可容错的遥控水下机器人递归神经网络控制

姜述强[1], 金鸿章[1], 魏凤梅[2,3]

(1. 哈尔滨工程大学 自动化学院, 150001 哈尔滨; 2. 哈尔滨工程大学 计算机科学与技术学院, 150001 哈尔滨;
3. 哈尔滨学院 工学院, 150086 哈尔滨)

摘 要: 针对遥控水下机器人(ROV)需要长时间稳定可靠工作的问题,提出递归模糊神经网络及可容错分配推力的控制方法.使用扩展函数链改进递归模糊神经网络控制器,提高了控制器对机器人非线性特性的识别和处理能力;基于反向梯度传播原理,由能量函数设计了该网络的学习算法,并根据微粒群优化确定学习率参数,从而保证整个网络的收敛性;在推力分配方面,针对开架式遥控水下机器人的两种推力器布置形式进行建模,将容错问题转化为对偶优化问题,建立能量函数实现故障条件下的推力优化分配.实验结果表明,所设计控制器不仅增强了遥控水下机器人对干扰的反应能力,并且提高了对机器人非线性特性的控制能力,减少了控制误差.当部分主推或侧推等推力器失效时,仍可以通过推力优化分配实现机器人在水平面上的准确位置控制,从而保证了遥控水下机器人长时间可靠工作.

关键词: 遥控水下机器人;递归神经网络;扩展函数链;推力分配;容错控制

中图分类号: TP24 **文献标志码:** A **文章编号:** 0367-6234(2013)09-0057-07

A fault-tolerable recurrent neural network controller for remote operated vehicle

JIANG Shuqiang[1], JIN Hongzhang[1], WEI Fengmei[2,3]

(1. College of Automation, Harbin Engineering University, 150001 Harbin, China; 2. College of Computer Science and Technology, Harbin Engineering University, 150001 Harbin, China; 3. School of Technology, Harbin University, 150086 Harbin, China)

Abstract: A fuzzy recurrent neural network controller with fault tolerable thrust allocation control strategy has been proposed for Remote Operated Vehicles (ROV). Extended functional link is issued for fuzzy recurrent neural network controller to improve the ability of identification and response. Online training algorithm is developed based on gradient descent method. The learning rate parameters are determined according to particle swarm optimization, hence the whole network convergence is guaranteed. On the aspect of force allocation, a model has been established according to thruster positions for open frame remote operated vehicles based on which the fault tolerable problem has been transformed into dual optimization problem. The energy function concerning ROV control has been established so as to obtain thrust optimal allocation under fault conditions. Experiments have demonstrated that the controller can improve the ROV capacity to handle nonlinear characteristics and strong disturbance, reduce control errors. When one of main or side thrust fails, the controller can still accomplish precise horizontal position control through thrust optimization allocation strategy. Therefore, the reliable operation of ROV for the long time-span has been improved.

Key words: ROV; recurrent neural network; Extended functional link; duality principle; thrust allocation; fault tolerance control

遥控水下机器人(ROV)在科学考察、海洋工程的巡检、深海救援等领域有广泛的应用[1-2].保证ROV长时间可靠工作,就必须研究推力器的容错控制.在水下机器人的控制方面,神经网络(NN)控制通过在线学习和训练,可较大地提高水下机器人的抗扰能力[3].但由于其训练和学习的时间相对较长,对外界环境的剧烈变化反应滞后[4],从而容易造成控制结果振荡.而基于类人思维的高层推理的模糊方法与神经网络相结合组成的模糊神经网络

收稿日期: 2012-12-11.
基金项目: 国家自然科学基金资助项目(51279039, 51209050);高等学校博士学科点专项科研基金(20122304120003).
作者简介: 姜述强(1978—),男,助理研究员,博士研究生;
金鸿章(1946—),男,教授,博士生导师.
通信作者: Jiangshuqiang@hrbeu.edu.cn.

(FNN)控制器广泛地应用于国内外水下机器人的运动控制[5-6],但其对于存在非线性特性的机器人控制效果并不理想. 和反向传播的神经网络相比,递归神经网络(RNN)对环境的动态变化具有更强的处理能力[7]. 在水下机器人的容错控制方面,主动容错控制由于较小地依赖系统的运动控制器得到了广泛的研究和使用[8]:其中文献[9]基于 H_∞ 容错控制方法,分析了螺旋桨的状态反馈;文献[10]和[11]通过分析 ROV 的推进器布置形式,提出了容错控制策略;但对于 ROV 来说,推力器的故障属于突发和不确定事件,为了将容错控制技术更好地应用,文献[12]基于现有的基础运动控制器,针对推力器的布置,根据故障诊断进行容错的推力重分配,进一步增强了容错控制的针对性,提高了控制效率.

由于 ROV 的推力器布置一般是冗余的,这就为其推力分配提供了多种可能,所以本文将基于递归神经网络的思想首先实现 ROV 的运动控制,其次优化推力分配,使其在不同的推力器故障状态下都能够取得良好的控制效果.

1 ROV 的运动控制器

1.1 基于扩展函数的递归模糊神经网络控制器

1.1.1 扩展函数

扩展函数(FL)作为输入变量可以产生一组线性独立的函数,即扩展函数. 函数链的建立主要是为了进一步提高系统对水下机器人的非线性特性的识别和处理能力. 本文将使用三角函数,因为它比高斯函数更紧凑,而且易于计算. 所以对于输入矢量 $\boldsymbol{X} = [x_1 \quad x_2]^{\mathrm{T}}$,输入变量可以扩展为

$$\boldsymbol{\Phi} = [\phi_1 \quad \phi_2 \quad \cdots \quad \phi_M]^{\mathrm{T}} =$$

$$[1 \ x_1 \ \sin(\pi x_1) \ \cos(\pi x_1) \ x_2 \ \sin(\pi x_2) \ \cos(\pi x_2) \ x_1 x_2]^{\mathrm{T}}.$$

式中:x_1 是机器人在所要控制的沿某一方向的位置误差,x_2 机器人在该方向上的速度误差,所以 FL 网络的输出是

$$\hat{f}_j = \sum_{M=1}^{8} w_{Mj}\phi_M(X), j = 1,2,\cdots,m.$$

其中:w_{Mj} 是连接权值,$\phi_M(X)$ 是扩展输出函数.

