基于GMDH神经网络的高炉炼铁智能控制

摘要

炼铁过程是一个离散加入，连续冶炼，离散输出的复杂生产过程。本文针对高炉炼铁的智能控制进行分析，探讨了铁水含硅量的影响因素和关于硅含量的动态预测控制方案，以及对铁水硫含量进行的优化。

问题一中本文针对题目提供的数据，通过查找资料发现$[Si]$的含量和之前炉的$[Si]$含量有密切关系，并且喷煤量对$[Si]$的影响具有时滞性，因此本文选取了6个自变量作为输入。考虑到输入变量之间的相关性以及高炉炼铁的复杂性，选取了基于数据分组处理（GMDH）算法的神经网络作为预测模型预测$[Si]$的含量。本文选取了1-699组数据作为学习样本，然后对数据进行预处理来训练网络。最后，将一步预测的$[Si]$ 含量作为模型新的输入，迭代求得二步预测的$[Si]$含量。

对于问题二，首先选取了700-1000组数据作为验证样本，分别计算其预测数值正确率和方向正确率，数值正确率达到81.4%。其中对于方向预测，由于大样本训练的网络方向预测结果不是很理想，所以尝试减少样本数量来进行预测。在样本量为30个时，网络的方向预测正确率最高，达到74.6%，但由于样本量较小，所以正确率有一定的波动，大致在60%-70% 左右波动。针对动态预测控制方案，本文首先通过相关性分析，度量鼓风量、喷煤量和$[Si]$含量之间的关系。由于$[Si]$含量和鼓风量、喷煤量是非线性关系，基于相关性分析假设其符合负相关，以简化模型。在动态预测控制方案中，本文采取线性比例控制，根据当前值、预测值和标准值的差值来线性变化步长，根据步长来修改下一炉的输入参数，以达到动态控制$[Si]$含量的目的。经过测试，该动态预测线性控制方案在初始值过大的情况下，在第20 步收敛，在标准值附近微小波动。

针对问题三，$[S]$的含量作为质量指标，含硫量低，铁水质量好。本文根据第一问采取的方法，以$[S]$的历史数据、硅的含量、鼓风量、喷煤量为自变量，建立了一个GMDH 神经网络，在绝对误差为0.004的条件下，正确率达到80%。以$[S]$ 的预测模型为基础，加入硅的预测值和动态预测控制方案，得到了关于$[S]$的优化模型。通过模拟结果可知，加入硅的动态预测控制后，$[S]$ 的含量均小于0.0018，结果表明硅的动态预测控制效果较好。

对于问题四，本题的高炉炼铁，其内部原理十分复杂，甚至会包括混沌系统，难以从机理上求得最优解，所以需要通过大数据的数据挖掘技术对过程进行优化。在复杂工业流程智能控制的大数据中，需要对数据进行预处理、去除噪声点，并根据数据量的大小规模选择合适的模型。

关键词：时间序列预测；数据分组处理；神经网络；动态预测控制

目录

1 问题重述

高炉炼铁过程是按一定顺序加入原燃料，由高炉下部连续鼓入热风、喷入煤粉进行炉温调整的冶炼过程，在该过程中需要时长对一些量进行检测并根据结果对炼铁条件进行调整以获得最优质的铁。从机理上求解该过程的最优解是十分复杂的，但通过大数据挖掘技术对过程进行优化是一种可取的方法。炼铁过程中人们会依据时间采集一个高维大数据时间序列数据，其中炉温（铁水含硅量$[Si]$）与铁的质量息息相关，因而对它的准确预测控制能为冶炼过程优化与预测控制提供依据和保障。

为了实现对高炉炼铁过程中预测控制的优化，本文将使用给定由铁水含硅量$[Si]$、含硫量$[S]$、喷煤量PML和鼓风量FL组成的关于时间序列的数据库作为基础进行数据挖掘，对以下四个问题进行解答：

1. 选取样本和算法，建立对$[Si]$的预测动态数学模型。
2. 对第1问建立的数学模型进行验证，包括对数值预测和炉温升降方向预测的成功率的计算，并说明动态预测控制的可行性。
3. 建立[S]的优化数学模型，并运用此模型验证依照$[Si]$对过程的预测控制的预期效果。
4. 谈谈大数据建模解决优化复杂流程工业智能控制问题的心得体会。

2 问题分析

- **问题1：**问题要求在给定的1000组数据中选取学习样本和算法，建立预测$[Si]$ 的动态数学模型。我们认为需要选取1-699 组数据作为学习样本，根据图表显示的结果以及高炉炼铁的复杂性选取合适的预测模型来预测$[Si]$的含量。然后将一步预测的硅含量作为模型新的输入，迭代得到二步预测结果，并简单分析模型的正确性。
- **问题2：**本文选取了700-1000组数据作为验证样本，验证训练好的模型预测数值准确率和方向准确率。对于炉温动态控制方案，本文认为预测控制需要先找出喷煤量、鼓风量和$[Si]$含量的相关性，然后通过调节下一时刻的喷煤量和鼓风量达到对$[Si]$ 含量的动态控制。最后，需要测试当硅含量异常时，预测控制方案是否能正常发挥作用，以此说明动态预测控制的可行性。
- **问题3：**对于建立质量指标$[S]$的优化模型，首先需要找出$[S]$和$[Si]$、喷煤量、鼓风量和$[S]$的历史数据之间的关系，然后沿用第一问$[Si]$ 的预测模型，建立关于$[S]$的优化模型。已知$[S]$ 是检测生铁质量优劣的质量指标，因此需要通过第二

问的预测控制方案来控制[Si]的含量，间接影响[S] 的含量，使[S]的含量在动态预测控制方案下均保持一个较低的水平。控制[S]的含量保持在较低水平，以此说明[Si] 预测控制方案的效果。

- **问题4：**问题要求讨论复杂流程工业智能控制大数据建模的心得体会，所以需要围绕使用的预测模型和动态预测控制方案，讨论使用模型的优缺点以及大数据建模遇到的问题及收获。

3 假设

为了排除一些因素对研究产生的不必要的影响，我们作出以下假设：

- 所给数据中含有的噪声皆为白噪声。
- 假设铁水中硫含量对铁水中硅含量无影响。
- 假设数据中的记录都真实可靠。

4 符号说明

符号	说明
$[Si]$	铁水含硅量（即炉温）
$[Si]_n$	当前时刻的铁水含硅量（即炉温）
$[Si]_{n-k}$	$k \in Z^+$，$2k$小时前的高炉铁水含硅量（即炉温）
$[Si]_{n+k}$	$k \in Z^+$，$2k$小时后的高炉铁水含硅量（即炉温）
$[Si]'_{n+k}$	$k \in Z^+$，$2k$小时后的高炉铁水含硅量（即炉温）的预测值
$[S]$	含硫量
$[S]_n$	当前时刻的铁水含硫量
$[S]'_{n+k}$	$k \in Z^+$，$2k$小时后的铁水含硫量的预测值
PML	喷煤量
PML_n	当前时刻的喷煤量
PML_{n-k}	$k \in Z^+$，$2k$小时前的喷煤量
PML_{n+k}	$k \in Z^+$，$2k$小时后的喷煤量
FL	鼓风量
FL_n	当前时刻的鼓风量
FL_{n-k}	$k \in Z^+$，$2k$小时前的鼓风量
FL_{n+k}	$k \in Z^+$，$2k$小时后的鼓风量
L_{PML}	当前喷煤量的变化步长
L_{FL}	当前鼓风量的变化步长
L'_{PML}	变化后的喷煤量的变化步长
L'_{FL}	变化后的鼓风量的变化步长

