应用 TCP/IP 实时通信框架的核电控制智能仿真平台

陈芷安[1]，王泰哲[2]，佘兢克[1](✉)，钱科宇[1]，曾之傲[3]
1. 湖南大学，信息科学与工程学院，长沙 410082
2. 北京市中保网盾科技有限公司，北京，100083
3. 南京邮电大学 210003
shejingke@hnu.edu.cn

摘 要：为了解决核电站智能控制仿真中依赖离线预测参数对传统控制进行智能优化的问题，本文将现有智能控制优化仿真平台解耦为 Simulink 控制仿真平台、基于 TCP/IP 的实时通信框架以及在深度学习框架 PyTorch 内独立实时运行的长短期记忆网络（LSTM）模型三大部分。Simulink 控制仿真平台提供实时工况仿真作为智能预测模型的预测依据，并从智能预测模型中获取实时预测的工况参数用于控制回路的反馈补偿。二者所依赖的数据实时互通功能由实时通信框架提供。在针对核电站稳压器控制的验证实验中，智能 PID 控制通过实时通信框架实时获取控制优化参数，实现了传统 PID 控制的性能提升。其超调量降低幅度可达 63.60%，同时调节时间最多可缩短 280 秒，优化比例达到 73.35%。本研究证明了利用实时通信框架实现在线、解耦的训练-仿真逻辑的可行性，为进一步优化核电站智能控制仿真系统结构、提升核电站智能化控制水平提供了有益的借鉴。

关键词：稳压器智能控制；深度学习；TCP/IP；实时通信框架

中图分类号：（作者写，如 TK08） **文献标识码：**B **文章编号：**（杂志社写）

0 引言

机器学习方法已经被广泛应用于解决大型控制问题[1]。基于机器学习的智能优化方法消除了传统控制中普遍存在的对于复杂数学模型的依赖。智能控制模块通过数据驱动，不再需要构建被控对象的数学模型[2]，并且能够保证较好的稳定性能[3]，与传统控制方法相比具有极大的优势。基于神经网络的智能控制实验涉及到神经网络训练、神经网络与传统控制结构结合等综合性问题，对仿真实验提出了较高的要求。

当前，智能控制实验主要是利用 MathWorks 公司推出的 Simulink 软件对模型进行设计和仿真的。Simulink 平台提供了图形化、模块化的设计界面，用户可以使用包括深度学习工具箱在内的多种工具箱快速地进行包括神经网络控制[4]、强化学习控制[5]、模糊控制[6]和专家系统[7]等在内的多种系统设计和仿真。以 PID 智能优化控制为例，文献[8]通过深度学习工具箱实现了具备长时间序列处理与数据预测能力的 LSTM 神经网络，使用 LSTM 执行预测补偿，对传统 PID 控制进行了优化，大幅降低了控制超调量和调节时间。文献[9]实现了基于神经网络的 PID 自整定控制器，得到了比 Ziegler-Nichols 方法更好的效果。

但是，Simulink 执行神经网络智能控制仿真存在一些不足之处。一方面，系统执行的是离线的预测，没有实现实时在线的数据预测，为后期硬件实际部署工作带来困难； 另一方面，Simulink 搭建的智能网络模块与其他控制模块高度耦合，不利于后期维护；此外，Simulink 中深度学习工具箱功能非常有限，难以构建大型的、复杂的网络，也无法方便地进行模型的导入和导出，实验生态较为封闭。而近年来，随着机器学习应用的全面推广，PyTorch 等开源机器学习框架得到了越来越多的关注。文献[10]指出，PyTorch 框架因为其动态计算图特性而广受研究人员欢迎，可以进行快速的网络原型设计与实验，不仅提供了构建强大的人工智能模型所需的工具，而且还支持行业所需的可拓展性和效率。文献[11]在 PyTorch 实现的 MM-MADRL 算法的基础上做了改进，可用于 PID 参数自动整定。文献[12]利用 PyTorch 实现的 CNN-LSTM 网络对电池热管理系统 PID 控制做了预测优化。由此可见，PyTorch 在智能控制中可以取代深度学习工具箱的功能。若同时结合 Simulink 的图形化系统设计优势，可以实现在线的数据仿真实验平台，为智能控制实验验证带来极大的便利。

本文利用文献[8]中提出的核电站稳压器压力模型作为基础模型，提出了一种基于 TCP/IP 协议实现的通信框架。该实时通信框架能够将 Simulink 系统输出的仿真数据实时传递给外部由 PyTorch 实现的 LSTM 网络执行工况预测，并将预测结果反馈至 Simulink 系统，从而将神经网络逻辑从原 Simulink 仿真控制逻辑中解耦，形成数据流闭环。经过仿真实验，证实该通信框架能够高效地传输数据，在确保数据完整性的情况下，弥补了 PID 控制器信号传输延迟，有效地优化了传统 PID 的控制性能，并为 PID 控制的智能优化研究提供了具备实时数据感知与反馈的仿真平台。