1.1.2 递归模糊神经网络控制器的结构

FNN 控制由于其较强的自适应性和学习能力,在国内外水下机器人控制领域得到了广泛的应用,并取得了一定的效果. 为了在控制中进一步考虑 ROV 系统的不确定性和非线性,本文将通过使用上述的扩展函数链在更多维空间上扩展和线性分隔输入量,并采用基于函数链的递归模糊神经网络(FLFNN)控制器提高控制精度和鲁棒性[13]. 其在 ROV 控制系统结构中的具体实现方法如图 1. 信号的传递和网络每层的基本函数如下:

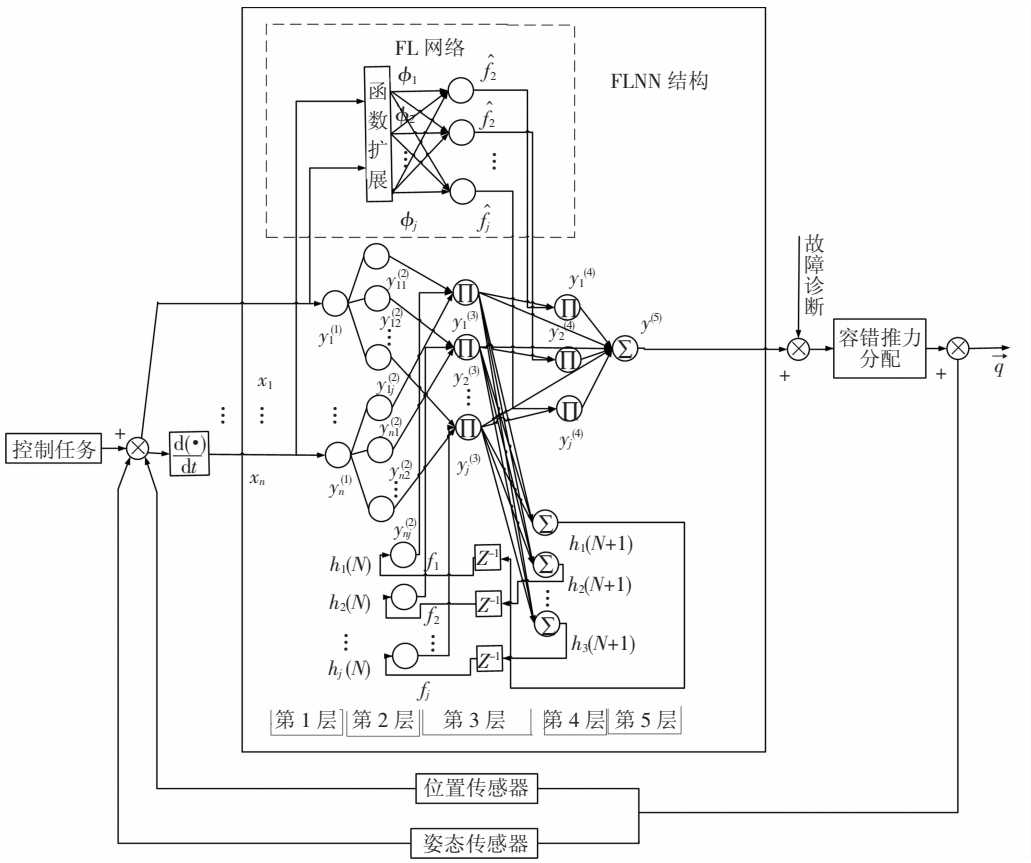


图 1 ROV 控制系统结构

1）输入层. 该层的每个节点直接把输入变量 $x_i(i = 1,2,\cdots,n)$ 直接传递到下一层，即 $y_i^{(1)} = x_i^{(1)}, x_{1i}^{(1)} = \rho_{di} - \rho_{1i} = e_{1i}, x_{2i}^{(1)} = \dot{e}_{1i}$. ρ_{1i} 是实际被控制变量，ρ_{di} 是被控制变量的期望值.

2）隶属度层. 对层内每一个节点，用高斯型隶属度函数计算该层的输出量. 其中 m_i^j 和 σ_i^j 分别是高斯函数的均值和标准差.

$$y_{ij}^{(2)} = \exp\left(-(y_i^{(1)} - m_i^j)^2/(\sigma_i^j)^2\right), j = 1,2,\cdots,j.$$

3）递归层. 该层引入动态反馈以提高学习的效率，对于递归部分的内部变量 h_k，选择 Sigmoid 函数作为激励函数，

$$f_j = (1 + \exp(-h_k))^{-1}, k = j = 1,2,\cdots,m.$$

其中 $h_k = \sum_{j=1}^{m} y_j^{(3)}\theta_{jk}$ 是递归单元即记忆元，θ_{jk} 为递归权值；而且该层的神经元代表模糊规则的预处理值. 所以根据模糊逻辑规则，第 3 层上第 j 个节点的输出为 $y_j^{(3)} = f_j\prod_{i=1}^{2} y_{ij}^{(2)}$.

4）归一化层. 该层的神经元是来自扩展函数的输入信号 $\hat{f}_j$ 和第 3 层的输出 $y_j^{(3)}$ 的乘积. 每一个节点的输出函数是 $y_j^{(4)} = y_j^{(3)}\hat{f}_j$.

5）解模糊层. 该层的每节点对应一个变量的输出 $y^{(5)}$. $y^{(5)}$ 即为求得的相应的推进器控制电压输出值.

1.1.3　在线学习算法

本文将采用梯度下降的算法在线学习和训练递归神经网络的各层参数. 首先定义能量函数为

$$V = 0.5(x_1 - x_{1d})^2 = 0.5e^2.$$

式中：x_{1d} 是被控制量 x_1 所在方向的期望位置.