5 模型的建立与问题的求解

5.1 基于GMDH的神经网络的$[Si]$动态预测模型的建立

5.1.1 模型简介

炼铁过程是一个十分复杂的过程，影响因素众多且存在噪声影响，一般的数据挖掘技术不能收到很好的效果，我们选择使用基于GMDH 的神经网络来进行预测模型的建立。

自组织数据挖掘技术（Self-organize Data Mining，SODM），它从一个简单的初始模型集合出发，按照一定的法则进行组合，生成新的竞争模型，再筛选，重复这个过程直到最终获得最优复杂度模型 [1]。这种方法适用于从影响因素众多又互相关联的复杂系统进行数据挖掘，并用于预测 [2]。自组织数据挖掘可以导入外部准则（即已知的经验、知识或理论）来选择最优复杂度模型，从而提高准确度，这是它相对人工神经网络（ANN）的一个优势 [3]。数据分组处理方法（Group Method of Data Handling，GMDH）是SODM的核心技术。GMDH 算法将观测样本分为训练集和验证集，在训练集上利用内准则生成中间待选模型，在验证集上利用外准则进行中间待选模型的选留 [1]。

利用GMDH的神经网络与一般的前馈神经网络不同，它的结构确定于训练过程中，可大大提高网络的性能，能对数据进行很好的预测 [4]。

5.1.2 模型的建立过程

根据相关研究 [5]，我们按照以下步骤建立基于GMDH 的神经网络模型。

1. 数据集的划分

根据原理，首先将数据分为训练集和验证集。给定数据中有1000组数据，由于我们需要实现二步预测，我们仅使用999组数据，第1-699 炉的数据为训练集，700-999炉为验证集。

2. 自变量的选择和关系的建立

根据题意，炉温$[Si]$是预测和控制炼铁质量的关键指标，而影响炉温变化的因素也有很多。为了实现对炉温的预测，我们首先需要弄清这些影响因素，也就是用于预测的自变量。

炼铁是一个连续的过程，炉温在这个过程中会不断变化，根据相关研究，当前炉温的值不仅和当前炉内状态有关，也受到前几炉的炉温的影响，且因为炼铁的周期是连续的，所以当前周期的前几炉的炉温也会与上一周期的后几炉的炉温的影响 [7]，炉温在一个较长的时间内呈现周期性变化。$[Si]$的时滞性使用自相关函数计算得出为3 [6]，因此我们选取预测炉温$[Si]$的自变量为前3炉的炉温的炉温即$[Si]_{n-1}$、$[Si]_{n-2}$、$[Si]_{n-3}$。

考虑到炼铁过程的影响因素很多，如果只把单一的炉温$[Si]$作为时间序列进行预测建模，其预报命中率也不会理想 [2]，因此我们需要综合各种变量的影响。因此根据题意，喷煤量PML 和鼓风量FL 也是$[Si]$预测的影响因素。需要注意的是，根据相

关研究，喷煤量的影响具有时滞性，时滞约4个小时 [2]，因此我们考虑上上炉的喷煤量PML_{n-2} 作为炉温的影响因素。

综上，我们最终确定了5个用于炉温$[Si]$的预测的自变量，它们是：前3炉的炉温$[Si]_{n-1}$、$[Si]_{n-2}$、$[Si]_{n-3}$以及4小时前高炉喷煤量PML_{n-2} 和当前炉的鼓风量FL。

建立初始的因变量（输出）和自变量（输入）的一般关系，作为参考函数，一般常用K-G多项式进行表示 [8]：

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + ... \tag{5-1}$$

此处$m = 5$。

3. 外准则标准的建立

引入外准则可以让人的先验知识提高模型的准确度，从而提升数据挖掘的作用。因为高炉炉温预测控制具有特殊性，我们选择预测准则作为外准则标准 [9]：

$$\begin{aligned} l^2(A) &= \sum_{t\in C}(y_t - y_t^m(A))^2 \\ l^2(B) &= \sum_{t\in C}(y_t - y_t^m(B))^2 \\ l^2(W) &= \sum_{t\in W}(y_t - y_t^m(W))^2 \end{aligned} \tag{5-2}$$

这里的$y_t^m(A)$、$y_t - y_t^m(B)$和$y_t - y_t^m(W)$分别是在样本集A、B、W 上作的自回归模型估计。

4. 网络的训练和分析

用5个原始变量作为第一层的输入变量，传递函数选用二次函数形式，待选模型的产生原则选用线性最小二乘法，这样一直进行下去进行到外准则值不再减少，找出高炉炉温与自变量间的关系。

我们输入训练集数据对网络进行训练，训练结果如图 5-1 所示。

从图中可以看出，$MSE \approx 0.006$网络拟合的曲线和原始数据较为接近，虽然在极值点的拟合情况有一些偏差，但拟合曲线能够较好的反映数据的变化趋势。

