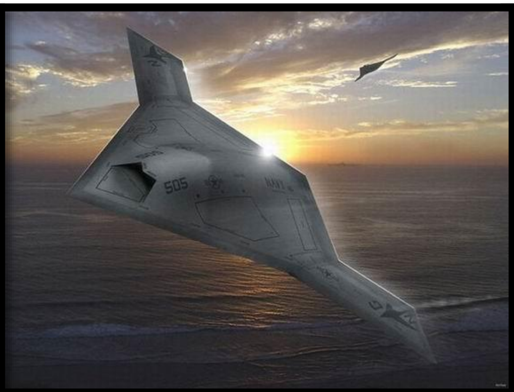
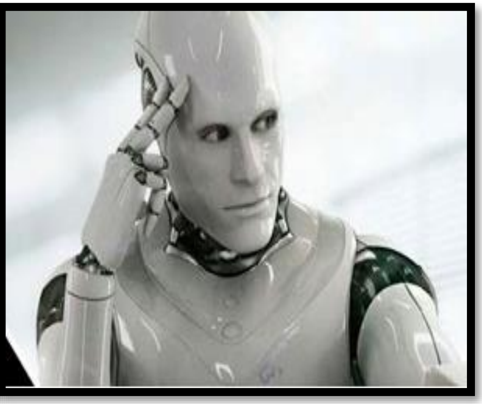
机器人智能控制