误差传播项到第 5 层为

$$\delta^{(5)} = -\partial V/\partial y^{(5)},$$

误差传播项到第 4 层误差项为

$$\delta^{(4)} = -\frac{\partial V}{\partial y_j^{(4)}} = -\frac{\partial V}{\partial y^{(5)}}\frac{\partial y^{(5)}}{\partial y_j^{(4)}} = \delta^{(5)}\left(\sum_{j=1}^{m} y_j^{(3)}\right)^{-1}. \quad (1)$$

第 3 层：根据式(1)，扩展函数连接权值的更新算法为

$$\Delta w_{Mj} = -\eta_w\frac{\partial V}{\partial w_{Mj}} = -\eta_w\frac{\partial V}{\partial y_j^{(4)}}\frac{\partial y_j^{(4)}}{\partial \hat{f}_j}\frac{\partial \hat{f}_j}{\partial w_{Mj}} = \eta_w\delta_j^{(4)} y_j^{(3)}\phi_M.$$

其中 η_w 是学习率，可以通过微粒群算法(PSO)来优化. 这样，每步扩展函数的连接权值 w_{Mj} 按下式更新为

$$w_{Mj}(N+1) = w_{Mj}(N) + \Delta w_{Mj}.$$

在递归部分中，动态反馈的连接权值为

$$\Delta\theta_{jk} = -\eta_\theta\frac{\partial V}{\partial \theta_{jk}} = = \eta_\theta\delta_j^{(3)}\left(\prod_{i=1}^{2} y_{ij}^{(2)}\right)f_j(1 - f_j)y_j^{(3)}.$$

η_θ 是学习率，同样可以通过 PSO 算法来优化. 该层的系数 θ_{jk} 同样由 $\theta_{jk}(N+1) = \theta_{jk}(N) + \Delta\theta_{jk}$ 递推得到.

第2层中的 m_i^j 和 σ_i^j 也按照同样的方法递推得到

$$\begin{cases} m_{ij}(N+1) = m_{ij}(N) + \Delta m_{ij}; \\ \sigma_{ij}(N+1) = \sigma_{ij}(N) + \Delta\sigma_{ij}. \end{cases}$$

其中

$$\Delta m_i^j = -\eta_m\frac{\partial V}{\partial m_{ij}} = -\eta_m\frac{\partial V}{\partial y_{ij}^{(2)}}\frac{\partial y_{ij}^{(2)}}{\partial m_{ij}} = \eta_m\delta_{ij}^{(2)} y_{ij}^{(2)}\frac{2(y_i^{(1)} - m_i^j)}{(\sigma_i^j)^2},$$

$$\Delta\sigma_i^j = -\eta_\sigma\frac{\partial V}{\partial \sigma_i^j} = -\eta_\sigma\frac{\partial V}{\partial y_{ij}^{(2)}}\frac{\partial y_{ij}^{(2)}}{\partial \sigma_i^j} = \eta_\sigma\delta_{ij}^{(2)} y_{ij}^{(2)}\frac{2(y_i^{(1)} - m_i^j)^2}{(\sigma_i^j)^3}.$$

式中，η_m 和 η_σ 也都通过 PSO 算法优化得到.

由于 ROV 的动力学仍有许多不确定性，根据 delta 自适应法则设定误差初值 $\delta^{(5)} \cong e + \dot{e}$.

2　推力器的容错推力分配

2.1　ROV 的推力分配模型

由于 ROV 在垂直面的运动只需一个推力器便可实现，所以本节主要研究 ROV 在水平面上的容错运动控制. 通常一台开架式 ROV 配有 4 个水平推力器控制机器人的 3 个自由度运动，记为 iHT，$i = 1,2,\cdots,4$(图 2). 其布置主要有两种形式，X - 形和十字形布置[12]，其推力器构型如表 1和表 2. 根据表 1 和表 2 得到推力器对机器人施加的力和力矩表达式为

$$\vec{\boldsymbol{Q}}^{\mathrm{HT}} = \begin{bmatrix} {}^i\vec{\boldsymbol{F}} \\ {}^i\vec{\boldsymbol{M}} \end{bmatrix} = \begin{bmatrix} {}^iF\vec{\boldsymbol{e}}_i \\ {}^iF(\vec{\boldsymbol{r}}_i \times \vec{\boldsymbol{e}}_i) \end{bmatrix} =$$

$$\begin{bmatrix} e_{ix} & e_{iy} & (\vec{\boldsymbol{r}}_i \times \vec{\boldsymbol{e}}_i)_x & (\vec{\boldsymbol{r}}_i \times \vec{\boldsymbol{e}}_i)_y \end{bmatrix}^{\mathrm{T}} {}^iF =$$

$$\underbrace{\begin{bmatrix} e_{1x} & e_{2x} & \cdots & e_{4x} \\ e_{1y} & e_{2y} & \cdots & e_{4y} \\ (\vec{\boldsymbol{r}}_1 \times \vec{\boldsymbol{e}}_1)_x & (\vec{\boldsymbol{r}}_2 \times \vec{\boldsymbol{e}}_2)_x & \cdots & (\vec{\boldsymbol{r}}_4 \times \vec{\boldsymbol{e}}_4)_x \\ (\vec{\boldsymbol{r}}_1 \times \vec{\boldsymbol{e}}_1)_y & (\vec{\boldsymbol{r}}_2 \times \vec{\boldsymbol{e}}_2)_y & \cdots & (\vec{\boldsymbol{r}}_4 \times \vec{\boldsymbol{e}}_4)_y \end{bmatrix}}_{\vec{R}} \underbrace{\begin{bmatrix} {}^1F \\ {}^2F \\ {}^3F \\ {}^4F \end{bmatrix}}_{\vec{F}} =$$

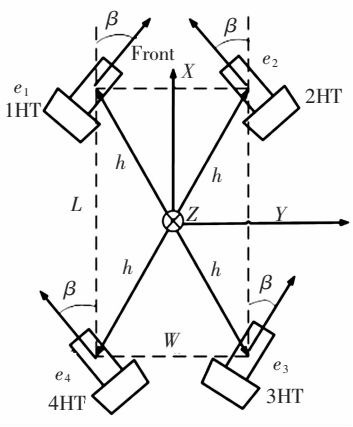
$$\vec{\boldsymbol{R}}\vec{\boldsymbol{F}}. \quad (2)$$

式中：e_{ix}、e_{iy} 代表该推力器的推力的方向，$\vec{\boldsymbol{R}}$ 代表 4 × 4 推力器构型矩阵，$\vec{\boldsymbol{F}}$ 代表 4 维控制力矢量.