本文第 1 部分在说明现有稳压器压力模型的基础上，探讨了进程间通信方式与方优化案的选型；基于 TCP/IP 的实时仿真通信框架实现在第 2 部分中呈现，并展示了基于该框架的稳压器模型所进行的智能优化控制仿真实验结果及其分析；全文在第 3 部分给出研究总结。

1 仿真模型改进

1.1. 稳压器压力模型改进

现有仿真平台为文献[8]提出的实验模型。为了验证本研究提出的实时通信框架的正确性，将现有实验模型中的智能模块替换为本文研究的实时通信模块，从而实现 Simulink 与 Python 程序间的数据通信。原有智能预测模块则解耦到外部 PyTorch 框架中的 Python 程序中实现，用于根据 Simulink 产生的实时工况数据反馈工况预测结果，由此对 Simulink 屏蔽智能模块具体逻辑，实现数据处理的功能分离，提升仿真实验中对于控制平台优化性能的测试。

1.2. 进程间通信方式的探究

为了实现仿真模型的数据实时传输与预测逻辑解耦，需要研究如何让 Python 程序与 Simulink 程序通信。当前常用的进程间通信（inter-process communication, IPC）方法包括：

（1）管道（Pipe）。如图 1 所示，管道是一种经典的 IPC 方式，允许两个相互关联的进程进行通信，如父进程与子进程；

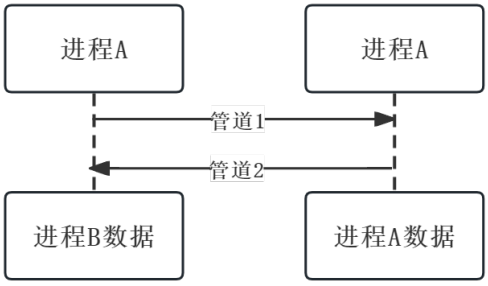

图 1 Pipe IPC 通信原理图

（2）消息队列（Message Queue）。如图 2 所示，消息队列通信将维护一个队列并对队列元素执行收发处理，能够用于跨多个平台的应用异步通信的情形[13]；

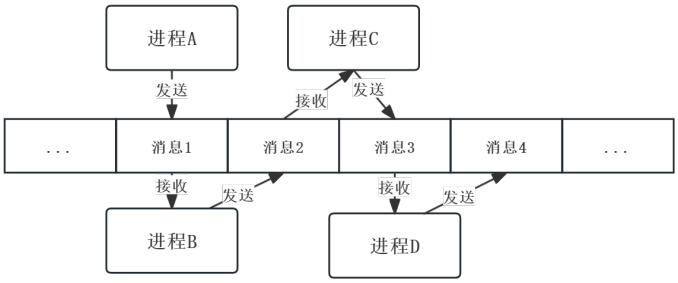

图 2 Message Queue IPC 通信原理图

（3）共享内存（Shared Memory）如图 3 所示，共享内存将一段物理内存映射到多个进程的地址空间中，通常用于需要交换大量数据同时需要保证效率的通信情形，行业内已开始利用多核和多核共享内存架构来提升处理器运算性能[14]；

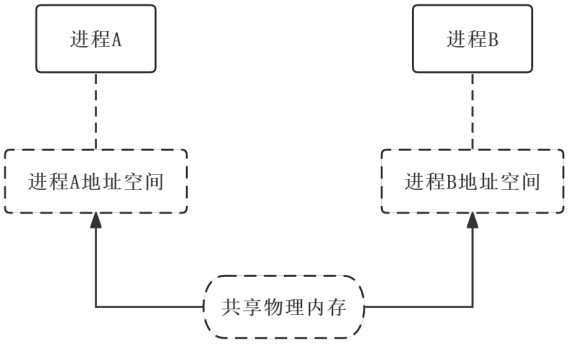

图 3 Shared Memory IPC 通信原理图

（4）Socket（套接字）。如图 4 所示，Socket 在网络编程中用于进程间通信，因而 Socket 编程也被称为网络编程。Socket 遵循网络协议进行通信，操作系统通常会提供原始的 Socket 应用程序接口，用户可以通过各种编程语言进行 Socket 编程。Socket 通信基于 C/S 架构，需要一个客户端（client）和服务端（server），分别绑定对应的网络地址和端口号执行通信。互联网的流行大大促进了 Socket 编程的应用，Socket 编程已在跨设备通信中扮演着重要的角色[15]；