导出网络第一阶段生成的变量，当$m = 5$时，自变量间的关系如表 5-1所示。

表 5-1中，一行代表一个变量，1表示有影响，0表示没有影响，这个新变量为有影

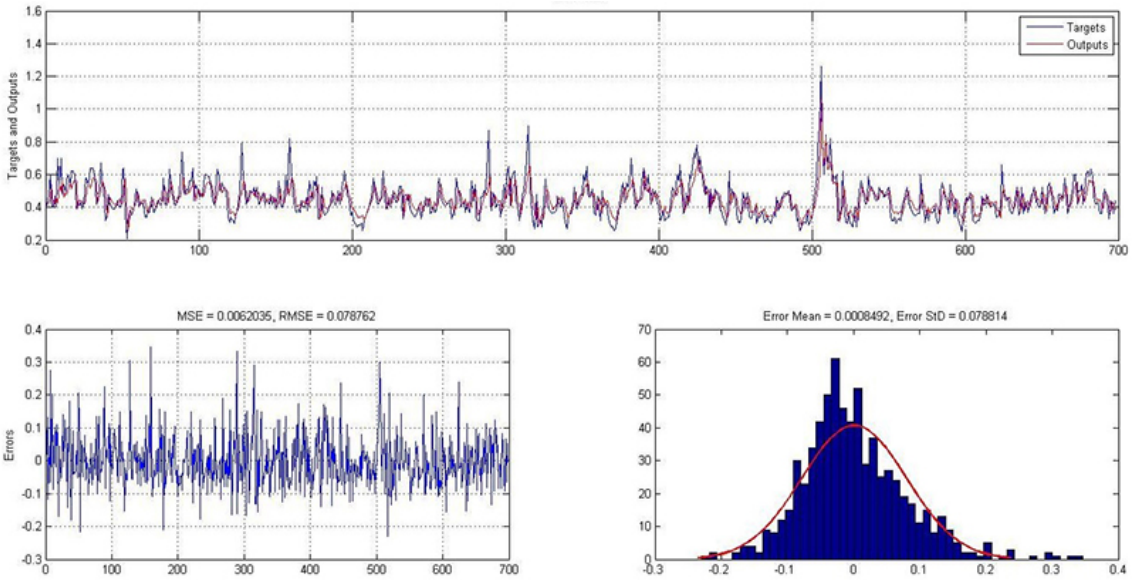

图 5-1: 网络训练效果图

$[Si]_{n-1}$	$[Si]_{n-2}$	$[Si]_{n-3}$	PML_{n-2}	FL
1	1	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
0	1	0	0	1
0	1	1	0	0
0	1	0	1	0

表 5-1: 自变量之间的相互关系

响的两个原始变量的乘积，我们一共找出7 个影响炉温的新变量：

$$\begin{aligned} v_1 &= [Si]_{n-1} * [Si]_{n-2} \\ v_2 &= [Si]_{n-1} * FL \\ v_3 &= [Si]_{n-1} * PML_{n-2} \\ v_4 &= [Si]_{n-1} * [Si]_{n-3} \\ v_5 &= [Si]_{n-2} * FL \\ v_6 &= [Si]_{n-2} * [Si]_{n-3} \\ v_7 &= [Si]_{n-2} * PML_{n-2} \end{aligned} \tag{5-3}$$

根据上述这些变量可以看出：

1. 炼铁过程各变量之间的相互关系较为明显，说明我们在分析高炉炼铁过程时，除了简单考虑炉温关于时间的变化，考虑其它变量对炉温的影响是必要的、不能忽略的；
2. 前几炉的炉温对于当前炉温的影响显著，因此历史炉温是现在炉温的重要指标，是我们需要考虑的因素，同时这一点也是非常符合实际情况的；
3. 前炉的炉温和喷煤量会影响当前炉的炉温，说明高炉具有滞后性，也与实际情况相符。

由上面的分析可得，基于自组织数据挖掘GMDH的神经网络在高炉炼铁上的初步应用具有成效。

5. 二步预测

根据相关研究 [7]，二步预测分为迭代一步预报和直接二步预报，为了方便计算，我们选择迭代一步预报作为这里二步预报的方法，将一步预测的硅含量作为模型新的输入，迭代得到二步预测结果。如图图 5-2为二步预测的结果图。