易建强 蒲志强 袁如意

中国科学院自动化研究所

2020年秋季

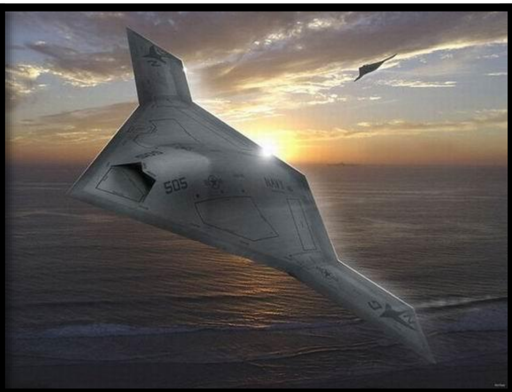


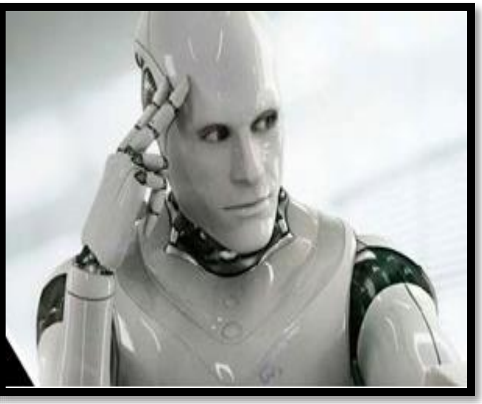


神经网络控制

袁如意 高级工程师

ruyi.yuan@ia.ac.cn





本讲的主要内容

一、人工神经元控制系统

二、RBF神经网络控制

三、CMAC神经网络控制

四、Hopfield网络优化

本讲的主要内容

一、人工神经元控制

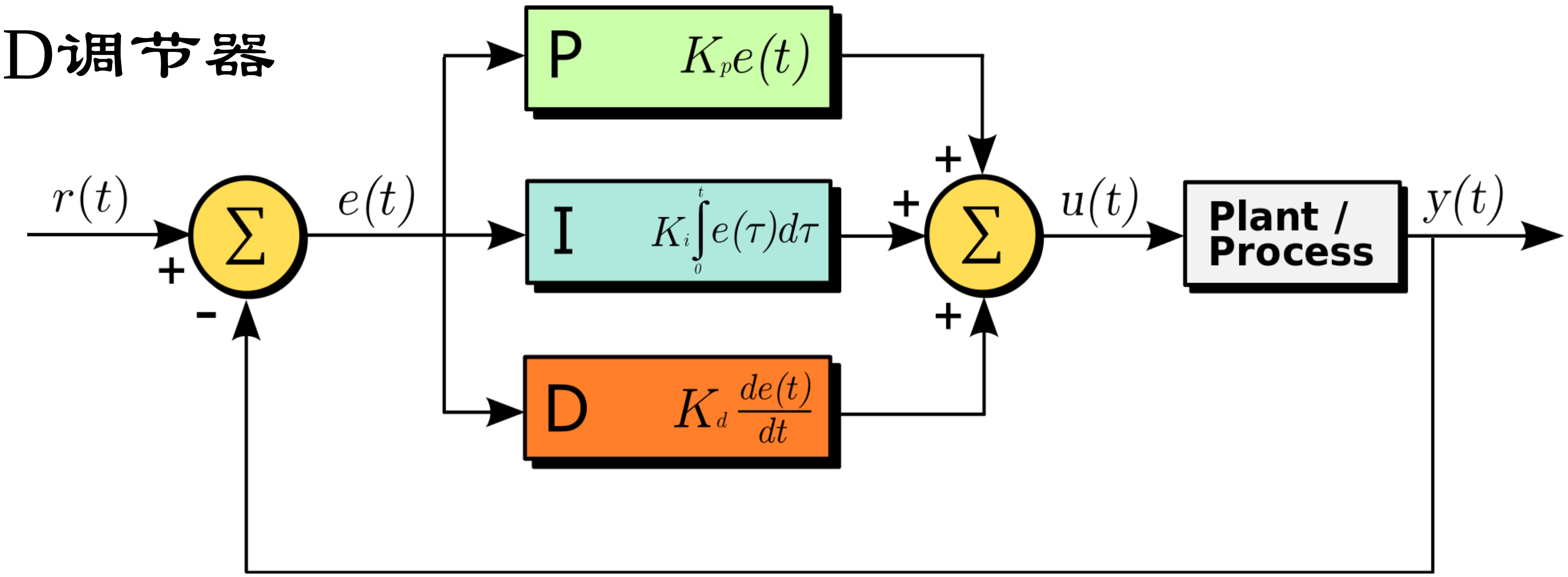
二、RBF神经网络控制

三、CMAC神经网络控制

四、Hopfield网络优化

人工神经元的PID调节器

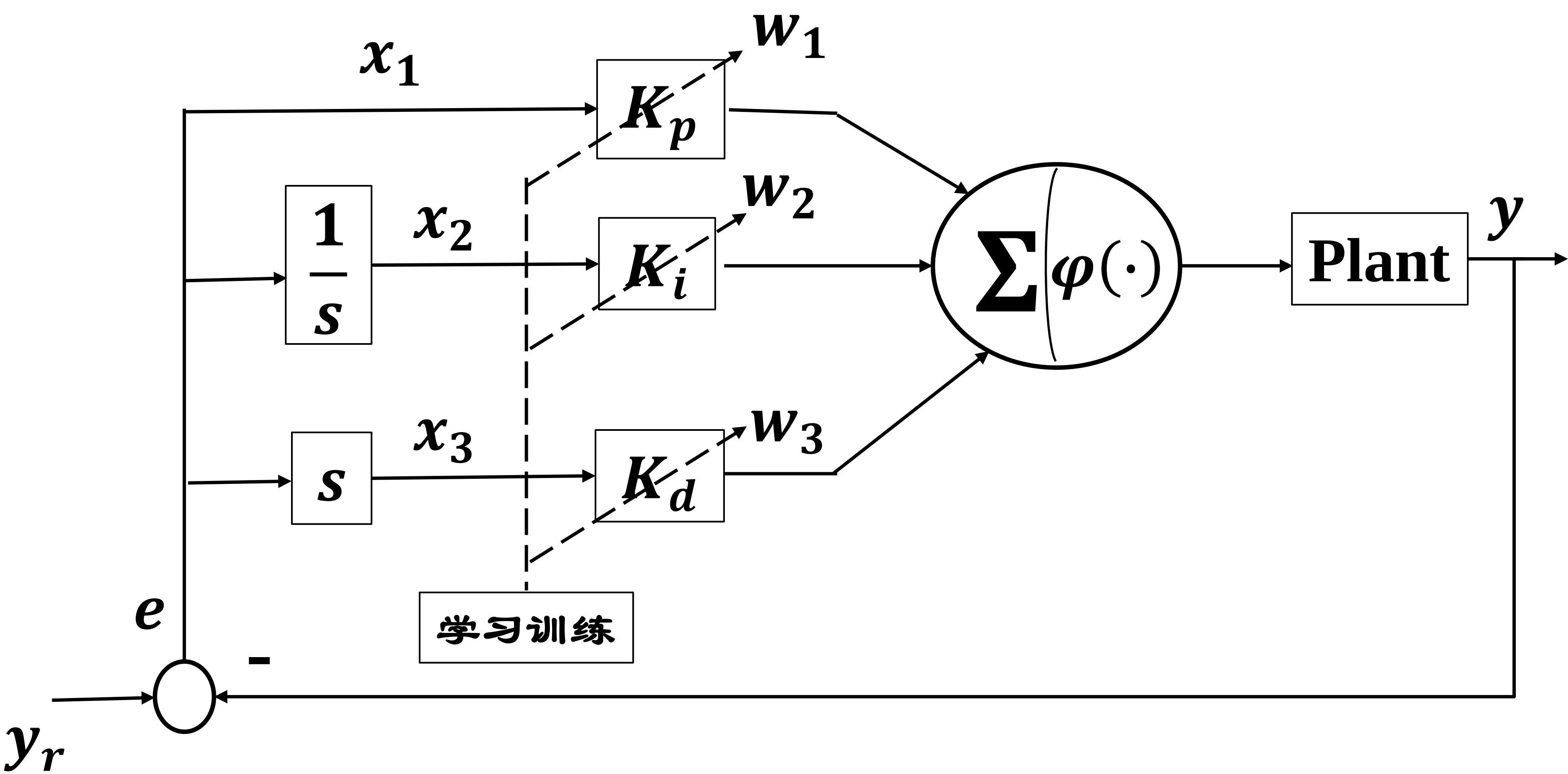
- PID调节器



$$u(t) = K_p e(t) + K_i \int e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

$$= K_p \left[e(t) + \frac{1}{T_I} \int e(\tau)d\tau + T_D \frac{de(t)}{dt} \right]$$