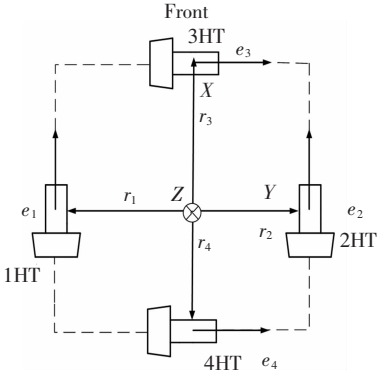
2.2　基于 l_∞ 范数的推力器的容错优化分配

推力分配的目的是把运动控制的控制指令转化成推力器的推力值，即控制指令的实现过程. 发现推力器故障时的推力重分配过程也可以看作是

一个最优规划的过程,即面向控制指令的需要寻找一个最优解. 本节接下来将根据对偶原理,定义能量函数,通过求取能量函数的极值得到故障时容错推力分配的最优解[13].



(a) X - 形布置



(b) 十字形布置

图 2 两种水平推力器布置形式

表 1 推力器的位置

布置形式	1HT	2HT	3HT	4HT
X 形布置	$\begin{bmatrix} \frac{L}{2} \\ -\frac{W}{2} \end{bmatrix}$	$\begin{bmatrix} \frac{L}{2} \\ \frac{W}{2} \end{bmatrix}$	$\begin{bmatrix} -\frac{L}{2} \\ \frac{W}{2} \end{bmatrix}$	$\begin{bmatrix} -\frac{L}{2} \\ -\frac{W}{2} \end{bmatrix}$
十字形布置	$\begin{bmatrix} 0 \\ r_1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ r_2 \end{bmatrix}$	$\begin{bmatrix} r_3 \\ 0 \end{bmatrix}$	$\begin{bmatrix} r_4 \\ 0 \end{bmatrix}$

表 2 推力器的方向

布置形式	$\boldsymbol{e}_1$	$\boldsymbol{e}_2$	$\boldsymbol{e}_3$	$\boldsymbol{e}_4$
X 形布置	$\begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix}$	$\begin{bmatrix} \cos\beta \\ -\sin\beta \end{bmatrix}$	$\begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix}$	$\begin{bmatrix} \cos\beta \\ -\sin\beta \end{bmatrix}$
十字形布置	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$

通过利用推力器输出推力的 l_∞ 范数作为容错控制的推力分配优化准则,当推力器无故障时,每个推力器的控制范围是

$$\boldsymbol{Y} = \| \vec{\boldsymbol{F}} \|_\infty = \max_{1 \leqslant i \leqslant 4} | \boldsymbol{I}_i^{\mathrm{T}} \vec{\boldsymbol{F}} |.$$

式中:$\vec{F}$ 代表推力器的推力.

该线性问题也可改写为下面的对偶命题:

$$\text{minimize} \quad \boldsymbol{W}^{\mathrm{T}} \vec{\boldsymbol{q}} \quad \text{s. t.} \begin{cases} \boldsymbol{C}_2 \vec{\boldsymbol{q}} = \vec{\boldsymbol{D}}_2, \\ \boldsymbol{C}_1 \vec{\boldsymbol{q}} \geqslant \vec{\boldsymbol{D}}_1. \end{cases} \tag{3}$$

式中:$\boldsymbol{W} = [0 \quad \cdots \quad 0 \quad 1]^{\mathrm{T}} \in \mathbf{R}^5$;

$\vec{\boldsymbol{q}} = [\vec{\boldsymbol{Q}}^{\mathrm{HT}} \ \boldsymbol{Y}]^{\mathrm{T}} \in \mathbf{R}^5$;

$$\boldsymbol{C}_1 = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{I} & -\boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I}_4 & \boldsymbol{I}_4 & \boldsymbol{0}_{4\times1} & \boldsymbol{0}_{4\times1} \end{bmatrix}^{\mathrm{T}} \in \mathbf{R}^{16\times5},$$

$\boldsymbol{C}_2 = [\vec{\boldsymbol{R}}^{\mathrm{HT}} \quad \boldsymbol{0}_3] \in \mathbf{R}^{5\times3}$;

$\vec{\boldsymbol{D}}_1 = [\boldsymbol{0}_4 \quad \boldsymbol{0}_4 \quad -\vec{\boldsymbol{Q}}_u \quad \vec{\boldsymbol{Q}}_l]^{\mathrm{T}} \in \mathbf{R}^{16}$,

$\vec{\boldsymbol{D}}_2 = \vec{\boldsymbol{Q}} \in \mathbf{R}^3$.