根据计算结果，在绝对误差为0.2的情况下，二步预测的正确率约为62%，相比一步预测，成功率有所下降。

5.2 预测成功率的计算和动态预测控制

5.2.1 预测成功率的计算

根据上一问所描述的方法，我们将网络建立并训练好，随后将样本容量为300 的验证集输入网络进行预测。结果见图 5-3。

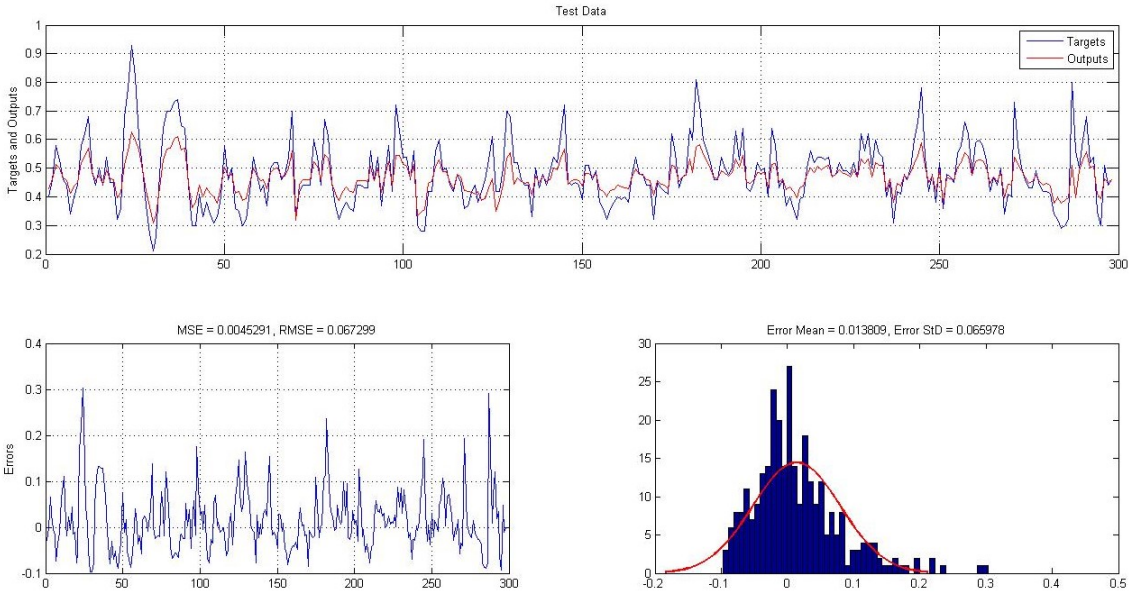

图 5-2: 二步预测效果图

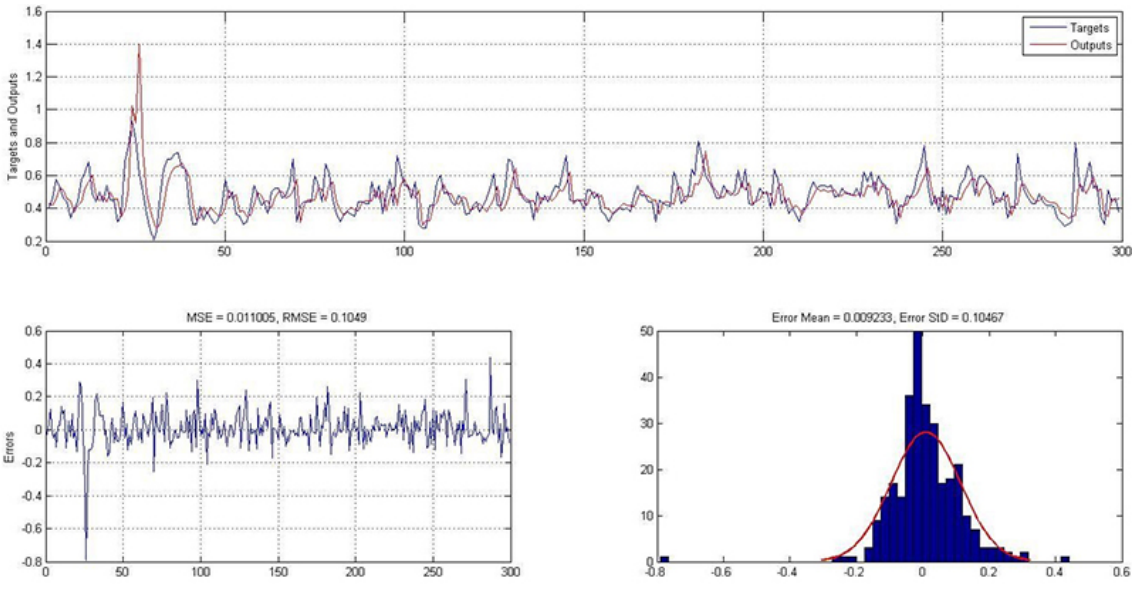

图 5-3: 网络测试效果图

根据图 5-3可以看出，拟合曲线真实地反映了数据的趋势，均方误差$MSE \approx 0.011$取值比较比较小。总体来说，拟合效果尚可。下面计算数值计算成功率和炉温升降方向预测成功率。

1. 数值预测成功率

$[Si]'_{n+1}$为下一炉的炉温预测值，$[Si]_{n+1}$为下一炉的炉温实际值，$\Delta[Si] = [Si]'_{n+1} - [Si]_{n+1}$，当$\Delta[Si] > 0.1$ 时认为是数值预测错误。经过计算，数值预测的平均成功率为81.2%。

2. 炉温升降方向预测成功率

设1代表炉温上升方向，0代表炉温下降方向，用后一时刻的炉温与前一时刻的炉温相减，若差为正数则标记为1，否则标记为0。分别计算真实值与预测值的“1-0序列”，并相比较，两者匹配则表示预测成功，否则失败。

实践发现，直接使用全部验证集进行方向预测的效果不尽人意。我们对模型进行改进，使用样本容量较小的样本作为输入，发现方向预测成功率有所提升，在样本容量取30 的时候，方向预测成功率达到最大值，在历次实验中大致在$60 \sim 70\%$ 之间波动。

5.2.2 依据$[Si]$的动态预测控制的可行性分析

我们的目的是根据$[Si]$的预测值自动控制生产过程中的变量的取值，从而提高生产出的铁的质量。具体想法是，确定炉温$[Si]$的标准值，将$[Si]$的预测值与标准值进行实时监控和比较，若发现预测值偏离标准值时，采取适当的方法降低炉温至标准值附近，从而将其控制在一定合理的范围内。具体步骤如下：

1. 相关性分析

因为我们将会在$[Si]$的预测值偏离标准值时将其调整至标准值附近，所以弄清如何通过喷煤量PML 和鼓风量FL 控制$[Si]$就十分必要。虽然炼铁是一个十分复杂的过程，量之间很可能不存在线性相关关系，但我们仍可以使用相关相关性分析看出它们的大致关系。结果见表 5-2。

由结果可以看出，它们之间具有弱负相关性，但无法通过显著性检验。这说明它们之间确实不是简单的线性相关关系。我们猜想：提升PML 和FL可以降低$[Si]$，减少PML和FL可以升高$[Si]$，并以此来建立我们的预测控制方案。

2. [Si]的标准值的求解

	PML	FL
Pearson相关性	-.022	-.018
显著性（双尾）	.489	.569
N	1000	1000

表 5-2: PML、FL与$[Si]$的相关性分析

标准值是我们判断$[Si]$的预测值是否合理的标准和依据。这个标准值是一个常数，在整个炼铁过程中，我们希望炉温的值不会大幅度波动，而是在该值附近上下波动。

求解标准值采用取平均的方法，即去除所给1000 组数据中[Si] 的最大5%和最小5%的数据样本点，然后求其平均值。求解出[Si]的标准值$[Si]_0$为：

$$[Si]_0 = 0.4533 \tag{5-4}$$

3. PML和FL的变化步长的设定

PML和FL的变化步长作为每次喷煤和鼓风的变化值的基准，根据具体情况的不同而动态改变步长的取值，实现PML和FL 的动态变化，从而实现动态预测控制。定义PML和FL的变化步长L_{PML} 和L_{FL}如下：

$$L_{PML} = \frac{1}{4}(max(PML) - min(PML)) \tag{5-5}$$

$$L_{FL} = \frac{1}{4}(max(FL) - min(FL)) \tag{5-6}$$

值得注意的是，这里步长定为最大值与最小值的差值的1/4 是合理的，但也是十分主观的。在随后的动态预测控制方案中，我们会动态变化步长的取值，因而步长初值取值的主观性问题会得到解决。

4. 预测控制方案设计

预测控制方案要实现根据$[Si]$的当前值和预测值的大小自动控制PML和FL的取值，从而控制$[Si]$保持在标准值的附近。设$[Si]_n$为当前时刻的炉温值，$[Si]'_{n+1}$为下一炉（2小时之后）的炉温预测值，$[Si]_0$为上文求解出的炉温标准值。具体方案如下：

①当$[Si]_n > [Si]_0$ 且$[Si]'_{n+1} > [Si]_0$时，步长L_{PML}和L_{FL} 变为原来的2倍，喷煤量PML和鼓风量FL等于原值加上步长。即

$$\begin{aligned} PML_{n+1} &= PML_n + L'_{PML} \\ &= PML_n + 2 * L_{PML} \end{aligned} \tag{5-7}$$

$$\begin{aligned} FL_{n+1} &= FL_n + L'_{FL} \\ &= FL_n + 2 * L_{FL} \end{aligned} \tag{5-8}$$

根据之前的相关性分析和猜想，喷煤量和鼓风量都与炉温负相关，所以当当前炉温和下一时刻炉温都大于标准值时，大幅度提高PML 和FL 的加入量，使$[Si]$能较快下降到标准值附近。

②当$[Si]_n < [Si]_0$ 且$[Si]'_{n+1} < [Si]_0$时，步长L_{PML}和L_{FL} 变为原来的2倍，喷煤量PML和鼓风量FL等于原值减去步长。即

$$\begin{aligned} PML_{n+1} &= PML_n - L'_{PML} \\ &= PML_n - 2 * L_{PML} \end{aligned} \tag{5-9}$$

$$\begin{aligned} FL_{n+1} &= FL_n - L'_{FL} \\ &= FL_n - 2 * L_{FL} \end{aligned} \tag{5-10}$$

这种情况下，大幅度降低PML 和FL的加入量，使$[Si]$能较快上升到标准值附近。

③当$[Si]_n > [Si]_0$ 且$[Si]'_{n+1} < [Si]_0$时，步长L_{PML}和L_{FL} 变为原来的1/3 倍，喷煤量PML和鼓风量FL等于原值减去步长。即

$$\begin{aligned} PML_{n+1} &= PML_n - L'_{PML} \\ &= PML_n - \frac{1}{3} * L_{PML} \end{aligned} \tag{5-11}$$

$$\begin{aligned} FL_{n+1} &= FL_n - L'_{FL} \\ &= FL_n - \frac{1}{3} * L_{FL} \end{aligned} \tag{5-12}$$

