大的误差，BSO-PIDNN 控制的精确度有明显提升，但其在三个变量的控制中都出现了较大的超调，而所提出的 BSO-PI$^{\lambda}$D$^{\mu}$NN 控制器可以很好的将变量之间解耦，虽然在 $y_3$ 中出现了微量超调，但可以保证所有变量快速精确地跟踪到理想信号，大大加快了跟踪速度，提供了非常精确、稳定的解耦控制效果。

（2）由图 4.5、图 4.6 和表 4.2 看出，BSO-PI$^{\lambda}$D$^{\mu}$NN 给数值系统提供了稳定正确的输入，

使得三个输出变量在保证精确度的情况下均快速追踪到理想信号，收敛时间均在 0.005s 以内，对比其他控制器极大地提升了收敛速度，展现了 BSO-PI$^{\lambda}$D$^{\mu}$NN 优越的控制性能以及 BSO 算法良好的优化性能，验证了 PI$^{\lambda}$D$^{\mu}$NN 能够提供稳定、优越的解耦控制性能。

**例 4.2：**考虑如 2.3 节中的温室系统仿真实例，即如下的简化三输入二输出模型[80]：

$$\begin{cases} y_1(k)=0.897y_1(k-1)+0.202x_1(k)-0.371x_2(k)+0.250x_3(k) \\ y_2(k)=0.400y_2(k-1)+0.250x_1(k)+0.500x_2(k)-0.250x_3(k)[y_2(k-1)-0.450] \end{cases} \tag{4.41}$$

其中，$\boldsymbol{y}=[y_1,y_2]^{\mathrm{T}}$ 和 $\boldsymbol{x}=[x_1,x_2,x_3]^{\mathrm{T}}$ 分别是系统输出和输入向量。目标信号设置为：$\boldsymbol{r}=[0.5,0.7]^{\mathrm{T}}$。

针对此多变量、强耦合温室系统，使用 PIDNN、基于 PSO 算法的 PI$^{\lambda}$D$^{\mu}$NN（PSO-PI$^{\lambda}$D$^{\mu}$NN）和 BSO-PI$^{\lambda}$D$^{\mu}$NN 控制器对其进行解耦控制，并对比其仿真结果。学习率设置为 $\eta_1=0.005$，$\eta_2=0.002$，分数阶次设置为 $\lambda=0.98,\ \mu=0.9$，训练次数设置为 200，天牛个体设置为 50 只。BSO 算法参数设置为：$b_1=b_2=1.5$，$\vartheta_{\max}=0.9$，$\vartheta_{\min}=0.1$，$T=30$。

图 4.7 和图 4.8 分别显示了温室系统在 PIDNN、PSO-PI$^{\lambda}$D$^{\mu}$NN 和 BSO-PI$^{\lambda}$D$^{\mu}$NN 三种解耦控制器下的输出响应和输出误差收敛曲线，图 4.9 显示了 BSO-PI$^{\lambda}$D$^{\mu}$NN 的输出。