- 人工神经元闭环调节系统



神经网络控制

- 离散PID

$$\boldsymbol{u(t)=K_p\left[\overbrace{e(t)}^{x_1(t)}+\frac{1}{T_I}\underbrace{\sum e_iT}_{x_2(t)}+T_D\underbrace{\frac{e(k)-e(k-1)}{T}}_{x_3(t)}\right]}$$

$$I(t)=K_p\sum_{i=1}^{3}\frac{w_i(t)x_i(t)}{\|W\|}\qquad \|W\|=\sum_{i=1}^{3}|w_i(t)|$$

神经元输出

$$u(t)=u_{max}\frac{1-e^{-I(t)}}{1+e^{-I(t)}}$$

$$J(t)=\frac{e^2(t)}{2}$$

学习算法：梯度下降算法

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$\Delta w_i(t) = -\eta_i \frac{\partial J}{\partial w_i(t-1)}$$

$$= -\eta_i \frac{\partial J}{\partial y(t)} \frac{\partial y(t)}{\partial u(t-1)} \frac{\partial u(t-1)}{\partial I(t-1)} \frac{\partial I(t-1)}{\partial w_i(t-1)}$$

- ✓ $\frac{\partial y(t)}{\partial u(t-1)} \approx \frac{y(t)-y(t-1)}{u(t-1)-u(t-2)}$
- ✓ $\frac{\partial y(t)}{\partial u(t-1)} \approx \operatorname{sgn}\left[\frac{y(t)-y(t-1)}{u(t-1)-u(t-2)}\right]$

稳定性分析

取Lyapunov函数

$$V(t)=\frac{1}{2}e^2(t)$$

$$e(t+1)=e(t)+\Delta e(t)=e(t)+\frac{\partial e(t)}{\partial W}\Delta W^T$$

$$\Delta V(t)=V(t+1)-V(t)=\frac{1}{2}e^2(t+1)-\frac{1}{2}e^2(t)=\frac{1}{2}[2e(t)\Delta e(t)+\Delta^2 e(t)]$$

$$P=\left[\frac{\partial e(t)}{\partial w_1},\frac{\partial e(t)}{\partial w_2},\frac{\partial e(t)}{\partial w_3}\right]^T,D=\mathrm{diag}\{\eta_1,\eta_1,\eta_1\}$$

$$\Delta e(t)=-e(t)P^T DP$$

当$0<D<2(PP^T)^{-1}$

$$\Delta V(t)=-\frac{1}{2}[e(t)P]^T(2D-DPP^T D)[e(t)P]<0$$

学习算法：有监督的Hebb学习规则

$$u(k)=u(k-1)+\Delta u(k)$$

$$\Delta u(k)=K_p\left\{e(k)-e(k-1)+\frac{T}{T_I}e(k)+\frac{T}{T_D}(e(k)-2e(k-1)+e(k-2))\right\}$$

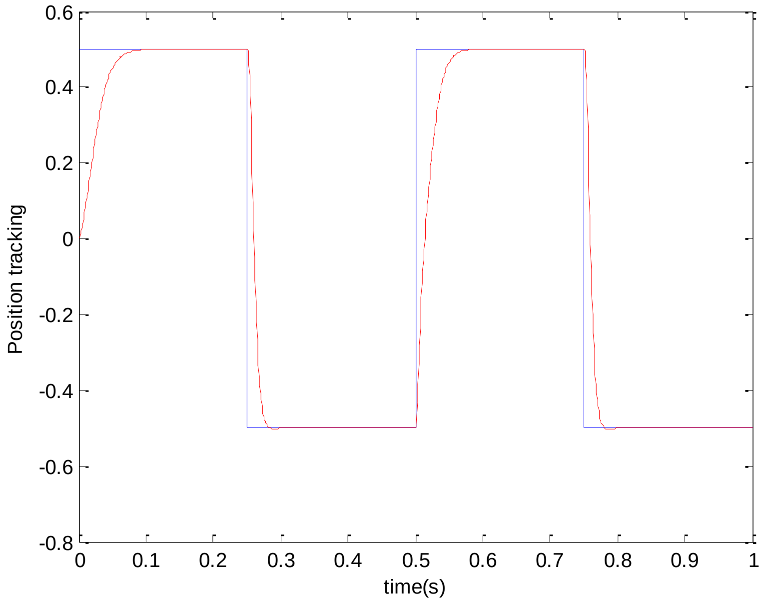
$$x_1(k)=e(k), x_2(k)=e(k)-e(k-1), x_3(k)=e(k)-2e(k-1)+e(k-2)$$