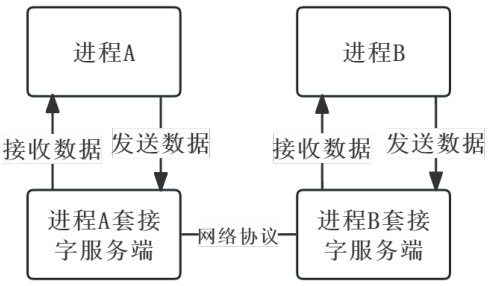

图 4 Socket IPC 通信原理图

对上述 IPC 方案进行选型需要考虑各种方案的特点以及本研究的实际需要。针对以上方案进行的思考包括：

1） 管道与共享内存在不同的编程框架下（本文是 Python 和 MATLAB）的实现存在差异，进程间通信协调难度较大，相关代码跨平台拓展性差；

2） 消息队列适合的异步场景不符合本文所述的实时通信框架的数据同步需求；

3） Socket 通信相对简单，实时性好，大多数编程语言提供的标准库均配有 Socket 支持，屏蔽了底层实现细节。构建 Socket 节点后，程序只需要处理数据接收和发送逻辑就可以进行通信。

4） Simulink 模拟仿真时采样数据为 Double 类型， PyTorch 框架与配套数据处理工具亦可处理 Double 类型数据。

综上所述，Socket 通信是本研究理想的 IPC 方案，最终具体模型构建方法为：

（1）Simulink 侧，在系统比较环节处增加 M 函数（MATLAB 语言编写）socket 模块，负责与外部 Python socket 的数据通信。

（2）Python 侧，利用 PyTorch 实现 LSTM 神经网络时序预测模块。通过对历史误差进行学习，预测接下来的误差，补偿 PID 控制；实现 socket 服务器模块，负责与外部 Simulink socket 的数据通信

构建流程如下图所示：

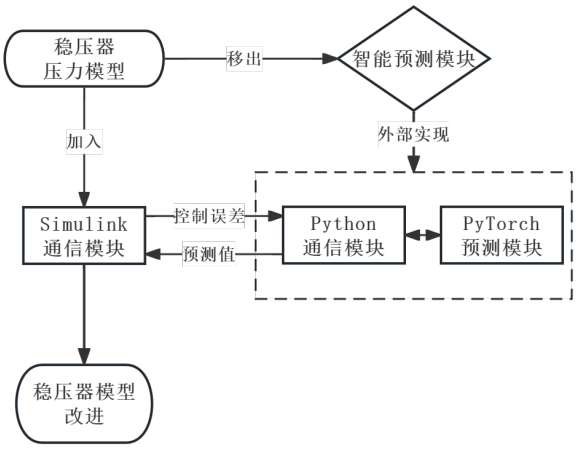

图 5 实时通信仿真平台构建流程

2 基于 TCP/IP 的通信模块实现与仿真

2.1. 基于 TCP/IP 协议的 socket 编程

传输控制协议 TCP 和网际协议 IP 包含在互联网传输协议套件（Internet Protocol Suite, IPS）中，属于 IPS 最早的两个核心协议，因而 IPS 通常也被称为 TCP/IP 协议。IPS 采用分层结构，自底向上包括四层：网络链接层、网络互连层、传输层和应用层。

对 Socket 进行编程时，封装程序的主要关注点为传输层协议，即 TCP 和 UDP（User Datagram Protocol, 用户数据报协议）协议。TCP 通过重传策略保障数据不丢失的可靠传输，符合本研究对数据传输的需求。

Python 通过 socket 库引入了对 socket 编程的支持，socket 类接受 type 参数用于确认该 socket 连接锁使用的协议类型，可选类型包括 SOCK_STREAM（TCP）和 SOCK_DGRAM（UDP）两类。Socket 库同时提供了 receive 和 send 方法对字节数据进行收、发处理。MATLAB 库提供 TCP、UDP 相关 M 函数，可用于创建对应的客户端、服务端。本研究使用 Python socket 库实现了 TCP 服务端，利用 MATLAB tcpclient 函数创建 TCP 客户端并整合到 Simulink 中。具体通信流程是：Python 程序从 TCP 端口接收字节数据，利用 numpy 提供的工具函数将数据转换成 np.double 类型，再封装为 PyTorch 所需的张量数据，交给 LSTM 网络进行预测；预测结果张量逆向转换为字节数组后，发回 Simulink TCP Client 模块作为系统补偿。工作原理图如下图所示：