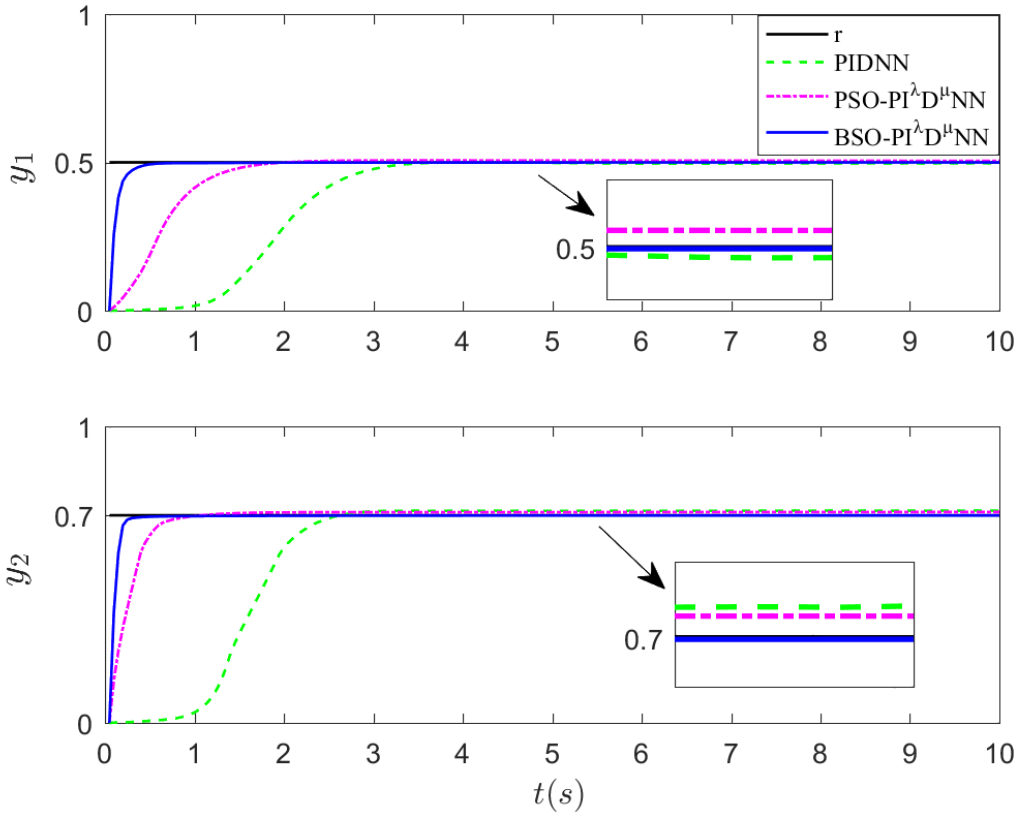


图 4.7　不同解耦控制器下温室系统的输出响应

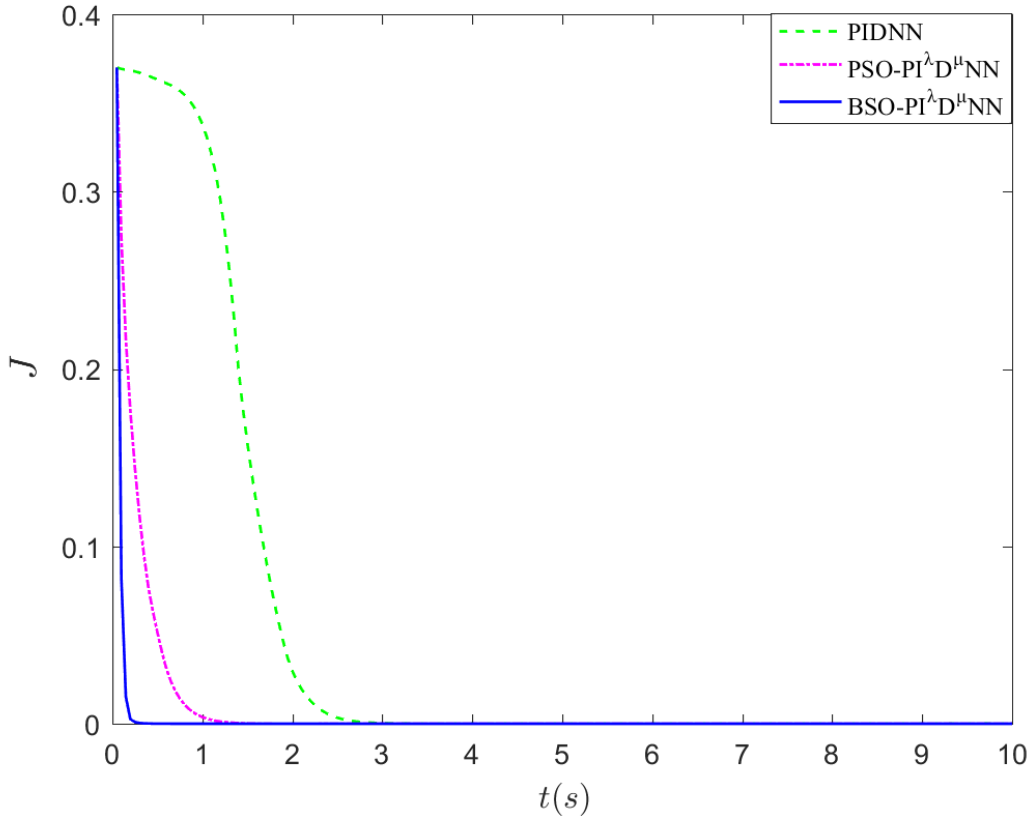


图 4.8 不同解耦控制器下的温室系统输出误差收敛曲线

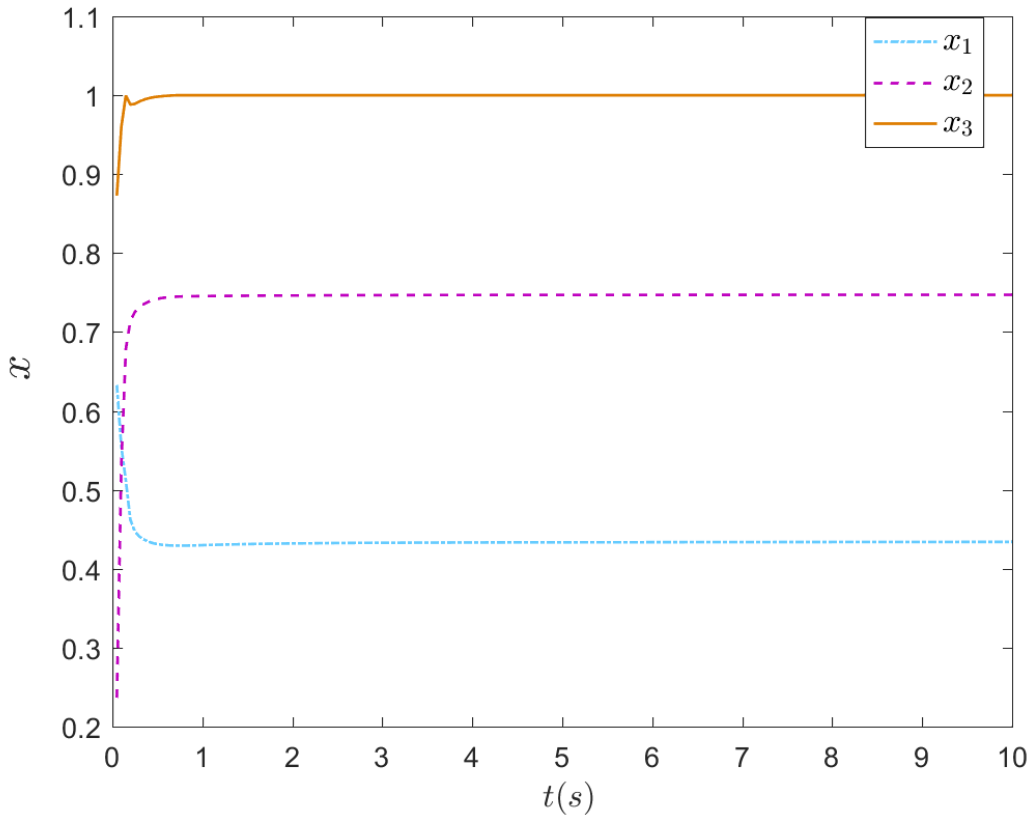


图 4.9 BSO-PI$^{\lambda}$D$^{\mu}$NN 的输出

为更好的验证本章所提出的解耦控制器的有效性，引用文献[80]中的基于粒子群算法优化的 PIDNN（PSO-PIDNN）的结果来加强对比，各类不同控制器下系统输出变量的最小追踪时间如表 4.3 所示，$T_s$ 代表输出变量达到理想信号所需的最短时间。

表 4.3 不同解耦控制器下温室系统输出变量最小追踪时间

| 控制器 / 追踪时间 $T_s(s)$ | PIDNN | PSO-PIDNN | PSO-PI$^{\lambda}$D$^{\mu}$NN | BSO-PI$^{\lambda}$D$^{\mu}$NN |
|---|---|---|---|---|
| $y_1$ | 3.35 | 2.20 | 1.85 | 0.60 |