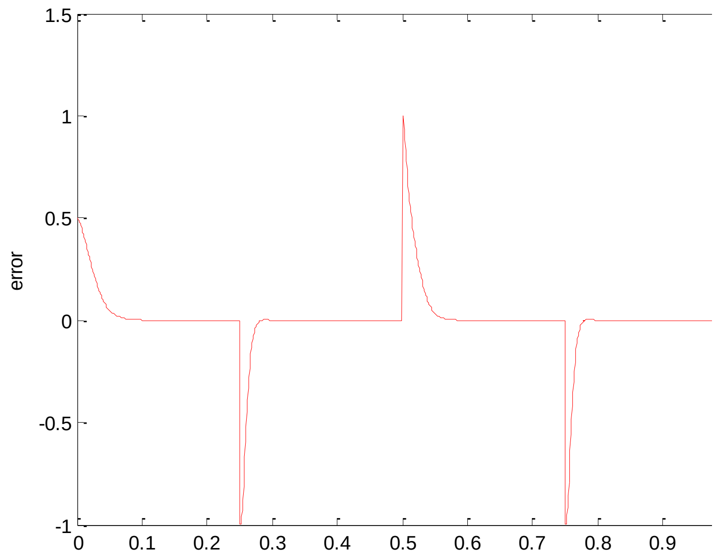
$$u(k)=u(k-1)+K_p\sum_{i=1}^{3}w_i(k)x_i(k)$$

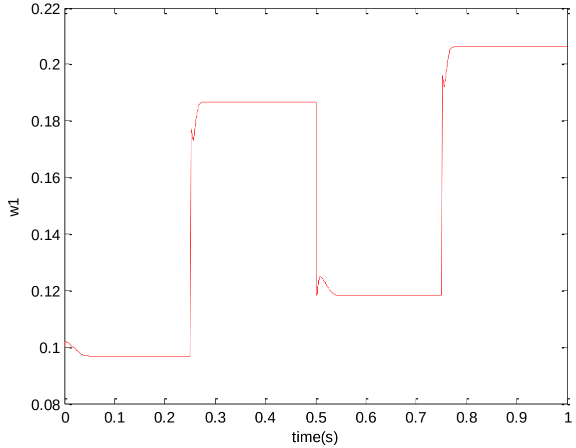
$$w_1(k)=w_1(k-1)+\eta e(k)u(k)x_1(k)$$
$$w_2(k)=w_2(k-1)+\eta e(k)u(k)x_2(k)$$
$$w_3(k)=w_3(k-1)+\eta e(k)u(k)x_3(k)$$

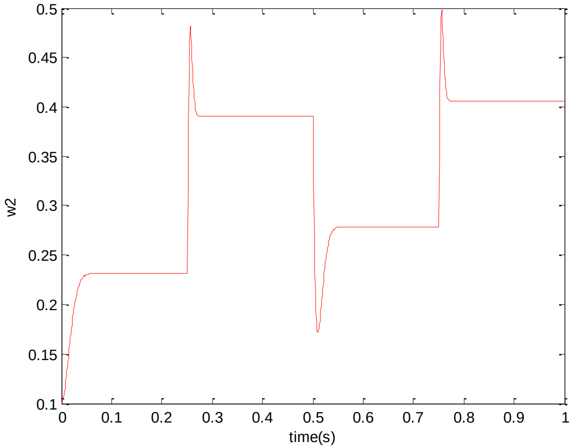
神经网络控制

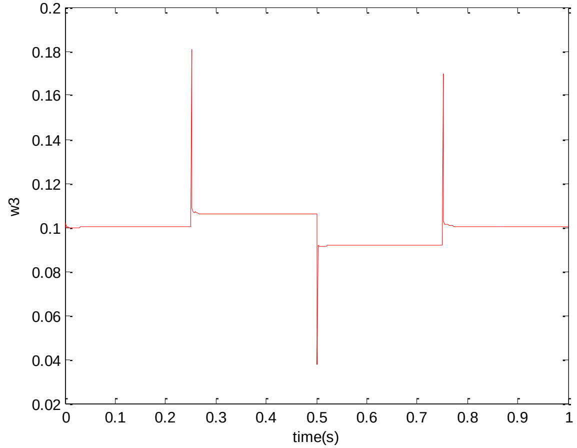
$$y(k) = 0.368y(k-1) + 0.26y(k-2) + 0.10u(k-1) + 0.632u(k-2)$$