当前值大于标准值，而下一时刻的预测值已经低于标准值，说明下降得过快，这时应该稍减少PML和FL的加入量，使炉温小幅上升回到标准值附近。

④当$[Si]_n < [Si]_0$ 且$[Si]'_{n+1} > [Si]_0$时，步长L_{PML}和L_{FL} 变为原来的1/3 倍，喷煤量PML和鼓风量FL等于原值加上步长。即

$$\begin{aligned} PML_{n+1} &= PML_n + L'_{PML} \\ &= PML_n + \frac{1}{3} * L_{PML} \end{aligned} \tag{5-13}$$

$$\begin{aligned} FL_{n+1} &= FL_n + L'_{FL} \\ &= FL_n + \frac{1}{3} * L_{FL} \end{aligned} \tag{5-14}$$

当前值小于标准值，而下一时刻的预测值已经超过标准值，说明炉温上升得过快，这时应该稍增加PML和FL的加入量，使炉温小幅下降回到标准值附近。

5. 控制方案的效果检验

我们任取所给数据中的一段数据，这里取第1至90组，画出$[Si]$的变化折线图。同时，我们从第4个样本点开始运用上述动态预测控制方案，画出在控制状态下的$[Si]$变化折线图。这两条折线的变化情况如图 5-4所示。

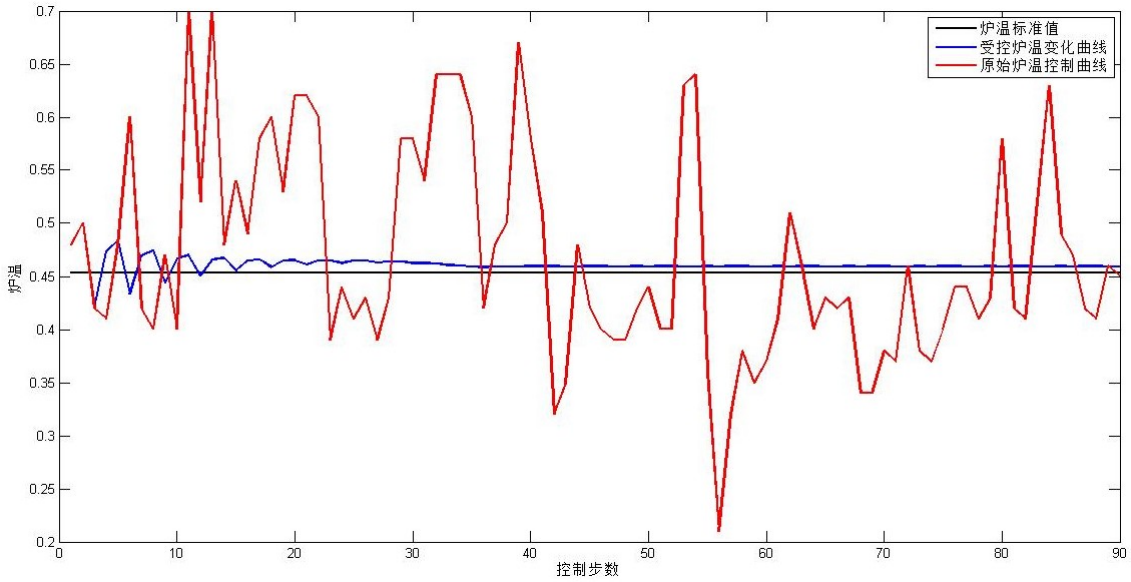

图 5-4: $[Si]$在运用和不运用动态控制方案情况下的变化情况示意图

由图像我们可以得到，该控制方案可以实现将炉温$[Si]$ 控制在标准值附近。

综上，我们通过监控和预测$[Si]$的变化情况，运用上面所提出的动态控制方案，通过动态控制PML 和FL的取值，成功实现将$[Si]$ 控制在标准值附近。因此，我们之前的猜想，即提升PML 和FL可以降低$[Si]$，减少PML和FL可以升高$[Si]$，$[Si]$的动态预测控制是可行的。

5.3 $[S]$优化模型的建立和$[Si]$动态预测控制效果的讨论

5.3.1 $[S]$优化模型的建立

1. 数据预处理

在高炉上采集的数据是多种多样的，它们的单位和数值不尽相同，为了排除数据的单位、取值范围对研究产生干扰，需要对输入的自变量进行归一化处理。考虑到因

变量S的量的取值也比较集中，我们也对它进行归一化处理。归一化使用的函数如下：

$$y = \frac{(y_{max} - y_{min})(x - x_{min})}{x_{max} - x_{min}} + y_{min} \tag{5-15}$$

与此同时，$[S]$的数据波动剧烈，噪声点多，这会对模型造成干扰，所以我们对$[S]$进行平滑化、去噪声。

2. 变量的选取和$[S]$预测模型的建立

为了找到含硫量$[S]$与其它量之间的关系，从而实现对$[S]$水平的预测，需要首先选择影响$[S]$的自变量。根据前文引证过的相关研究 [6] [7]，自变量由前两小时的含硫量$[S]_{n-1}$、$[S]_{n-2}$，由问题2中的$[Si]$的预测模型输出的炉温在每个时刻的预测值$[Si]'_n$，以及经$[Si]$的预测控制中修改后的每一时刻的PML_n、FL_n组成。因变量则为当前时刻的含硫量$[S]_n$。

随后我们使用同第2问的方法，即基于GMDH的神经网络，建立起S的预测模型。训练该网络时，使用归一化处理、$[S]$平滑去噪后的数据进行输入。训练的结果如图 5-5所示。