| $y_2$ | 2.50 | 1.70 | 0.95 | 0.45 |

从图 4.7～4.9 和表 4.3 可以看出：

（1）在图 4.7 中，系统中温度和湿度均实现了单独控制，表明三种控制器都可以达到解耦控制效果，但 PIDNN 在初始阶段的响应较慢，需要较长的时间来跟踪目标信号，结合上两节的结果可以看出，PIDNN 解耦控制效果十分不稳定。PIDNN 和 PSO- PI$^{\lambda}$D$^{\mu}$NN 在温度和湿度的控制中都出现了少量误差，而采用提出的 BSO-PI$^{\lambda}$D$^{\mu}$NN 时，精确度有了显著的提高，跟踪误差迅速减小至近零，达到稳态且无超调。

（2）图 4.8、图 4.9 表明了 BSO-PI$^{\lambda}$D$^{\mu}$NN 收敛速度快，能够给温室系统提供稳定正确的输入，相较于经典 PIDNN，温度的追踪时间减少 6 倍左右，湿度追踪时间减少 5 倍左右，大大提高了 PIDNN 的实用性。

（3）表 4.3 的数据验证了 BSO 算法相较于 PSO 算法可以使 PIDNN 具有更快的收敛速度，说明 BSO 算法具有更好的优化能力，加上分数阶神经元的优势进一步提升了响应速度。