本讲的主要内容

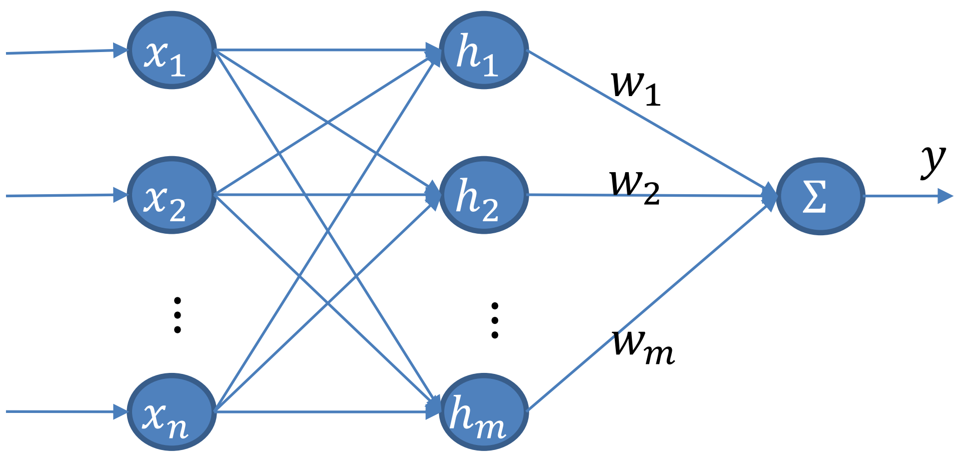
一、人工神经元控制

二、RBF神经网络控制

三、CMAC神经网络控制

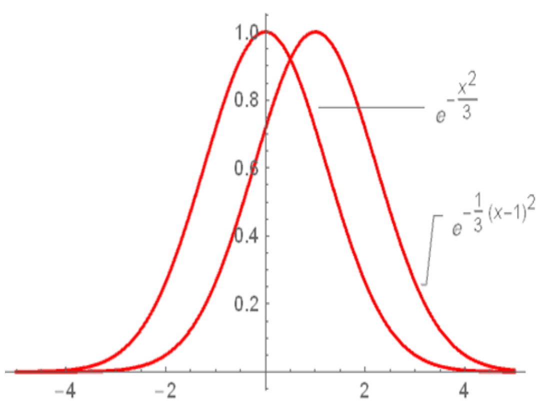
四、Hopfield网络优化

RBF (Radial Basis Function) 神经网络

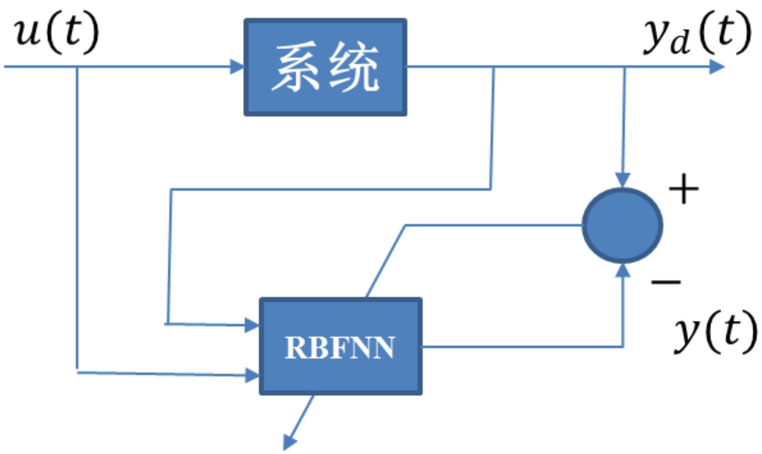


- 隐层由径向基函数构成
- 具有良好的泛化能力，结构简单，收敛速度快
- 能在一个紧凑集和任意精度下逼近任何非线性函数
- 输入层到隐藏层之间不是通过权值和阈值进行连接的，而是通过输入样本与隐藏层点之间的距离（与中心点的距离）连接的

- $h_j = exp\left(\dfrac{-\left\|x-c_j\right\|^2}{2b_j^2}\right)$
- $w = [w_1, w_2, \cdots, w_m]^T, h = [h_1, h_2, \cdots, h_m]^T$
- $y(t) = w^T h = w_1 h_1 + w_2 h_2 + \cdots + w_m h_m$



RBF神经网络建模



RBF神经网络逼近

$$E(t)=\frac{1}{2}(y_d(t)-y(t))^2$$

$$\Delta w_j(k)=-\eta\frac{\partial E}{\partial w_j}=\eta\big(y_d(k)-y(k)\big)h_j$$

$$w_j(k)=w_j(k-1)+\Delta w_j(k)+\alpha\big(w_j(k-1)-w_j(k-2)\big)$$

$$\Delta b_j(k)=-\eta\frac{\partial E}{\partial b_j}=\eta(y_d(k)-y(k))w_jh_j\frac{\left\|x-c_j\right\|^2}{b_j^3}$$

$$b_j(k)=b_j(k-1)+\Delta b_j(k)+\alpha\big(b_j(k-1)-b_j(k-2)\big)$$

$$\Delta c_{ji}(k)=-\eta\frac{\partial E}{\partial c_{ji}}=\eta(y_d(k)-y(k))w_jh_j\frac{x_j-c_{ji}}{b_j^2}$$

$$c_{ji}(k)=c_{ji}(k-1)+\Delta c_{ji}(k)+\alpha\big(c_{ji}(k-1)-c_{ji}(k-2)\big)$$

仿真实例

考虑第III类系统 $$\boldsymbol{y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k)}$$