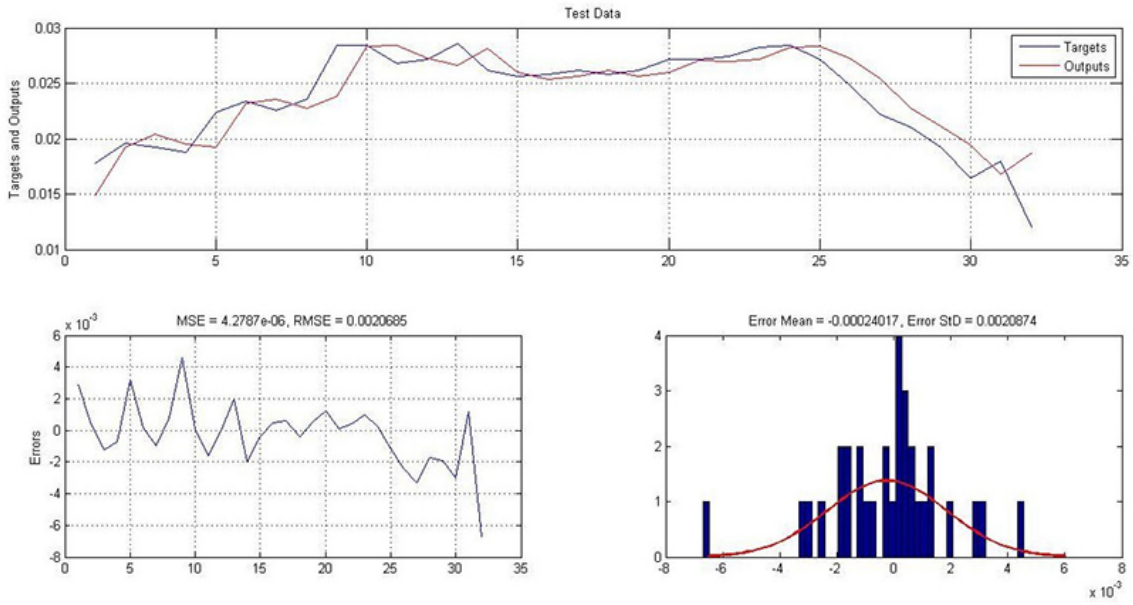

图 5-5: 预测模型验证结果图

可以看出，虽然在极值点，拟合值会产生一定偏差，但拟合曲线较好的反映了数值的变化趋势，$MSE \approx 4.28 * 10^{-6}$，因此该模型是具有实用性的。

5.3.2 $[Si]$动态预测控制效果的讨论

1. $[S]$的优化模型的建立

我们将第2问中建立的[Si]的动态控制和上文所建立的[S]的预测模型结合在一起，形成[S]的优化模型。该模型的输入为第2问中使用预测控制方案后的[Si]值作为优化模型的输入，得到在控制下的[Si]和众多其它因素影响下的[S] 的取值。

2. [Si]预测控制效果分析

优化模型的输出结果如图 5-6。

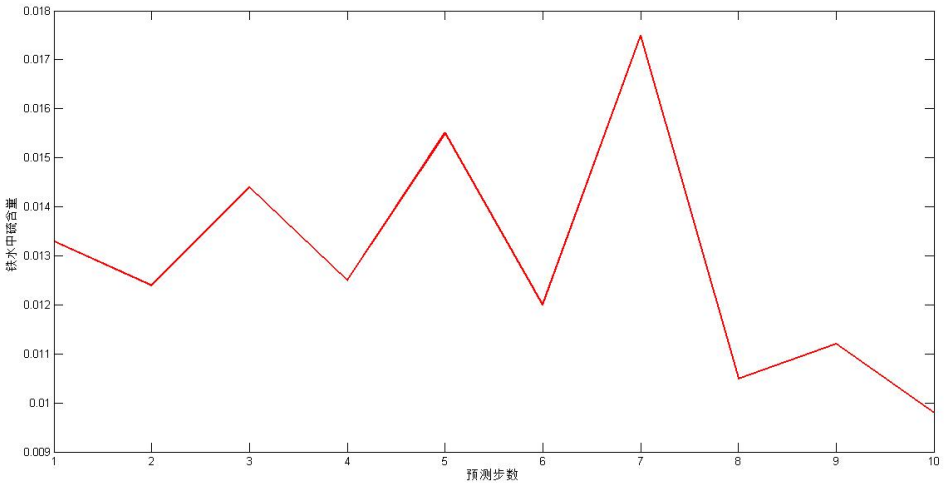

图 5-6: 优化模型效果图

从结果中我们发现，加上[Si]的动态预测方案后，[S]的含量均小于0.0018，结果表明硅的动态预测控制效果较好。

5.4 关于复杂流程工业智能控制大数据建模的心得体会

- 复杂流程工业，比如本题的高炉炼铁，其内部原理十分复杂，甚至会包括混沌系统，难以从机理上求得最优解，所以需要通过大数据的数据挖掘技术对过程进行优化。
- 大数据建模首先需要对数据进行一些简单的处理，例如归一化、标准化，对于有可能出错的数据还需要去除异常点。如果数据波动起伏较大或者建立的模型对于数据精度要求较高，则还需要对数据进行降噪。
- 对于大数据建模，数据量的大小规模有时候会决定所选用的模型效果是否表现良好，因此在选取模型之前，需要观察数据量的规模大小
- 在工业智能控制大数据建模中，训练的模型并非是精度越高越好，精度过高的模型可能存在过拟合，所以需要验证样本来检测模型的泛化能力

6 模型的优点与存在的问题

优点

本文第一问和第三问使用的基于GMDH算法的神经网络有较强的抗干扰能力，可以减少噪声的干扰，并且从一个简单的初始模型集合出发，按照先验知识构造的准则生成新的网络，适用于这种影响因素众多又相互关联的复杂系统，能对数据进行很好的预测。

第二问中使用的动态预测控制方案是一种比例线性控制，可以通过动态修改变化得步长来消除步长选择的主观性，并且能够更快的收敛。

存在的问题

本文存在的缺点是，题中提供的变量个数较少，在网络训练中会对模型的精度存在一定的限制，若存在更多的变量，网络训练的效果会更好、更加贴近实际。

参考文献

[1] 张宾, 贺昌政. 自组织数据挖掘方法研究综述[J]. 哈尔滨工业大学学报, 2006, 38(10):1719-1723.

[2] 贺诗波. 自组织数据挖掘在高炉炉温预测控制中的应用[D]. 浙江大学理学院 浙江大学, 2008.

[3] 贺昌政, 张宾, 俞海. 自组织数据挖掘与人工神经网络方法比较研究[J]. 系统工程理论与实践, 2002, 22(11):11-14.

[4] 肖婷, 黄国建. GMDH神经网络在数据预测中的应用[J]. 江苏科技大学学报自然科学版, 2000, 14(4):72-76.

[5] 陈建平, 杨宜民, 张会章,等. 一种基于GMDH模型的神经网络学习算法[J]. 云南大学学报自然科学版, 2008, 30(6):569-574.

[6] 郜传厚, 渐令, 陈积明,等. 复杂高炉炼铁过程的数据驱动建模及预测算法[J]. 自动化学报, 2009, 35(6):725-730.

[7] 徐雅娜. 高炉铁水含硅量及铁水含硫量实时预报专家系统的设计与实现[D]. 东北大学, 2001.

[8] Müller J A. GMDH algorithms for complex systems modelling[J]. Mathematical & Computer Modelling of Dynamical Systems, 1998, 4(4):275-316.

[9] 刘光中, 王綾. 自组织方法(GMDH)中准则的抗干扰性[J]. 系统工程理论与实践, 1995, 15(11):1-15.