（4）两个仿真例子具有不同的输入和输出数量，但提出的 BSO-PI$^{\lambda}$D$^{\mu}$NN 都非常精确快速地完成了解耦控制目标，验证了 BSO-PI$^{\lambda}$D$^{\mu}$NN 可以很容易地调整输入层和输出层神经元的数量以适应不同系统的输入和输出数量，而不影响神经网络性能，表明其在多变量系统中具有广泛的应用价值。

## 4.5 本章小结

本章介绍了分数阶 PID 算法，并根据其定义了 P，I$^{\lambda}$，D$^{\mu}$ 神经元，给出了 PI$^{\lambda}$D$^{\mu}$NN 的基本结构，然后提出了一种基于 BSO 算法的 PI$^{\lambda}$D$^{\mu}$NN 控制器，采用 BSO 算法对权值进行初始化，避免了随机初始权值带来的各种问题。同时应用李雅普诺夫稳定性理论，推导得出了一个保证神经网络稳定性的充分条件。最后通过仿真实验结果验证了提出的 BSO-PI$^{\lambda}$D$^{\mu}$NN 可以使被控变量更快、更精确、更稳定地追踪理想信号，展现了理想的解耦控制性能。

# 第五章 总结与展望

## 5.1 研究结论

本文主要在 PIDNN 框架下研究解决多变量、强耦合复杂系统的高质量解耦控制问题，针对 PIDNN 初始权值随机以及精确度不高两个关键问题展开研究，从 PIDNN 的内部算法与结构入手，将智能优化算法、分数阶学习算法以及分数阶 PID 算法与 PIDNN 相结合，提出了 IBSO-PIDNN，SSA-FPIDNN 和 BSO-PI$^{\lambda}$D$^{\mu}$NN 三种智能 PIDNN 进行了更加高效的解耦控制。通过仿真验证了提出的智能 PIDNN 可以加速收敛、提升控制精度、减少超调、提高稳定性，获得更加优异的解耦控制性能，为多变量、强耦合复杂系统的解耦控制提供了一些实际可行的方案。本文的主要工作和研究成果如下：

（1）针对传统 PIDNN 初始权值随机选取问题，所提出的 IBSO-PIDNN 相较于 PIDNN 收敛速度明显加快，控制效果更加稳定。IBSO 算法采用非线性方式更新天牛种群惯性权重并在全局最优位置加入突变因子，平衡了天牛局部搜索和全局搜索能力，降低了 BSO 算法陷入早熟风险的概率，可以更好地进行 PIDNN 最优初始权值搜索。通过仿真实验结果验证了 IBSO 算法明显改善了 PIDNN 解耦控制性能，且比 BSO 算法具有更好的优化能力。

（2）针对 PIDNN 精确度不足的问题，所提出的 SSA-FPIDNN 在精确度和收敛速度上都明显优于基于智能优化算法的 PIDNN。分数阶学习算法合理分配了过去不同时刻权值信息对当前权值计算的影响力，使 FPIDNN 能更加精准的更新修正当前时刻权值，提高了解耦控制的精确度和响应速度。同时引入了 SSA 算法获得 FPIDNN 最优初始权值，进一步提高了解耦控制效果。通过仿真实验结果验证了 SSA-FPIDNN 相较于基于智能优化算法的 PIDNN 有更高的控制精度和更快的收敛速度。

（3）针对 FPIDNN 控制效果中可能出现少量超调以及参数选取不适控制输出后期易发散的问题，提出的 BSO-PI$^{\lambda}$D$^{\mu}$NN 在控制性能以及稳定性上具有优越的表现。BSO-PI$^{\lambda}$D$^{\mu}$NN 隐含层为分数阶神经元，使用分数阶 PID 算法进行状态更新，同时采取 BSO 算法优化 PI$^{\lambda}$D$^{\mu}$NN 初值，提升了控制的精确度、加快了响应速度、减少了超调、提高了神经网络控制的稳定性，并通过推导证明了 PI$^{\lambda}$D$^{\mu}$NN 的稳定性。通过仿真实验结果验证了 BSO-PI$^{\lambda}$D$^{\mu}$NN 可以获得收敛快速、稳态精度高、稳定性好等出色的解耦控制表现。

## 5.2 工作展望

本文将智能优化算法、分数阶学习算法以及分数阶 PID 算法与 PIDNN 相结合，提出一系列改进后的智能 PIDNN 用于多变量、强耦合复杂系统的解耦控制，通过实际仿真结果表明所提出的智能 PIDNN 具有更加优越的解耦控制性能。但是由于个人精力以及条件有限，有些研究还值得进一步深入：