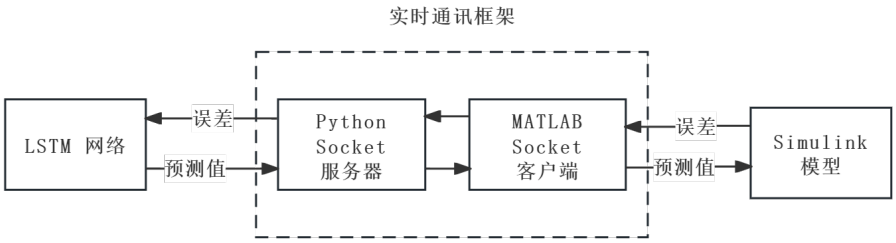

图 6 通信网络工作流程

为了解决同步问题，本文做了如下考虑和设计。Double 类型数据在 MATLAB 和 Python 中都占用 8 个字节的空间，接收数据时，需要考虑在接收到 8 个字节时就截断的情况；此外，也应保证缓冲区内不留存无效数据。因此，实验中 Python 程序在 TCP 服务端初始化时，设定好缓冲区大小为 8 字节，至多接收客户端传来的 8 字节数据；MATLAB 发送数据时将自动根据对侧缓冲区容量调整发送速度，暂未发送的数据保存在 MATLAB 程序缓冲区中，从而保证数据不丢失或者因速率不同步而产生无效数据。

2.2. LSTM 模型训练与 Simulink 模型搭建

LSTM 模型的数据集来源于传统 PID 控制仿真数据，模型包含一个堆叠 LSTM 层和一个全连接层，模型参数如表 1 所示。采用经典的 Adam 优化器执行 LSTM 训练迭代，Batch size 设置为 200，迭代次数 1000 次，学习率设置为 0.01。损失函数采用 MSE 均方误差函数。

表 1 LSTM 模型参数

主要参数	参数值
Epoch	1000
Batch Size	200
FC2 Units	1
LSTM Cell	8

LSTM 网络与 Python TCP/IP 服务端程序构成 Python 处理程序。

文献[8]中提出的两种加热器工况包括单组加热器工况（工况 1）和四组加热器工况（工况 4），其传递函数可分别定义为：

$$G(s)=\frac{0.004838}{(3400s+1)(100s+1)} \tag{2-1}$$

$$G(s)=\frac{1.776}{3300s^2+450s+1.34} \tag{2-2}$$

其中公式（2-1）代表单组加热器工况下稳压器阶跃响应函数，公式（2-2）代表四组加热器工况下稳压器阶跃响应函数。

在文献[8]使用的对比控制回路的基础上，将集成在 Simulink 中的 LSTM 智能控制模块替换为 TCP/IP 通信客户端模块，使智能预测逻辑与控制逻辑分离，形成 TCP/IP PID 控制器。最终形成的实验智能控制器回路示意图如下图 7 所示；具体回路实现如图 8 和图 9 所示。