其中,

$$\boldsymbol{R}^{\mathrm{HT}} = \begin{bmatrix} \overbrace{\cos b}^{\mathrm{HT}_1} & \overbrace{\cos b}^{\mathrm{HT}_2} & \overbrace{\cos b}^{\mathrm{HT}_3} & \overbrace{\cos b}^{\mathrm{HT}_4} \\ \sin b & -\sin b & \sin b & -\sin b \\ h & -h & -h & h \end{bmatrix} (X\text{形布置})$$

或$\begin{bmatrix} \overset{\mathrm{HT}_1}{1} & \overset{\mathrm{HT}_2}{1} & \overset{\mathrm{HT}_3}{0} & \overset{\mathrm{HT}_4}{0} \\ 0 & 0 & 1 & 1 \\ r_1 & r_2 & r_3 & r_4 \end{bmatrix}$(十字形布置)代表水平面 3 个自由度的推力构型;

$\boldsymbol{I} = [\boldsymbol{I}_4 \quad \boldsymbol{I}_4 \quad \boldsymbol{I}_4 \quad \boldsymbol{I}_4]$, $\boldsymbol{I}_4 = [1 \quad 1 \quad 1 \quad 1]^{\mathrm{T}}$;

$\vec{\boldsymbol{Q}}_l$、$\vec{\boldsymbol{Q}}_u$ 分别代表推力值的下限和上限,

$$\vec{\boldsymbol{Q}}_l = [\vec{\boldsymbol{Q}}_{1l} \quad \vec{\boldsymbol{Q}}_{2l} \quad \vec{\boldsymbol{Q}}_{3l} \quad \vec{\boldsymbol{Q}}_{4l}],$$

$$\vec{\boldsymbol{Q}}_u = [\vec{\boldsymbol{Q}}_{1u} \quad \vec{\boldsymbol{Q}}_{2u} \quad \vec{\boldsymbol{Q}}_{3u} \quad \vec{\boldsymbol{Q}}_{4u}].$$

2.3 故障适应性方法

推力器的故障状态一般可分为 3 种类型:无故障、部分故障和完全故障. 定义推力器的故障状态权值为 $\boldsymbol{h} = \mathrm{diag}(\eta_1 \quad \eta_2 \quad \eta_3 \quad \eta_4)$,当推力器正常工作时 $\eta_i = 1$,当推力器部分故障时 $0 < \eta_i < 1$,推力器完全故障时 $\eta_i = 0$. 当推力器部分故障时,可继续限制使用. 推力器故障时,定义 D_1 使

$$\overline{\boldsymbol{D}}_1 = [\boldsymbol{0}_{4\times1} \quad \boldsymbol{0}_{4\times1} \quad -\overline{\vec{\boldsymbol{Q}}}_u \quad \overline{\vec{\boldsymbol{Q}}}_l]^{\mathrm{T}},$$

其中$\overline{\vec{\boldsymbol{Q}}}_u = \eta\vec{\boldsymbol{Q}}_u$,$\overline{\vec{\boldsymbol{Q}}}_l = \eta\vec{\boldsymbol{Q}}_l$,这样即可通过式(2)进行推力器的容错推力分配.

进一步改写 $\boldsymbol{W}$ 中的各项使

$$w_i = 1 + \delta/\eta_i.$$

其中,$\delta > 0$,这就可以根据 w_i 确定推力器使用的优先级,$\boldsymbol{W}$ 即为推力器使用的加权矩阵.

2.4 容错推力分配策略

引理 1[14] 设 $\vec{\boldsymbol{q}}^* \in \mathbf{R}^4$ 是(3)的解,定义 Lagrange 函数 $L(\vec{\boldsymbol{q}}, \Omega, \gamma) = \boldsymbol{W}^{\mathrm{T}}\vec{\boldsymbol{q}}^*$. 其中 $\boldsymbol{\Omega} \in \mathbf{R}$、

$\boldsymbol{\Omega} \geqslant 0$、$\gamma \in \mathbf{R}$ 为 Lagrange 乘子. 若 $\bar{q}^*$ 是式(4) 中的一个约束点，则存在 $\bar{q}^*$、$\boldsymbol{\Omega}$ 和 γ 使 $\boldsymbol{L}(\vec{\boldsymbol{q}}, \boldsymbol{\Omega}, \gamma)$ 达到最大值.

当 $\boldsymbol{L}$ 梯度为 $\nabla_{\bar{q}} L(\vec{\boldsymbol{q}}, \boldsymbol{\Omega}, \gamma) = 0$ 时，$\boldsymbol{L}$ 取最大值. 于是进一步定义 Lagrange 函数 L 使

$$L(\vec{\boldsymbol{q}}, \boldsymbol{\Omega}, \gamma) = \vec{\boldsymbol{q}}(\boldsymbol{W} - \boldsymbol{C}_1^{\mathrm{T}}\boldsymbol{\Omega} - \boldsymbol{C}_2^{\mathrm{T}}\gamma) + \bar{\boldsymbol{D}}_1^{\mathrm{T}}\boldsymbol{\Omega} + \boldsymbol{D}_2^{\mathrm{T}}\gamma. \tag{4}$$

则式(4) 对 $\vec{\boldsymbol{q}}$ 求取梯度得

$$\nabla_{\vec{q}} L(\vec{\boldsymbol{q}}, \boldsymbol{\Omega}, \gamma) = \boldsymbol{W} - \boldsymbol{C}_1^{\mathrm{T}}\boldsymbol{\Omega} - \boldsymbol{C}_2^{\mathrm{T}}\gamma. \tag{5}$$

分别对(4)和(5)应用对偶原理，将(2)代入引理 1，得

$$\begin{cases} \text{maximize} & \bar{\boldsymbol{D}}_1^{\mathrm{T}}\boldsymbol{\Omega} + \boldsymbol{D}_2^{\mathrm{T}}\gamma, \\ \text{s.t.} & \boldsymbol{C}_1^{\mathrm{T}}\boldsymbol{\Omega} + \boldsymbol{C}_2^{\mathrm{T}}\gamma - \boldsymbol{W} = 0. \end{cases} \tag{6}$$

其中 $\boldsymbol{\Omega} \geqslant 0$，式(6) 就是原命题(2) 的对偶命题.