（1）本文所提出的一系列智能 PIDNN 控制器现阶段的研究仅限于仿真应用，下一步可以在此研究基础上， 将其应用于实际的耦合系统中，开展实际的实验研究工作。

（2）FPIDNN 采取分数阶学习算法更新修正权值，分数阶学习算法的稳定性尚未证明，后续工作将考虑提高算法的稳定性。

（3）$PI^{\lambda}D^{\mu}NN$ 的分数阶阶次是采取实验法获取，后续可以考虑采用自适应算法或智能算法进行整定，获得更加良好的控制效果。

（4）本文提出的分数阶神经元可以考虑将其应用于其他类型的神经网络，使其他网络获得更好的精确度，同时使分数阶神经元获得更加广泛的应用。

# 参考文献

[1] Sun W, Lin J W, Su S F, et al. Reduced Adaptive Fuzzy Decoupling Control for Lower Limb Exoskeleton[J]. IEEE Transactions on Cybernetics, 2021, 51(3): 1099-1109.

[2] Shi K, Yin X, Jiang L, et al. Perturbation Estimation Based Nonlinear Adaptive Power Decoupling Control for DFIG Wind Turbine[J]. IEEE Transactions on Power Electronics, 2020, 35(1): 319-333.

[3] Lin X B, Sun C Y. A Decoupling Control for Quadrotor UAV Using Dynamic Surface Control and Sliding Mode Disturbance Observer[J]. Nonlinear dynamics, 2019, 97(1): 781-795.

[4] 黄金侠, 侯艳, 宋国义, 等. PID神经元网络在水稻秧棚控制系统中的应用研究[J]. 农机化研究, 2019, 41(7): 105-109.

[5] Nasseri S, Khalesi M R, Ramezani A, et al. An Adaptive Decoupling Control Design for Flotation Column: A Comparative Study Against Model Predictive Control[J]. IETE Journal of Research, 2022, 68(6): 3994-4007.

[6] Zhang Y, Chai T Y, Wang H, et al. Nonlinear Decoupling Control with ANFIS-based Unmodeled Dynamics Compensation for a Class of Complex Industrial Processes[J]. IEEE Transactions on Neural Networks & Learning Systems, 2018, 29(99): 2352-2366.

[7] Glida H E, Abdou L, Chelihi A, et al. Optimal Model-free Backstepping Control for a Quadrotor Helicopter[J]. Nonlinear dynamics, 2020, 100(4): 3449-3468.

[8] Liao Q F, Da S. Sparse and Decoupling Control Strategies Based on Takagi-Sugeno Fuzzy Models.[J]. IEEE transactions on cybernetics, 2019, 51(2): 947-960.

[9] 王博华. 基于模糊控制法的工业炉燃烧系统模糊解耦控制研究[J]. 工业加热, 2022, 51(8): 39-41.

[10] Wang S S, Zhu H Q, Wu M Y, et al. Active Disturbance Rejection Decoupling Control for Three-degree-of-freedom Six-pole Active Magnetic Bearing Based on BP Neural Network[J]. IEEE Transactions on Applied Superconductivity, 2020, 30(4): 3603505.

[11] Jeyaraj P R, Nadar E R S. Real-time Data-driven PID Controller for Multivariable Process Employing Deep Neural Network[J]. Asian Journal of Control, 2022, 24(6): 3240-3251.

[12] Li L, Yuan R, Chen X C, et al. Design of MSCSG Control System Based on ADRC and RBF Neural Network[J]. Journal of Beijing University of Aeronautics and Astronautics, 2020, 46(10): 1966-1972.

[13] Wu L B, Ju H P, Xie X P, et al. Neural Network Adaptive Tracking Control of Uncertain MIMO Nonlinear Systems with Output Constraints and Event-triggered Inputs[J]. IEEE Transactions on Neural Networks & Learning Systems, 2020, 32(2): 695-707.

[14] 唐佳豪, 闻新, 王宁, 等. 基于神经网络的控制力矩陀螺自抗扰解耦控制[J]. 兵工自动化, 2023, 42(2): 35-41.

[15] 舒怀林. PID神经网络对强解耦合带时延多变量系统的解耦控制[J]. 控制理论与应用, 1998, 15(6): 920-924.

[16] Boksenbom A S, Hood R. General Algebraic Method Applied to Control Analysis of Complex Engine types[J]. Technical Report Archive & Image Library, 1949: 581-593.

[17] Waller K V. Impressions of Chemical Process Control Education and Research in the USA[J]. Chemical Engineering Education, 1981, 15(1): 30-34.

[18] Chai T Y, Zhai L F, Yue H. Multiple Models and Neural Networks Based Decoupling Control of Ball Mill Coal-pulverizing Systems[J]. Journal of Process Control, 2011, 21(3): 351-366.