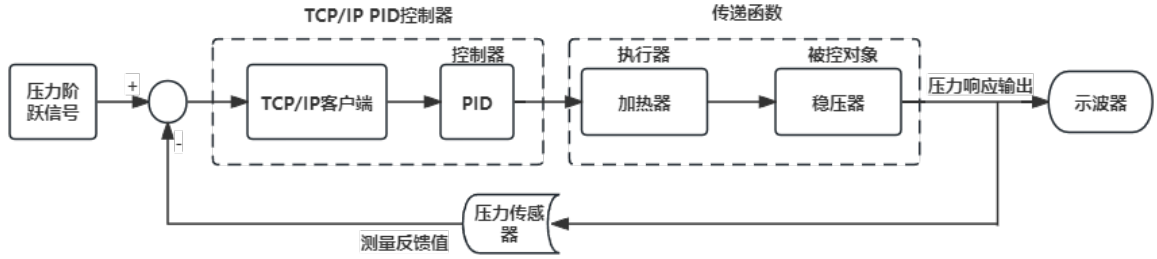

图 7 智能控制回路示意图

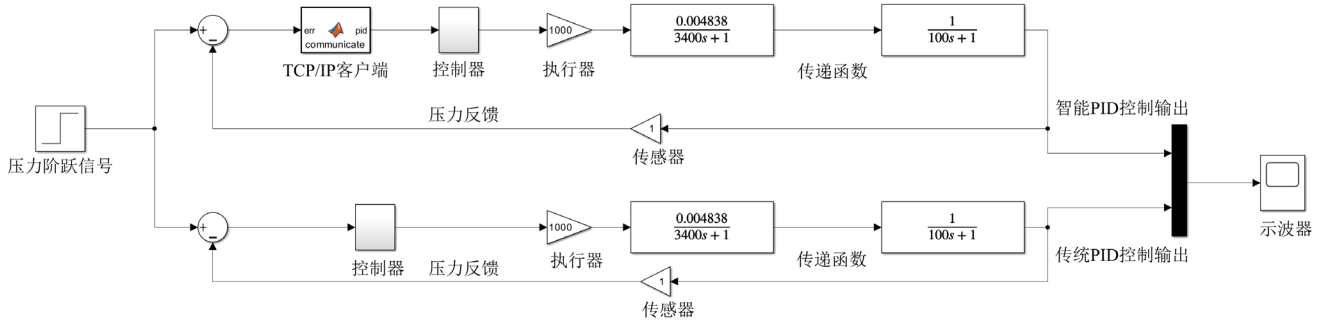

图 8 单组加热器压力控制对比回路

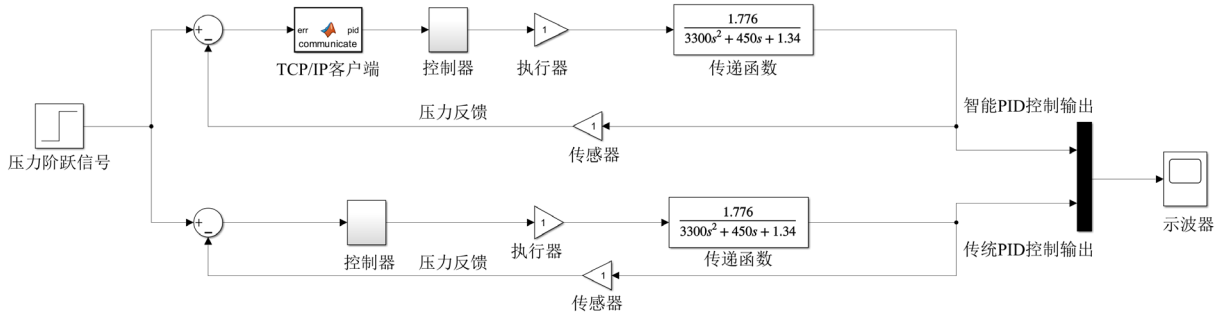

图 9 四组加热器压力控制对比回路

2.3. 实验结果分析

为验证本文提出的实时通信框架的功能，设计了系统模拟仿真实验，仿真相关参数如下表 2 所示。

表 2 Simulink 模型参数

各项参数	单组加热器	四组加热器
K_p	63.8	10
K_i	0.385	0.3
K_d	1000	50
执行器	1000	1
传感器	1	1

实验结果如图 10、图 11 所示。

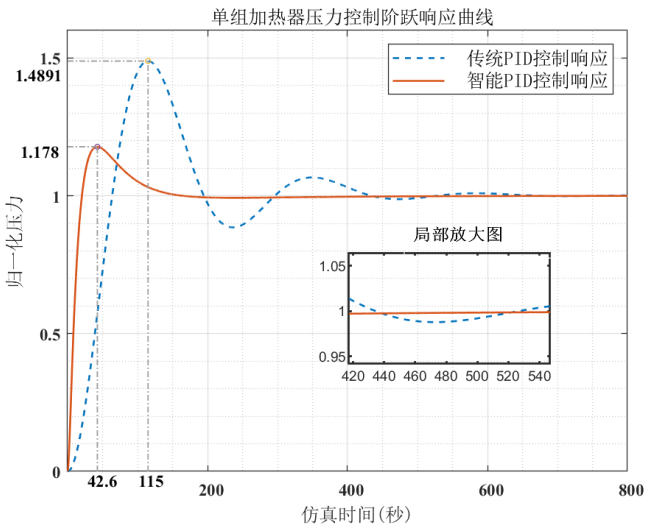

图 10 单组加热器压力控制阶跃响应曲线

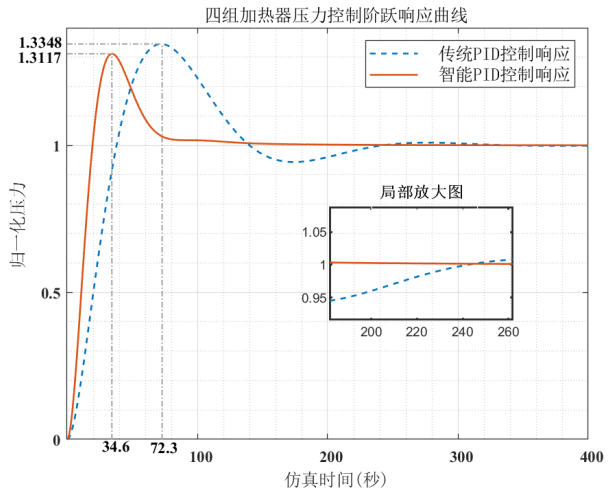

图 11 四组加热器压力控制阶跃响应曲线

PID 控制效果的衡量指标为超调量σ%、从响应为 0 开始到响应稳定在峰值阈值±5%所需调节时间t_s、从响应为 0 开始到第一次上升到调控阈值所需上升时间t_r和峰值时间t_p。从实验结果上看，智能 PID 控制相比于传统 PID 控制，超调量σ%有明显的降低，其中单组加热器工况超调量最大优化率为 63.60%；调节时间、上升时间和峰值时间均有所降低，其中单组加热器工况调节时间优化率为 73.35%，四组加热器工况调节时间优化率为 64.64%，上升时间、峰值时间优化率均超过 40%。