根据式(3)、(6)和引理 1 定义连续可微的能量函数 $\boldsymbol{E}$，使

$$\begin{aligned} \boldsymbol{E} = {} & 0.5 \| \boldsymbol{C}_1^{\mathrm{T}}\boldsymbol{\Omega} + \boldsymbol{C}_2^{\mathrm{T}}\gamma - \boldsymbol{W} \|_2^2 + 0.5 \| \boldsymbol{C}_2\vec{\boldsymbol{q}} - \boldsymbol{D}_2 \|_2^2 + \\ & 0.5(\boldsymbol{W}^{\mathrm{T}}\vec{\boldsymbol{q}} - \boldsymbol{D}_1^{\mathrm{T}}\boldsymbol{\Omega} - \boldsymbol{D}_2^{\mathrm{T}}\gamma)^2 + 0.25(\boldsymbol{C}_1\vec{\boldsymbol{q}} - \bar{\boldsymbol{D}}_1)^{\mathrm{T}} \cdot \\ & (\boldsymbol{C}_1\vec{\boldsymbol{q}} - \bar{\boldsymbol{D}}_1 - | \boldsymbol{C}_1\vec{\boldsymbol{q}} - \bar{\boldsymbol{D}}_1 | + 0.25\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{W} - | \boldsymbol{W} |)). \end{aligned} \tag{7}$$

根据(7)和对偶原理，式(6)的解可转化为下面的对偶问题：

$$\begin{aligned} & \text{Minimize } \boldsymbol{E}, \\ & \text{s.t. } \vec{\boldsymbol{u}} = [\vec{\boldsymbol{q}} \quad \boldsymbol{\Omega} \quad \gamma]^{\mathrm{T}}. \end{aligned}$$

根据引理 1 求取能量函数 E 的梯度，得到

$$\frac{\mathrm{d}\vec{\boldsymbol{u}}}{\mathrm{d}t} = -\kappa \nabla \boldsymbol{E}.$$

式中：$\kappa \in R_+^{\cdot}$ 其取值将决定式(2) 的收敛速度. 于是根据(7) 得

$$\begin{cases} \dfrac{\mathrm{d}\vec{\boldsymbol{q}}}{\mathrm{d}t} = \kappa[\boldsymbol{W}(\boldsymbol{W}^{\mathrm{T}}\vec{\boldsymbol{q}} - \boldsymbol{D}_1^{\mathrm{T}}\boldsymbol{\Omega} - \boldsymbol{D}_2^{\mathrm{T}}\gamma) + \boldsymbol{C}_2^{\mathrm{T}}(\boldsymbol{C}_2\vec{\boldsymbol{q}} - \boldsymbol{D}_2) + \boldsymbol{C}_1^{\mathrm{T}}(\boldsymbol{C}_1\vec{\boldsymbol{q}} - \bar{\boldsymbol{D}}_1)^{-}], \\ \dfrac{\mathrm{d}\boldsymbol{\Omega}}{\mathrm{d}t} = -\kappa[\boldsymbol{\Omega}^{-} - \bar{\boldsymbol{D}}_1(\boldsymbol{W}^{\mathrm{T}}\vec{\boldsymbol{q}} - \boldsymbol{D}_1^{\mathrm{T}}\boldsymbol{\Omega} - \boldsymbol{D}_2^{\mathrm{T}}\gamma) + \boldsymbol{C}_1(\boldsymbol{C}_1^{\mathrm{T}}\boldsymbol{\Omega} + \boldsymbol{C}_2^{\mathrm{T}}\gamma - \boldsymbol{W})], \\ \dfrac{\mathrm{d}\gamma}{\mathrm{d}t} = -\kappa[-\boldsymbol{D}_2(\boldsymbol{W}^{\mathrm{T}}\vec{\boldsymbol{q}} - \boldsymbol{D}_1^{\mathrm{T}}\boldsymbol{\Omega} - \boldsymbol{D}_2^{\mathrm{T}}\gamma) + \boldsymbol{C}_2(\boldsymbol{C}_1^{\mathrm{T}}\boldsymbol{\Omega} + \boldsymbol{C}_2^{\mathrm{T}}\gamma - \boldsymbol{W})]. \end{cases}$$

其中 $\partial_i^- = \min\left\{0 \quad \dfrac{\partial_i - | \partial_i |}{2}\right\}$.

图 3 为容错推力分配的算法流程图. 在故障条件下，通过图 3 的流程进行推导，使 $\boldsymbol{E}(\vec{q}^*, \boldsymbol{\Omega}^*, \gamma^*) < \delta$，就可以进行推力器的容错优化分配.

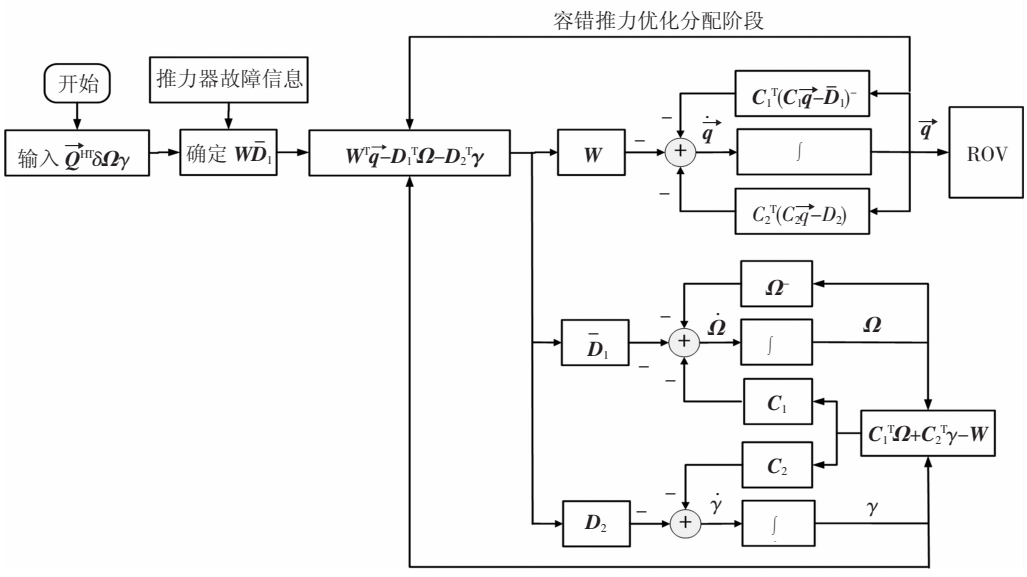


图 3　容错推力分配的算法流程图

3　仿真和实验结果

在控制器的仿真和试验验证中，本文采用 SY－II型 ROV，该机器人在水面使用 Pentium－IV 工控机发送控制信息，控制舱内使用 PC104 嵌入式计算机和水面通过光纤通讯，对机器人实现遥控. 该机器人的 4 个水平推力器采用图 2(a) 的十字交叉形布置并携带深度计和磁罗经作为运动传感器. 在表 3 和表 4 中给出了水动力系数及和运动控制相关的参数.