辨识模型M为$\boldsymbol{\hat{y}(k+1)=N_f[y(k)]+N_g[u(k)]}$

辨识模型M为$\boldsymbol{\hat{y}(k+1)=N[y(k),u(k)]}$

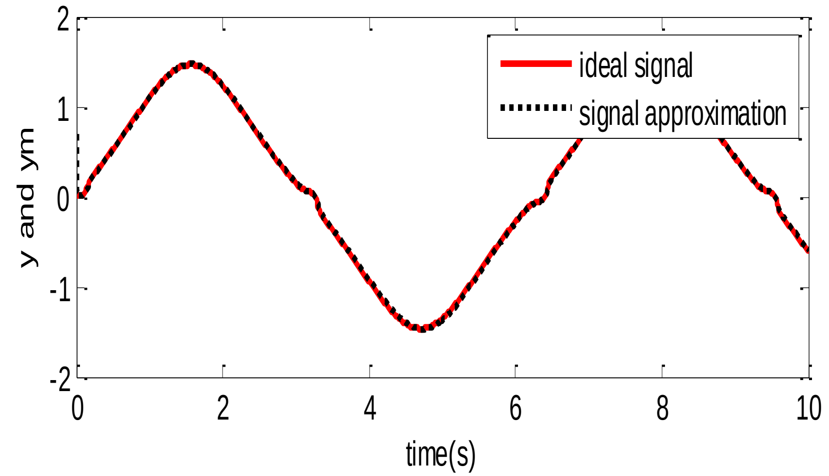
$$x(1)=u(t),x(2)=y(t),\alpha=0.15$$

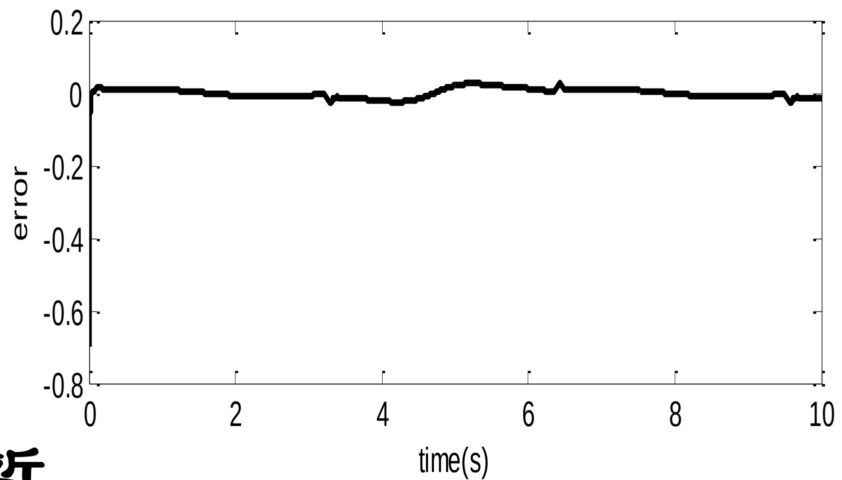
$x(1)\in[0,1],x(2)\in[0,1]$ **网络结构取2-5-1**

$$c_j=\begin{bmatrix}-1 & -0.5 & 0 & 0.5 & 1\\ -1 & -\mathbf{0}.5 & 0 & \mathbf{0}.5 & 1\end{bmatrix}^T, b_j=3.0$$