这证明了本文提出的基于 TCP/IP 通信方法的实时框架能够较好地完成仿真过程中的数据交互工作，智能模块得以及时预测得到 PID 所需的输入误差，对原始 PID 带有响应延迟的输入进行了补偿，优化了各项相应指标，保证了智能 PID 优异的阶跃输入响应特性。智能 PID 与传统 PID 各项性能指标对比如下表 3 与表 4 所示。

表 3 单组加热器控制性能对比

性能参数	传统 PID 控制	智能 PID 控制	优化比例（%）
上升时间（s）	63.9	20.7	67.61
峰值时间（s）	115	42.6	62.96
调节时间（s）	381.25	101.25	73.35
超调量	0.489	0.178	63.60

表 4 四组加热器控制性能对比

性能参数	传统 PID 控制	智能 PID 控制	优化比例（%）
上升时间（s）	38.1	19.7	48.29
峰值时间（s）	72.3	34.6	52.14
调节时间（s）	189.31	66.94	64.64
超调量	0.345	0.312	9.57

依据文献[8]的研究内容，本文在功能验证的基础上设计了周期性的阶跃输入以验证智能 PID 在复杂工况下的控制效果，测试智能控制器在更复杂的工况下的可靠性和鲁棒性。整定值变化幅值采用+1、-0.5 和+0.3。控制响应变化曲线如下图 12、图 13 所示：