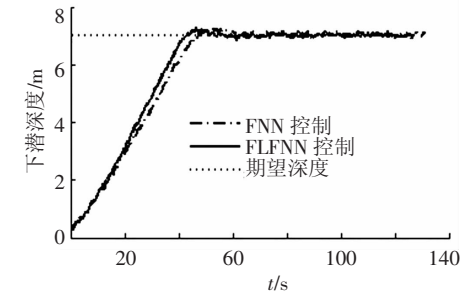
表 3　SY－II ROV 的水动力系数

无因次系数	纵向	横向	垂向	横倾	纵倾	偏航
一次项	27.65	47.26	44.86	62.63	57.55	141.33
平方项	95.62	281.65	258.12	19.18	22.68	33.84

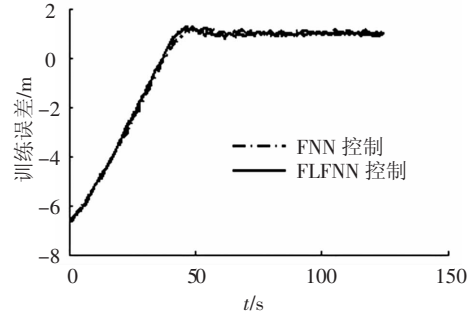
表 4　SY－II ROV 和运动相关的质量与惯性参数　$\mathbf{N \cdot m \cdot s^2}$

质量/kg	I_x	I_y	I_z	I_{xy}	I_{xz}	I_{xz}
111.9	97.3	26.1	56.8	0	0	0

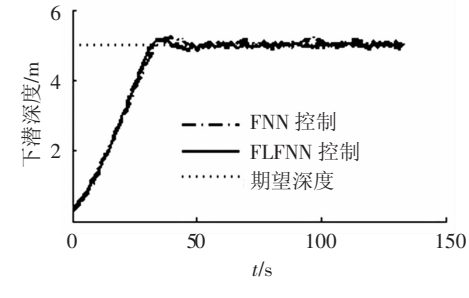
实验环境为哈尔滨工程大学水下机器人重点实验室的试验水池. 为验证 FLFNN 控制器,首先进行水池的定深控制,同时与传统的模糊神经网络控制器的实验结果比较,结果见图 4. 训练参数经 PSO 优化得到: $\eta_m = 167.5$, $\eta_\sigma = 0.15$, $\eta_\theta = 97.7$, $\eta_w = 14.8$



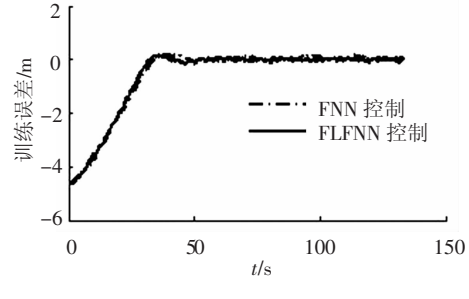
(a) 定深控制实验(7 m)



(b) 误差比较



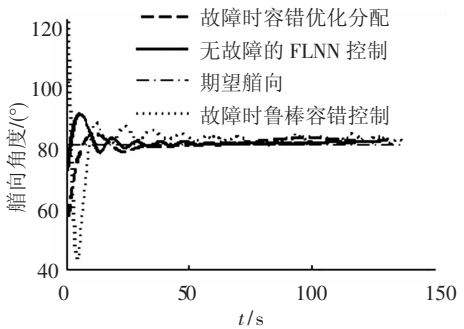
(c) 定深控制实验(5 m)



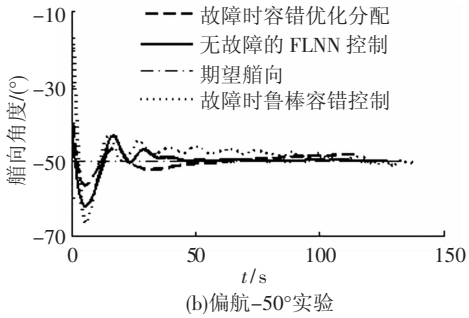
(d) 误差比较结果

图 4 "SY – II"定深实验与比较

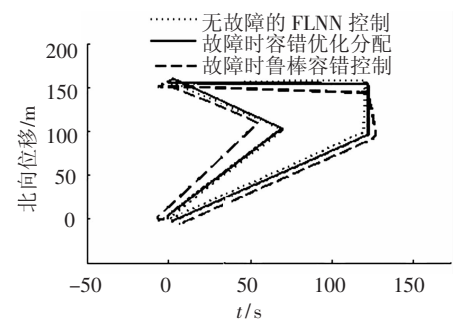
由图 4 可知,本文的 FLFNN 控制器比传统的模糊神经网络控制器在收敛速度和减小误差方面都有所提高. 在图 5 中,首先使用 FLFNN 控制器进行 ROV 的偏航控制,控制目标分别为艏向 80°和 –50°;其次假设左主推和艉侧推完全故障,使用 FLFNN 控制器结合推力优化分配进行可容错的定向控制实验,容错控制器的控制参数分别为 $[\kappa \quad \Omega \quad \gamma] = [10^6 \quad 1.95 \quad -0.9]$, 图 5(a)和(b)中的虚线即为试验结果;独立使用文献[9]中的鲁棒容错进行上述偏航控制(图 5(a)和(b)的点线所示),并与本文的方法加以比较. 对于 ROV 来说,脐带缆的干扰和机器人本身的长宽比将对定向控制的准确率造成比较大的影响,尤其是脐带缆,会给控制器造成较强的非线性干扰. 在偏航控制的开始阶段,脐带缆的反作用力将随机器人的加速度增加而增大,但当机器人艏向保持稳定以后其干扰反而减少,和 FNN 控制器相比,所设计 FLFNN 控制器中的扩展函数链可进一步考虑机器人的非线性特性,在偏航控制中取得较好的效果. 推力器故障时,通过优化分配推力可以继续保证比较准确地偏航控制,而且控制误差要比文献[7]中的鲁棒容错控制器小.