仿真实例

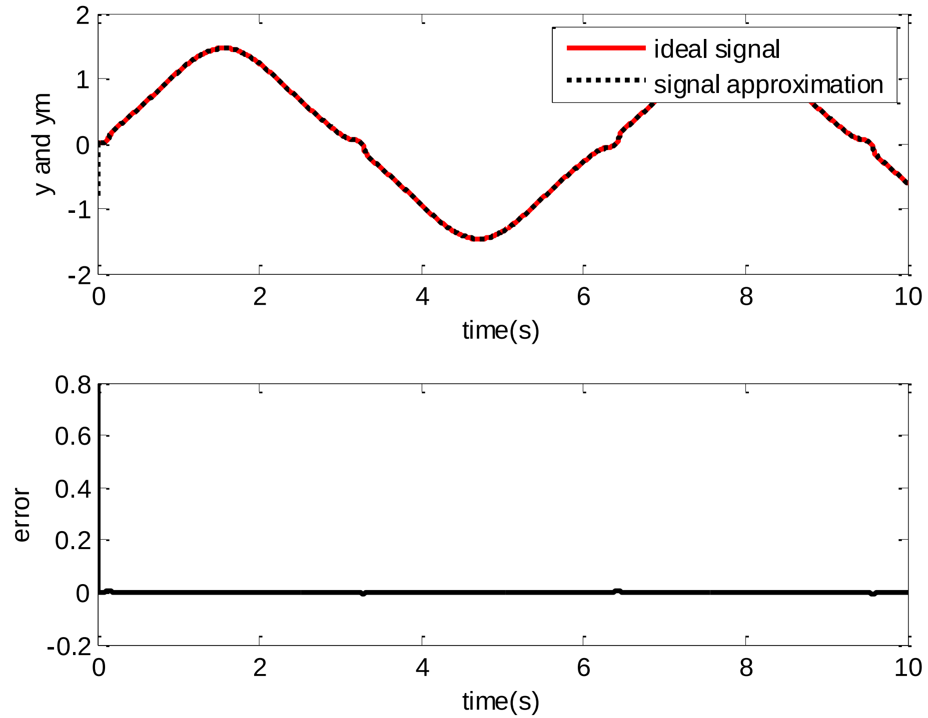
$$u(t) = sin(t), t = kT, T = 0.001$$





基于权值调整的RBF逼近

仿真实例



b =

3.0025
3.0011
3.0008
3.0032
2.9988

$$u(t) = sin(t), t = kT, T = 0.001$$

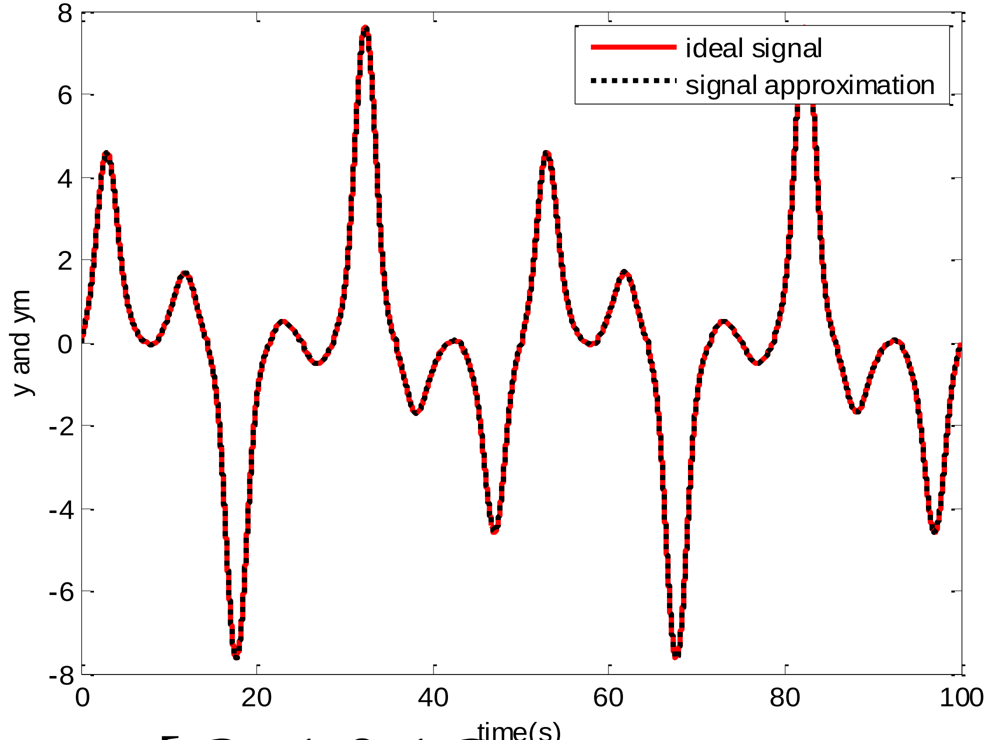
c =

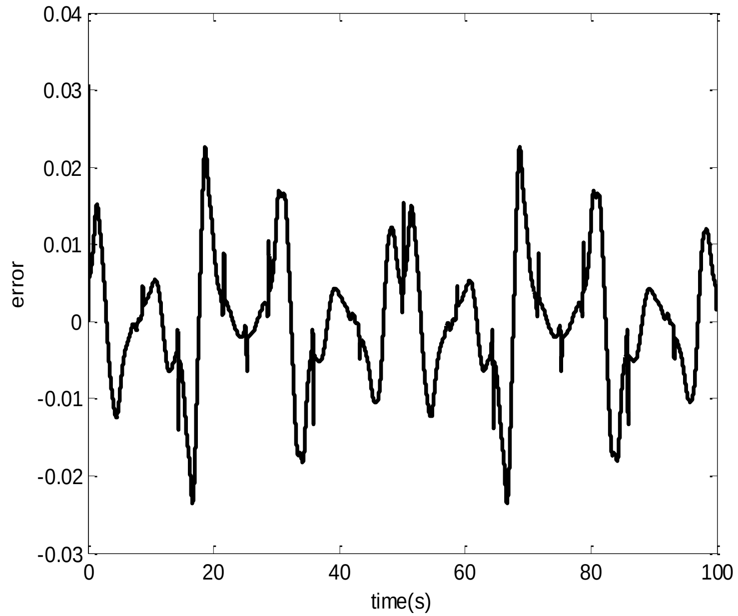
-0.9975	-0.4972	-0.0043	0.4966	1.0015
-0.9986	-0.4984	-0.0032	0.4979	1.0011

基于权值和基函数参数调整的RBF逼近

仿真实例

输入$u(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right)$





c=[-2 -1 0 1 2;
-10 -5 0 5 10];

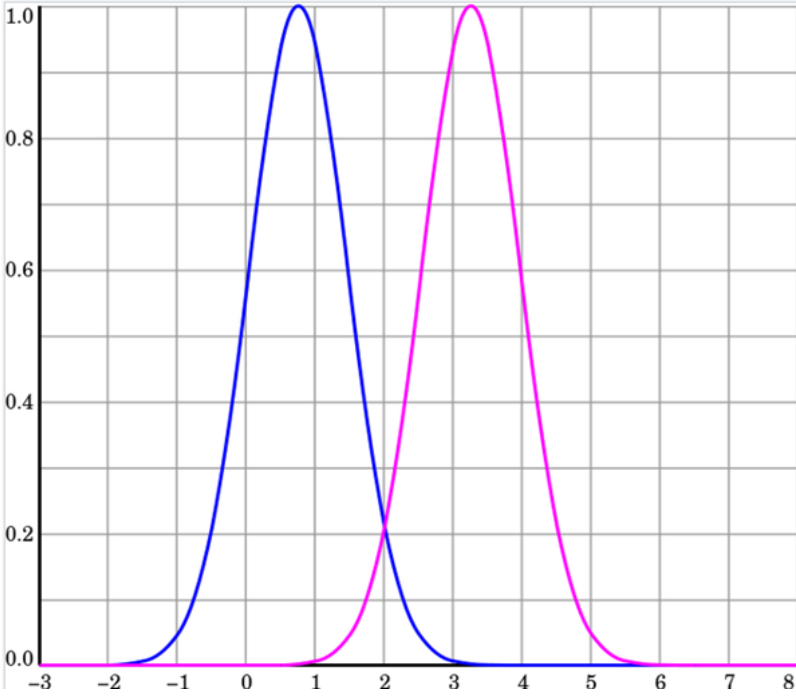
c = -2.0013 -1.0251 0.0172 1.0049 2.0003
-10.0194 -5.0680 0.0230 5.0512 9.9993