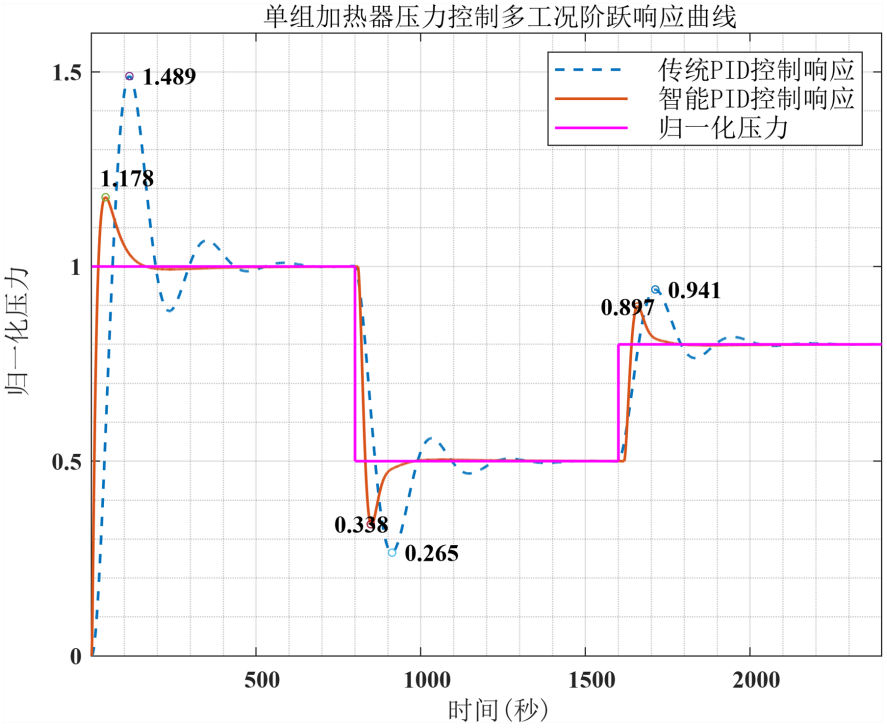

图 12 单组加热器压力控制多工况阶跃响应曲线

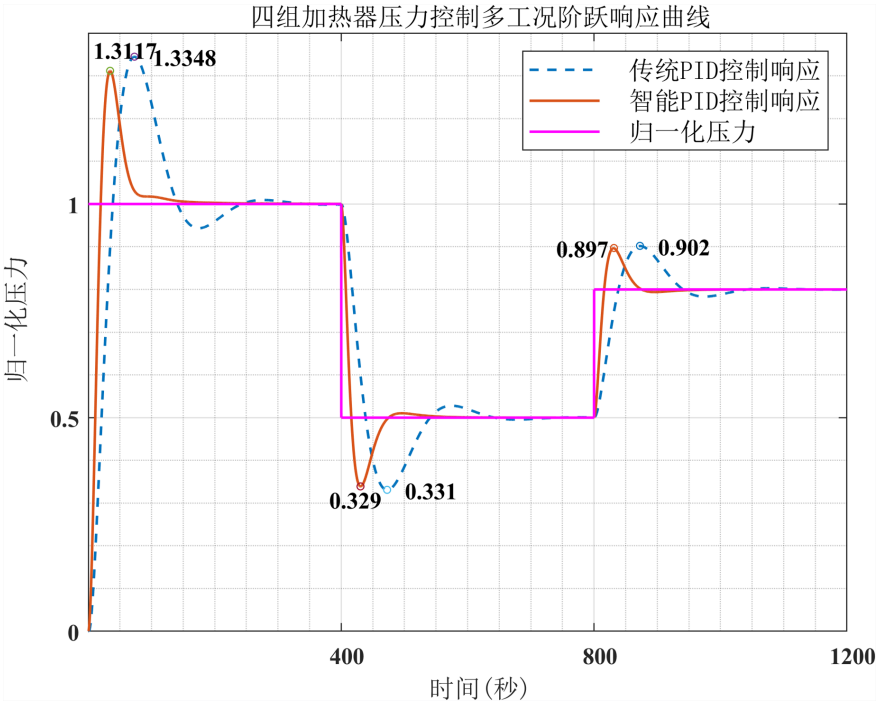

图 13 四组加热器压力控制多工况阶跃响应曲线

从多工况测试结果可以观察到，智能 PID 控制器在工况 1 和工况 4 下均具备比传统 PID 控制器更低的超调量、更短的上升时间和更优的响应特性。此外，不同阶段的调节变化显示出了相似的控制特征。传统 PID 控制在达到稳态之前可能会出现多次超调震荡，而智能 PID 控制则只有一次超调并且迅速达到稳态，表现出更高的可靠性和鲁棒性。对比两个工况的性能曲线还表明，工况 1 中由于只有一个加热器受控，智能 PID 控制的超调量较小但是收敛速度较慢；而在工况 4 中，智能 PID 控制的超调量相对较大，但能够在 200 秒内达到稳态。

为了研究神经网络模型预测效率与 TCP/IP 通信方案对智能仿真平台的影响，对不同整定区间下 PID 控制器给定控制量的平均时间间隔T_{pid}、PyTorch 模块从收到 Simulink 传递的数据到给出预测张量的平均时间间隔T_{py}和 TCP/IP 平均传输时间T_c进行测定，相关结果如表 5、表 6 所示。

表 5 单组加热器 T_{pid} T_{py} T_c 测定结果

整定阶段	$T_{pid}(ms)$	$T_{py}(ms)$	$T_c(ms)$
0~1	105.83	0.84	15.95
1~0.5	101.67	0.76	15.97
0.5~0.8	102.39	0.82	15.44

表 6 四组加热器 T_{pid} T_{py} T_c 测定结果

整定阶段	$T_{pid}(ms)$	$T_{py}(ms)$	$T_c(ms)$
0~1	73.32	0.82	15.94
1~0.5	74.85	0.87	15.47
0.5~0.8	69.12	0.79	14.95

表 5、表 6 的数据说明，神经网络和实时通信框架可以很好地适应整定目标的变化，不同工况、不同整定阶段中单次神经网络预测耗时稳定在 8ms 附近，单次 TCP/IP 数据传输时延T_c稳定在 15ms

左右。而 Simulink 求解器单次求解平均耗时T_{solve}可以由下式（2-3）得到：

$$T_{solve} = T_{pid} - 2 \times T_c \quad (2-3)$$

根据以上数据和公式计算，仿真主要的时间开销来源于 Simulink 求解器，本研究设计的 PyTorch 神经网络与 TCP/IP 通信框架具有较好的适应性与响应速度。

本文提出的方案解决了以往仿真实验中 Simulink 平台同时实现神经网络和传统控制模块而带来的逻辑高度耦合的问题，另一方面，Simulink 平台下实现复杂神经网络能力不足，以往的网络经过预训练后使用相同的数据再次进行仿真，这是一种离线预测的办法；而本文的方案不仅可以让神经网络执行实时的预测，后续也可以很方便地进行参数微调，实现网络升级迭代。