附录

以下代码使用Matlab 2013b进行编写和运行。

$[Si]$的预测模型

主函数

```
clc;
clear;
close all;

%% Load Data
step=2;%一步或者二步预测
data = load('d');
x1 = data.d; %的数据si
x = x1(:,1:end-1);
x2 = x1(:,2:end);
other1 = load('variables');
y1 = other1.variables; %煤的数据
y = y1(:,1:end-1);
y2 = y1(:,2:end);
other2 = load('variable');
z1 = other2.variable; %鼓风量的数据
z = z1(:,1:end-1);
z2 = z1(:,2:end);

Delays = [1 2 3];
%Delays = [1 2];
m=max(Delays);
[Inputs, Targets] = CreateTimeSeriesData(x,Delays); %创建的历史时间序列数据（为siInputs3的矩阵，每次计算按一列输入，即一次输入个*9973的数据si)
Inputs = [Inputs;y(:,m-1:size(x,2)-2);z(:,m+1:size(x,2))]; %喷煤量具有滞后性（假定为个小时4)

```

```
26
27 nData = size(Inputs,2);
28 Perm = 1:nData;
29
30 % Train Data
31 pTrain = 0.7; %训练集划分
32 nTrainData = round(pTrain*nData);
33 TrainInd = Perm(1:nTrainData); %选择列数
34 TrainInputs = Inputs(:,TrainInd);
35 TrainTargets = Targets(:,TrainInd);
36
37 % Test Data
38 pTest = 1 - pTrain; %测试集划分
39 nTestData = nData - nTrainData;
40 TestInd = Perm(nTrainData+1:end); %选择剩下的列数
41 TestInputs = Inputs(:,TestInd);
42 TestTargets = Targets(:,TestInd);
43
44 %% Create and Train GMDH Network
45
46 params.MaxLayerNeurons = 25;    % Maximum Number of Neurons in a
      Layer
47 params.MaxLayers = 5;           % Maximum Number of Layers
48 params.alpha = 0.2;               % Selection Pressure
49 params.pTrain = 0.7;            % Train Ratio
50 gmdh = GMDH(params, TrainInputs, TrainTargets); %建立神经网络GMDH
51
52 %% Evaluate GMDH Network
53 %一步预测
54 if step == 1
55    Outputs = ApplyGMDH(gmdh, Inputs); %神经网络模拟计算结果
56    TrainOutputs = Outputs(:,TrainInd);
57    TestOutputs = Outputs(:,TestInd);
58    Outputs1 = Outputs;
```

```
59    TrainOutputs1 = TrainOutputs;
60    TestOutputs1 = TestOutputs;
61  else if step == 2
62  %二步预测
63          Outputs = ApplyGMDH(gmdh, Inputs);
64          Targets = x2(:,m+1:size(x2,2));
65          x2(:,m:end-1) = Outputs;
66          [Inputs, default] = CreateTimeSeriesData(x2,Delays);
67          Inputs = [Inputs;y2(:,m-1:size(x2,2)-2);z2(:,m+1:size(x2
                ,2))];
68          TrainTargets = Targets(:,TrainInd);
69          TestTargets = Targets(:,TestInd);
70          Outputs = ApplyGMDH(gmdh, Inputs);
71          TrainOutputs = Outputs(:,TrainInd);
72          TestOutputs = Outputs(:,TestInd);
73      end
74  end
75
76  %% Show Results
77
78  figure;
79  PlotResults(TrainTargets, TrainOutputs, 'Train Data');
80
81  figure;
82  PlotResults(TestTargets, TestOutputs, 'Test Data');
83
84  figure;
85  PlotResults(Targets, Outputs, 'All Data');
86
87  if ~isempty(which('plotregression'))
88      figure;
89      plotregression(TrainTargets, TrainOutputs, 'Train Data', ...
90                     TestTargets, TestOutputs, 'TestData', ...
91                     Targets, Outputs, 'All Data');
```

```
92
93 end
```

创建时间序列数据

```
1  function [X, Y] = CreateTimeSeriesData(x, Delays)
2
3      T = size(x,2);
4
5      MaxDelay = max(Delays);
6
7      Range = MaxDelay+1:T;
8
9      X= [];
10     for d = Delays
11         X=[X; x(:,Range-d)];
12     end
13
14     Y = x(:,Range);
15
16 end
```

创建GMDH网络

```
1  function gmdh = GMDH(params, X, Y)
2
3      MaxLayerNeurons = params.MaxLayerNeurons;
4      MaxLayers = params.MaxLayers;
5      alpha = params.alpha;
6
7      nData = size(X,2);
8
9      % Shuffle Data
10     Permutation = randperm(nData); %打乱顺序，消除每列之间的联系
```

```
11      X = X(:,Permutation);
12      Y = Y(:,Permutation);
13
14      % Divide Data
15      pTrainData = params.pTrain;
16      nTrainData = round(pTrainData*nData);
17      X1 = X(:,1:nTrainData);
18      Y1 = Y(:,1:nTrainData);
19      pTestData = 1-pTrainData;
20      nTestData = nData - nTrainData;
21      X2 = X(:,nTrainData+1:end);
22      Y2 = Y(:,nTrainData+1:end);
23
24      Layers = cell(MaxLayers, 1);
25
26      Z1 = X1;
27      Z2 = X2;
28
29    for l = 1:MaxLayers
30
31          L = GetPolynomialLayer(Z1, Y1, Z2, Y2);
32
33          ec = alpha*L(1).RMSE2 + (1-alpha)*L(end).RMSE2; %竞争压力
                筛选
34          ec = max(ec, L(1).RMSE2); %求最小误差
35          L = L([L.RMSE2] <= ec); %消去不符合的神经元
36
37          if numel(L) > MaxLayerNeurons
38              L = L(1:MaxLayerNeurons);
39          end
40
41          if l==MaxLayers && numel(L)>1
42              L = L(1);
43          end
44          if l==1
```

```
45          for i=1:numel(L)
46              L(i).i
47              L(i).j
48          end
49          end
50          Layers{l} = L;
51
52          Z1 = reshape([L.Y1hat],nTrainData,[])';
53          Z2 = reshape([L.Y2hat],nTestData,[])';
54
55          disp(['Layer ' num2str(l) ': Neurons = ' num2str(numel(L
                ))', Min Error = ' num2str(L(1).RMSE2)]);
56
57          if numel(L)==1
58              break;
59          end
60
61      end
62
63       Layers = Layers(1:l);
64
65       gmdh.Layers = Layers;
66
67  end
```

获得网络隐藏层

```
1  function L = GetPolynomialLayer(X1, Y1, X2, Y2)
2
3      n = size(X1,1);
4
5      N = n*(n-1)/2; %第m层所需的神经元的需要为前+1层神经元每两个计算一
           次，所以需要mm*(m-1)个/2
6
```

```
7       template = FitPolynomial(rand(2,3),rand(1,3),rand(2,3),rand
            (1,3),[],0,0); %生成多项式 K-多项
            式G
8
9       L = repmat(template, N, 1); %重叠生成第层的所有神经元m
10
11      k = 0;
12      for i=1:n-1
13          for j=i+1:n
14              k = k+1;
15              L(k) = FitPolynomial(X1([i j],:), Y1, X2([i j],:),
                    Y2, [i j],i,j); %带入数据计算神经
                    元
16          end
17      end
18
19      [~, SortOrder] = sort([L.RMSE2]); %根据测试集的排序RMSE
20
21      L = L(SortOrder);
22
23  end
```