[19] Chekari T, Mansouri R, Bettayeb M. IMC-PID Fractional Order Filter Multi-loop Controller Design for Multivariable Systems Based on Two Degrees of Freedom Control Scheme[J]. International Journal of Control Automation & Systems, 2018, 16(2): 689-701.

[20] Wang C, Zhao W, Luan Z, et al. Decoupling Control of Vehicle Chassis System Based on Neural Network

Inverse System[J]. Mechanical Systems and Signal Processing, 2018, 106:176-197.

[21] Jiang C J, Zhu H Q, Wang X. Decoupling Control of Outer Rotor Coreless Bearingless Permanent Magnet Synchronous Generator Based on Fuzzy Neural Network Inverse System[J]. IEEE Transactions on Transportation Electrification, 2023, doi: 10.1109/TTE.2023.3253544.

[22] 吴雪纯, 王岩松, 郭辉, 等. 多通道PID神经网络解耦有源控制算法[J]. 噪声与振动控制, 2022, 42(6): 38-44.

[23] 马平, 杨金芳, 崔长春, 等. 解耦控制的现状及发展[J]. 控制工程, 2005, 12(2): 97-100.

[24] Benosman M. Model-based vs Data-driven Adaptive Ccontrol: An Overview [J]. International Journal of Adaptive Control and Signal Processing, 2018, 32(5): 753-776.

[25] 胡钦华, 董阿妮, 任庆昌. 自适应解耦补偿控制在变风量空调系统中的应用[J]. 计算机工程与应用, 2009, 45(3): 194-197.

[26] 桑保华, 薛晓中. 多变量解耦控制方法[J]. 火力与指挥控制, 2007, 32(11): 13-16.

[27] 平玉环, 于希宁, 孙剑, 等. 多变量系统模糊解耦方法综述[J]. 仪器仪表用户, 2010, 1(1): 1-2.

[28] An J Q, Yang J Y, Wu M. Decoupling Control Method with Fuzzy Theory for Top Pressure of Blast Furnace[J]. IEEE Transactions on Control Systems Technology, 2018, 27(6): 2735-2742.

[29] Zhou H, Deng H, Duan J. Hybrid Fuzzy Decoupling Control for a Precision Maglev Motion System[J]. IEEE/ASME Transactions on Mechatronics, 2017, 23(1): 389-401.

[30] 赵宏凯, 蒋科坚. 基于BP神经网络的电磁轴承逆系统解耦控制[J]. 电子设计工程, 2021, 29(6): 123-134.

[31] Xu D, Liu J, Yan X G, et al. A Novel Adaptive Neural Network Constrained Control for Multi-area Interconnected Power System with Hybrid Energy Storage[J]. IEEE Transactions on Industrial Electronics, 2018, 65(8): 6625-6634.

[32] 孙宇贞, 张婷, 李朵朵, 等. 基于BP神经网络和GA-PID的超超临界锅炉系统解耦控制研究[J]. 热能动力工程, 2018, 33(5): 92-98.

[33] 程艳明, 柳成, 吴晶, 等. 基于RBF神经网络自抗扰控制器的液体灌装机定位系统的设计[J]. 机床与液压, 2020, 48(18): 172-208.

[34] Zhang J, Xue X J, Liu C G, et al. Feedback Linearization Decoupling and Predictive Driving Control for Dual Independent Electric Drive Tracked Vehicle [J]. Acta Armamentarii, 2018, 42(4): 697-705.

[35] 宋帆, 马小晶, 王宏伟, 等. 基于分数阶的神经网络解耦控制优化方法[J]. 控制工程, 2022, 29(4): 692-698.

[36] 舒怀林. PID神经元网络多变量控制系统分析[J]. 自动化学报, 1999, 25(1): 105-111.

[37] Hao J, Zhang G, Liu W, et al. Data-driven Tracking Control Based on LM and PID Neural Network with Relay Feedback for Discrete Nonlinear Systems[J]. IEEE Transactions on Industrial Electronics, 2020, 68(11): 11587-11597.

[38] 徐会丽, 石明全, 张霞, 等. 基于PIDNN的六旋翼无人机飞行控制算法研究[J]. 传感器与微系统, 2017, 36(12): 25-27.

[39] 高燕, 郑欢欢. 基于优化PID神经网络的光照强度控制算法研究[J]. 电子设计工程, 2019, 27(14): 68-76.

[40] Jia D L, Wang F S, Xu D K. Study on PIDNN-based Flow Control in Layered Water Injection Technology in Oilfield[J]. Advanced Materials Research, 2011, 291-294: 2718-2722.