3 结论

本文针对当前神经网络控制仿真中存在的离线预测、模块高度耦合问题，提出了一种基于 TCP/IP 协议的实时通信框架。该框架将 MATLAB Simulink 仿真数据传输给外部 TCP/IP 端点，将不属于传统控制部分的智能模块抽离出来，实现系统解耦。此外，TCP/IP 客户端/服务端可以使用多种编程语言来实现，从而可以更便利地搭建智能控制仿真平台，灵活性也得到进一步的提升。经过实验验证，解耦后的稳压器控制智能仿真平台具有更高效、更精准以及更安全的控制性能。其中单组加热器工况超调量最大优化率为 63.60%；四组加热器工况仿真中上升时间、峰值时间优化率均超过 40%。通过多工况实验，验证了该智能仿真平台具有良好的稳定性与执行效率。

在仿真实验中，Simulink 仿真采样时不同求解器对时间步的解析存在一定差异，导致仿真数据存在不等距时间间隔。进行神经网络训练时，需要考虑不等距时间间隔的影响，利用符合要求的训练数据提升模型训练的收敛速度。同时也要考虑模型测试时客观存在的采样时间不等距问题，加强预测模型对于相关特征的学习。后续研究工作可以针对此类问题探索进一步提升预测精度的可能性。

4 致谢

本文作者感谢来自下列科研机构及科研项目的资金与技术支持：

1. 国家重点研发计划（2020YFB1713400），2020；
2. 中广核研究院有限公司；

参考文献

1. Sami, Farah.: Optimize Electric Automation Control Using Artificial Intelligence (AI). Optik 271, 170085 (2022)
2. Meng, D.Y.: Control Analysis and Synthesis of Data-Driven Learning for Uncertain Linear Systems. Automatica 148, 110734 (2023)
3. Norris, G., Guillaume, J.D., Christopher, O.: Neural Networks for Control: A Tutorial and Survey of Stability-Analysis Methods, Properties, and Discussions. 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), pp. 1-6. IEEE (2021)
4. Kuznetsova, T.A., Sukharev, A.A.: The Neural Network Controller for the Dry Low Emission Combustor of Gas-Turbine Power Plants. 2023 International Russian Smart Industry Conference (SmartIndustryCon), pp. 392-398. IEEE (2023)
5. Siraskar, R.: Reinforcement Learning for Control of Valves. Machine Learning with Applications 4, 100030 (2021)
6. Wan, N., Zeng, G., Zhang, C., Luo, Y.: Simulation of Vehicle ESP Based on Adaptive Fuzzy PID Control.12th International Conference on Advanced Computational Intelligence (ICACI), pp. 310-315. IEEE (2020)
7. Wu, X., Shi, X., Jia, J., Chen, Y., Li, X.: Expert System-Based EMI Modeling Method for IGBT in Electric Drive System of EV. IEEE Access 9, 42688-42696 (2021)
8. She, J.K., Li, W.Q., Ma, Y.F., Zhang, Y.F., Liu, L.: Pressurizer Control Optimization with Deep Learning-Based Predictions. International Symposium on Software Reliability, Industrial Safety, Cyber Security and Physical Protection for Nuclear Power Plant. Springer Nature Singapore, Singapore (2023)
9. Xu, P.Y.: Neural Network Based Self-Tuning PID Controller. 2022 2nd International Conference on Algorithms, High Performance Computing and Artificial Intelligence (AHPCAI), pp. 655-661. IEEE (2022)
10. Lee, M.F.R.: A Review on Intelligent Control Theory and Applications in Process Optimization and Smart Manufacturing. Processes 11.11, 3171 (2023)
11. Zhang, H.M., Wudhichai A., Yan, S.: Improved MM-MADRL Algorithm for Automatic Tuning of Multiparameter Control Systems. IEEE Access 10, 64729-64740 (2022)
12. Xie, J.H., Yang, R.F. Gooi H.B., Nguyen H.D.: PID-Based CNN-LSTM for Accuracy-Boosted Virtual Sensor in Battery Thermal Management System. Applied Energy 331, 120424 (2023)
13. Wei, Z.Q., Liu, J.W., Zhang, S.J., Hu, Z.T., Zhao, X.H.: An LSTM-based Method for Message Queue Throughput Prediction. Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI). pp. 1-6. ACM (2022)
14. Harel, R, Yuval, P., Gal, O.: Learning to Parallelize in a Shared-Memory Environment with Transformers. Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, pp. 451-452. ACM (2023)
15. Ozceylan, E., Cumali, Y., Baktygul, A.: Implementation of Socket Programming Simulation Using Quantum Communication Technologies. 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), pp. 1007-1012. IEEE, Turkey (2022)