计算某一神经元

```
1   function p = FitPolynomial(x1, Y1, x2, Y2, vars,i,j)
2
3       X1 = CreateRegressorsMatrix(x1);
4       c = Y1*pinv(X1);
5
6       Y1hat = c*X1;
7       e1 = Y1- Y1hat;
8       MSE1 = mean(e1.^2);
9       RMSE1 = sqrt(MSE1);
10
11      f = @(x) c*CreateRegressorsMatrix(x);
```

```
12
13      Y2hat = f(x2);
14      e2 = Y2- Y2hat;
15      MSE2 = mean(e2.^2);
16      RMSE2 = sqrt(MSE2);
17
18      p.vars = vars;
19      p.c = c;
20      p.f = f;
21      p.Y1hat = Y1hat;
22      p.MSE1 = MSE1;
23      p.RMSE1 = RMSE1;
24      p.Y2hat = Y2hat;
25      p.MSE2 = MSE2;
26      p.RMSE2 = RMSE2;
27      p.i=i;
28      p.j=j;
29
30  end
31
32  function X = CreateRegressorsMatrix(x)
33
34      X = [ones(1,size(x,2))
35           x(1,:)
36           x(2,:)
37           x(1,:).^2
38           x(2,:).^2
39           x(1,:).*x(2,:)];
40
41  end
```

使用GMDH网络模拟

```
1  function Yhat = ApplyGMDH(gmdh, X)
```

```
2
3       nLayer = numel(gmdh.Layers);
4
5       Z = X;
6       for l=1:nLayer
7           Z = GetLayerOutput(gmdh.Layers{l}, Z);
8       end
9       Yhat = Z;
10
11  end
12
13  function Z = GetLayerOutput(L, X)
14
15      m = size(X,2);
16      N = numel(L);
17      Z = zeros(N,m);
18
19      for k=1:N
20          vars = L(k).vars;
21          x = X(vars,:);
22          Z(k,:) = L(k).f(x);
23      end
24
25  end
```

画图表示结果

```
1  %
2  % Copyright (c) 2015, Yarpiz (www.yarpiz.com)
3  % All rights reserved. Please read the "license.txt" for license
       terms.
4  %
5  % Project Code: YPML120
6  % Project Title: Time-Series Prediction using GMDH
```

```
7  % Publisher: Yarpiz (www.yarpiz.com)
8  %
9  % Developer: S. Mostapha Kalami Heris (Member of Yarpiz Team)
10 %
11 % Contact Info: sm.kalami@gmail.com, info@yarpiz.com
12 %
13
14 function PlotResults(Targets, Outputs, Title)
15
16     Errors = Targets - Outputs;
17     m = size(Targets,2);
18     n=0;
19     t=0.1;
20     for i=1:m
21         if abs(Errors(i))<=t
22             n=n+1;
23         end
24     end
25     sprintf('在误差为的情况下，正确率为0.1%.33f%%',100*n/m)
26     MSE = mean(Errors.^2);
27     RMSE = sqrt(MSE);
28     ErrorMean = mean(Errors);
29     ErrorStd = std(Errors);
30
31     subplot(2,2,[1 2]);
32     plot(Targets);
33     hold on;
34     plot(Outputs,'r');
35     legend('Targets','Outputs');
36     ylabel('Targets and Outputs');
37     grid on;
38     title(Title);
39
40     subplot(2,2,3);
```

```
41      plot(Errors);
42      title(['MSE = ' num2str(MSE) ', RMSE = ' num2str(RMSE)]);
43      ylabel('Errors');
44      grid on;
45
46      subplot(2,2,4);
47      histfit(Errors, 50);
48      title(['Error Mean = ' num2str(ErrorMean) ', Error StD = '
            num2str(ErrorStd)]);
49
50  end
```

$[Si]$动态预测控制

```
1   x_initial =x(:,1:3)';%初始化第一次预测的输入数据
2   y_initial =y(:,2);
3   z_initial =z(:,4);
4   inputs=[x_initial;y_initial;z_initial];%整合初始输入数据成为一个时间
        点
5   control_out=zeros(1,900);%控制输出结果
6   control_out(1,1:3)=x_initial';%记录初始阶段的三个值
7   Sistandard=mean(Targets);%设置标准值
8   pmpath=(max(x)-min(x))/5
9   flpath=(max(y)-min(y))/5%设置步长
10  pinputs=[control_out(1,3);y_initial;z_initial];%从第三时刻的值开始
11  for i=4:900
12      flag=0;
13      control_out(1,i)=ApplyGMDH(gmdh,inputs);%应用已经建立好的模型进
            行预测GMDH
14      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15      %判断当前值以及预测值的情况并更改输入数据实现对硅含量的控制%%%
16      if control_out(1,i-1) > Sistandard && control_out(1,i) >
            Sistandard
17          y_inputs=inputs(4,1)+2*pmpath;
18          z_inputs=inputs(5,1)+2*flpath;
```

```
19      elseif control_out(1,i) > Sistandard && control_out(1,i-1) <
            Sistandard
20          y_inputs=inputs(4,1)+pmpath/3;
21          z_inputs=inputs(5,1)+flpath/3;
22      end
23      if control_out(1,i) < Sistandard && control_out(1,i-1) <
            Sistandard
24          y_inputs=inputs(4,1)-pmpath*2;
25          z_inputs=inputs(5,1)-flpath*2;
26      elseif control_out(1,i) < Sistandard && control_out(1,i-1) >
            Sistandard
27          y_inputs=inputs(4,1)-pmpath/3;
28          z_inputs=inputs(5,1)-flpath/3;
29      end
30      %根据情况调整下一时刻的喷煤量和风量的输入
31      x_inputs=control_out(1,i-2:i)';
32      inputs=[x_inputs;y_inputs;z_inputs]
33      pinputs(:,i-2)=[control_out(1,i-1);y_inputs;z_inputs]%记录下
            响应的结果一便硫浓度的预测,
34  end
```

[S]优化模型

```
1   x_initial =x(:,2:3)';%初始阶段
2   inputs=[x_initial;pinputs(:,1)];%初始输入为控制硅含量时记录下的结果
        （包含硅浓度、喷煤量、风量）pinputs
3   prid_out=zeros(1,100);%输出结果保存在此
4   prid_out(1,1:2)=x_initial;%选取硫的两个浓度作为初始硫浓度，以便预测下一
        时刻的硫浓度
5   prid_out(1,3)=ApplyGMDH(gmdh,inputs);
6   for i=2:10%对硫浓度进行十步预测
7       inputs=[prid_out(1,i:i+1)';pinputs(:,i)];
8       prid_out(1,i+2)=ApplyGMDH(gmdh,inputs);
9   end
```