[41] Liang H, Sang Z K, Wu Y Z, et al. High Precision Temperature Control Performance of a PID Neural Network-controlled Heater Under Complex Outdoor Conditions[J]. Applied Thermal Engineering: Design, Processes, Equipment, Economics, 2021, 195(1): 117234.

[42] Huang R N, Ding N. AUV Vertical Plane Control Based on Improved PID Neural Network Algorithm Huang Runan[J]. Journal of System Simulation, 2020, 32(2): 229-235.

[43] 仲元昌, 郭耿涛, 贾年龙, 等. 晶体生长炉的PID神经网络温度控制算法[J]. 人工晶体学报, 2010, 39(5): 1302-1307.

[44] Zhang H, Qiu Y, Liang H, et al. Research on Remote Intelligent Control Technology of Throttling and Back Pressure in Managed Pressure Drilling[J]. Petroleum, 2020, 7(2): 222-229.

[45] Ying Y Q, Lu J G, Chen J S, et al. PIDNN Based Intelligent Control of Ignition Oven[J]. Advanced Materials Research, 2012, 396-398: 493-497.
[46] Zeng G Q, Xie X Q, Chen M R, et al. Adaptive Population Extremal Optimization-based PID Neural Network for Multivariable Nonlinear Control Systems[J]. Swarm and Evolutionary Computation, 2019, 44: 320-334.
[47] 郭松林, 巴艳坤. 基于改进GWO优化的PID神经网络解耦控制[J]. 黑龙江科技大学学报, 2023, 33(1): 116-122.
[48] 王延年, 武云辉. 改进BP算法优化的纺织空调智能PIDNN控制[J]. 棉纺织技术, 2021, 49(3): 1-5.
[49] 何宁辉, 沙伟燕, 胡伟, 等. 基于QIA-BP神经网络算法的变压器故障诊断[J]. 变压器, 2021, 58(3): 50-56.
[50] 刘威, 付杰, 周定宁, 等. 基于反时限混沌郊狼优化算法的BP神经网络参数优化[J]. 控制与决策, 2021, 36(10): 2339-2349.
[51] Cheng R, Song Y, Chen D, et al. Intelligent Positioning Approach for High Speed Trains Based on Ant Colony Optimization and Machine Learning Algorithms[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 20(10): 3737-3746.
[52] Zhang X, Xu T, Zhao L, et al. Research on Flatness Intelligent Control via GA-PIDNN[J]. 2015, 26(2): 359-367.
[53] 董万里, 曲东才, 董伟洁. 一种基于GA-BP算法的PIDNN控制策略[J]. 兵工自动化, 2011, 30(2): 66-69.
[54] Feng D Q, Chao-Yang M A, Liu Y H. Nonlinear Control System of PID Neural Network Based on Artificia Fish Swarm Algorithm[J]. Computer Simulation, 2012, 29(9): 247-296.
[55] Mohamed M J , Hamza M K . Design PID Neural Network Controller for Trajectory Tracking of Differential Drive Mobile Robot Based on PSO[J]. Engineering and Technology Journal, 2019, 37(12): 574-583.
[56] Zhou J X, Zhao L I, Song D L, et al. PID Neural Network Decoupling Control Based on Improved Particle Swarm Optimization[J]. Machine Tool and Hydraulics, 2018, 46(24): 74-79.
[57] Zhang X L, Fan H M, Zang J Y, et al. The Stabilization and 3D Visual Simulation of the Triple Inverted Pendulum Based on CGA-PIDNN[J]. International Journal of Control, Automation and Systems, 2015, 13(4): 1010-1019.
[58] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.
[59] Abood D L H. A Survey Study of Fractional Order Control Techniques [J]. International Journal of Engineering Applied Sciences and Technology, 2021, 6(2): 1-4.
[60] Oustaloup A. La Dérivation non Entière: Théorie, Synthèse et Applications[M]. Paris: Hermès, 1995.
[61] Tarasov V E. On History of Mathematical Economics: Application of Fractional Calculus[J]. Mathematics, 2019, 7(6): 509-537.
[62] Mishra S, Mishra L N, Mishra R K, et al. Some Aapplications of Fractional Calculus in Technological Development[J]. Journal of Fractional Calculus and Applications, 2019, 10(1): 228-235.
[63] Zheng Z, Zhao W, Dai H. A New Definition of Fractional Derivative[J]. International Journal of Nonlinear Mechanics, 2019, 108: 1-6.
[64] Diethelm K, Kiryakova V, Luchko Y, et al. Trends, Directions for Further Research, and Some Open Problems of Fractional Calculus[J]. Nonlinear Dynamics, 2022, 107(4): 3245-3270.
[65] 薛定宇. 分数阶微积分学与分数阶控制[M]. 北京: 科学出版社, 2018: 31-35.
[66] Podlubny I. Fractional-order Systems and $PI^{\lambda}D^{\mu}$-controllers[J]. IEEE Transactions on Automatic Control, 1999, 44(1): 208-214.
[67] 潘志锋, 王孝洪, 吴春台, 等. 基于分数阶PID控制器和滑动平均滤波器的三相锁相环设计[J]. 高电压技术, 2022, 48(11): 4393-4402.
[68] 艾青林, 蒋锦涛, 刘刚江, 等. 基于遗传算法与模糊分数阶PID的钢结构损伤检测机器人姿态控制[J]. 高技术通讯, 2022, 32(6): 615-623.
[69] Dimeas I, Petras I, Psychalinos C, et al. New Analog Implementation Technique for Fractional-order