(a)偏航 80°实验



(b)偏航–50°实验



(c)运动仿真

图 5 推力优化分配实验与仿真的比较

由于 ROV 没有位置传感器，所以在图 5(c)的点位运动控制仿真中，推力器故障条件保持不变，设机器人的期望运动轨迹为[0　0]→[65　100]→[150　0]→[120　150]→[120　100]→[0　0]，再次比较上述几种容错控制方法. 可见使用前面的 FLNN 控制可以很好地完成水平面的点位轨迹控制，在故障时通过容错推力优化分配，虽然在前进方向转变时有些误差，但基本也可完成故障时的点位轨迹控制，其效果要好于文献[9]的鲁棒容错控制方法.

4　结　论

1)本文基于扩展函数设计了 FLFNN 控制器，通过扩展函数提高了控制器对机器人非线性特性的控制能力.

2)根据能量函数使用反向梯度传播方法设计了学习算法，由 PSO 算法确定了学习率参数，从而实现了算法的全局收敛.

3)在推力器的容错控制方面，通过对开架式遥控水下机器人的两种推力器布置形式加以建模，为推力分配提供了容错的可能. 并基于对偶原理使用的 l_∞ 范数建立能量函数，利用推力器的优先级矩阵 $\boldsymbol{W}$ 优化分配推力器的推力，使水平面的容错控制得以实现.

4)本文方法不仅可用于遥控水下机器人，也对自主水下机器人的控制有一定的借鉴意义.

5)在水池和仿真实验中，FLFNN 控制器和传统的模糊神经网络控制器相比，其对 ROV 控制的收敛速度、抗干扰能力和控制精度都有所提高. 在一个主推或侧推故障时，使用本文的方法进行推力优化分配，仍可在水平面实现准确的位置控制，这就进一步保障了 ROV 在水下的长时间可靠工作.

参考文献

[1] 刘建成，于华南，徐玉如. 水下机器人改进的 S 面控制方法[J]. 哈尔滨工程大学学报，2002，23(1)：33－36.

[2] 王建国，吴恭兴，万磊，等. 广义 S 面的水下机器人控制器设计[J]. 电机与控制学报，2009，13(增 1)：144－148.

[3] 俞建成，李强，张艾群，等. 水下机器人的神经网络自适应控制[J]. 控制理论与应用，2008，25(1)：9－13.

[4] 梁　霄，张均东，李　巍，等. 水下机器人 T－S 型模糊神经网络控制[J]. 电机与控制学报，2010，14(7)：99－104.

[5] WANG J S，LEE G C S. Self-adaptive recurrent neuro-fuzzy control of an autonomous underwater vehicle[J]. IEEE Transactions on Robotics and Automation，2003，19(2)：283－295.

[6] ISHAQUE K，ABUDULLAH S S，AYOB S M，*et al.* A simplified approach to design fuzzy logic controller for an underwater vehicle[J]. Ocean Engineering，2011，38：271－284.

[7] YOO S J，CHOI Y H，PARK J B. Generalized predictive control based on self-recurrent wavelet neural network for stable path tracking of mobile robots：Adaptive learning rates approach[J]. IEEE Trans Circuit Syst，2006，53(6)：1381－1394.

[8] 黄　海，万　磊，庞永杰，等. SY－II 遥控式水下机器人推力器容错控制的研究[J]. 应用基础与工程科学学报，2012，20(12)：1118－1127.

[9] CHENG Xiangqin，QU Jingyuan，YAN Zheping，*et al.* H_∞ robust fault-tolerant controller design for an autonomous underwater vehicle's navigation control system[J]. J Marine Sci Appl，2010，9：87－92.

[10] 杨立平，张铭钧，褚振忠，等. 水下机器人抗积分饱和控制及主动容错控制方法[J]. 哈尔滨工程大学学报，2010. 31(6)：755－760.

[11] 俞建成，张艾群，王晓辉. 7 000 m 载人潜水器推进器故障容错控制分配研究[J]. 机器人，2006，28(5)：519－524.

[12] OMERDIC E，ROBERTS G. Thruster fault diagnosis and accommodation for open-frame underwater vehicles[J]. Control Engineering Practice，2004，12(2)：1575－1598.

[13] CORRADINI M L，MONTERIÙ A，ORLANDO G. An actuator failure tolerant control scheme for an underwater remotely operated vehicle[J]. IEEE Transaction on control systems technology，2011，19(5)：1036－1046.

[14] TANG W，WANG J. A recurrent neural network for minimum infinity-norm kinematic control of redundant manipulators with an improved problem formulation and reduced architecture complexity[J]. IEEE Transactions on Systems，Man，and Cybernetics，2001，31(1)：98－105.

[15] PODEER T K，ANTONELLI G，SARKAR N. Fault tolerant control of an autonomous underwater vehicle under thruster redundancy：Simulations and experiments[C]//Proceeding of the 2000 IEEE International Conference on Robotics & Automation. San Francisco：[s. n.]，2000：1251－1256.

（编辑　杨　波）