Controller: A DC Motor Control[J]. International Journal of Electronics and Communications, 2017, 78: 192-200.
[70] Debbarma S, Dutta A. Utilizing Electric Vehicles for LFC in Restructured Power Systems Using Fractional Order Controller[J]. IEEE Transactions on Smart Grid, 2017, 8(99): 2554-2564.
[71] Chen M R, Chen B P, Zeng G Q, et al. An Adaptive Fractional-order BP Neural Network Based on Extremal Optimization for Handwritten Digits Recognition[J]. Neurocomputing, 2020, 391(28): 260-272.
[72] Chen B P, Chen Y, Zeng G Q, et al. Fractional-order Convolutional Neural Networks with Population Extremal Optimization[J]. Neurocomputing, 2022, 477(7): 36-45.
[73] 舒怀林. PID神经元网络及其控制系统[M]. 北京: 国防工业出版社, 2006: 23-34.
[74] Wang T T, Yang L, Liu Q. Beetle Swarm Optimization Algorithm: Theory and Application[J]. Filomat, 2020, 34(15): 5121-5137.
[75] Jiang X Y, Li S. BAS: Beetle Antennae Search Algorithm for Optimization Problems[J]. International Journal of Robotics and Control, 2017, 1(1): 1-3.
[76] Jiang X Y, Li S. Beetle Antennae Search Without Parameter Tuning (BAS-WPT) for Multi-objective Optimization, Filomat, 2020, 34(15): 5113-5119.
[77] Kennedy J, Eberhart R C. A Discrete Binary Version of the Particle Swarm Algorithm [C]. IEEE International Conference on Systems, Man, and Cybernetics, 1997: 4104-4108.
[78] Chen Y, Liu Y C, Wang C. Feedback Particle Swarm Optimization for Shipboard Power System Restoration[J]. Advanced Materials Research, 2015, 1070-1072: 1902-1905.
[79] Liang H T, Kang F H. Adaptive Mutation Particle Swarm Algorithm with Dynamic Nonlinear Changed Inertia Weight[J], International Journal for Light and Electron Optics, 2016, 127(19): 8036-8042.
[80] 王鹏, 龚瑞昆. 基于粒子群算法优化PIDNN的温室系统解耦控制[J]. 江苏农业科学, 2018, 46(23): 241-244.
[81] Hayashi Y, Buckley J J, Czogala E. Fuzzy Neural Network with Fuzzy Signals and Weights[J]. International Journal of Intelligent Systems, 1993, 8(4): 527-537.
[82] Petráš I. Fractional-order Nonlinear Systems[M]. Beijing: Higher Education Press, 2011: 9-20.
[83] Xue J K, Shen B. A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm[J]. Systems Science & Control Engineering, 2020, 8(1): 22-34.
[84] Li J H, Lei Y S, Yang S H. Mid-long Term Load Forecasting Model Based on Support Vector Machine Optimized by Improved Sparrow Search Algorithm[J]. Energy Reports, 2021, 8(5): 491-497.
[85] 印雷, 顾德, 刘飞. 基于改进麻雀搜索算法优化的DV-Hop定位算法[J]. 传感技术学报, 2021, 34(5): 670-675.
[86] 俞凯耀, 席东民. 人工鱼群算法优化的 PID 神经网络解耦控制[J]. 计算机仿真, 2014, 31(10): 350-353.
[87] Goodrich C, Peterson A C. Discrete Fractional Calculus[M]. Switzerland: Springer, Cham, 2015: 107-125.
[88] Ye H T, Li Z Q. PID Neural Network Decoupling Control Based on Hybrid Particle Swarm Optimization and Differential Evolution[J]. International Journal of Automation and Computing, 2015, 17(4): 1-6.