b =
3.0
3.0
3.0
3.0
3.0

b =
2.9628
2.9896
3.1813
2.9634
2.9997

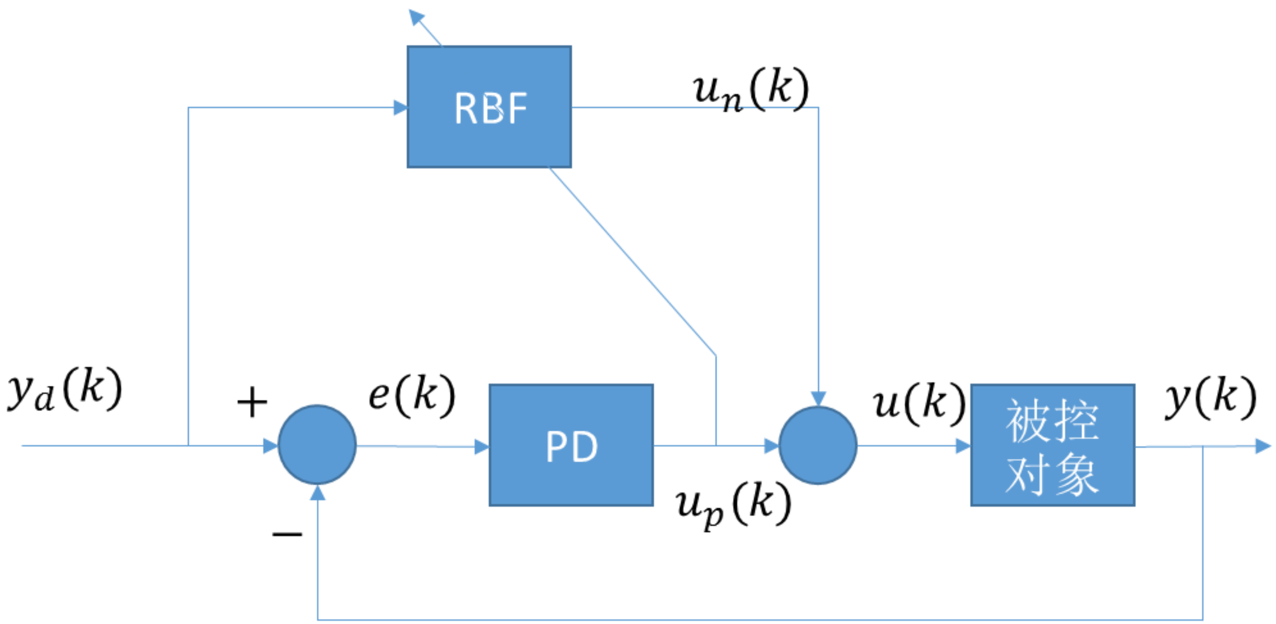
RBF神经网络

- 高斯基函数参数：
  - b_j为隐含层第j个神经元高斯基函数的宽度，b_j值越大，高斯基函数越宽，对输入的映射能力越大
  - c_j为隐含层第j个神经元高斯基函数中心点的坐标向量。c_j值离输入越近，高斯函数对输入越敏感，否则，高斯函数对输入越不敏感
  - 中心坐标向量c_j应使高斯基函数在有效的输入映射范围内，如RBF网络的输入为$[-3,+3]$，则c_j的取值范围应为$[-3,+3]$



基于RBF神经网络的监督控制

- 初始阶段采用PD反馈控制，然后过渡到神经网络控制
- 在控制过程中，如果出现极大误差，PD控制起主导作用，神经网络控制起调节作用



基于RBF神经网络的监督控制系统

基于RBF神经网络的监督控制

- RBF神经网络输出为$u_n(k) = h_1 w_1 + \cdots + h_m w_m$

总控制

$$u(k) = u_n(k) + u_p(k)$$

- 误差指标$E(k) = \frac{1}{2}(u_n(k) - u(k))^2$
- 采用梯度下降算法，网络权值学习算法为

$$\Delta w_j(k) = -\eta \frac{\partial E}{\partial w_j(k)} = \eta\big(u_n(k) - u(k)\big)h_j(k)$$

$$w(k) = w(k-1) + \Delta w(k) + \alpha\big(w(k-1) - w(k-2)\big)$$

$$\eta \in [0,1], \alpha \in [0,1]$$

仿真实例

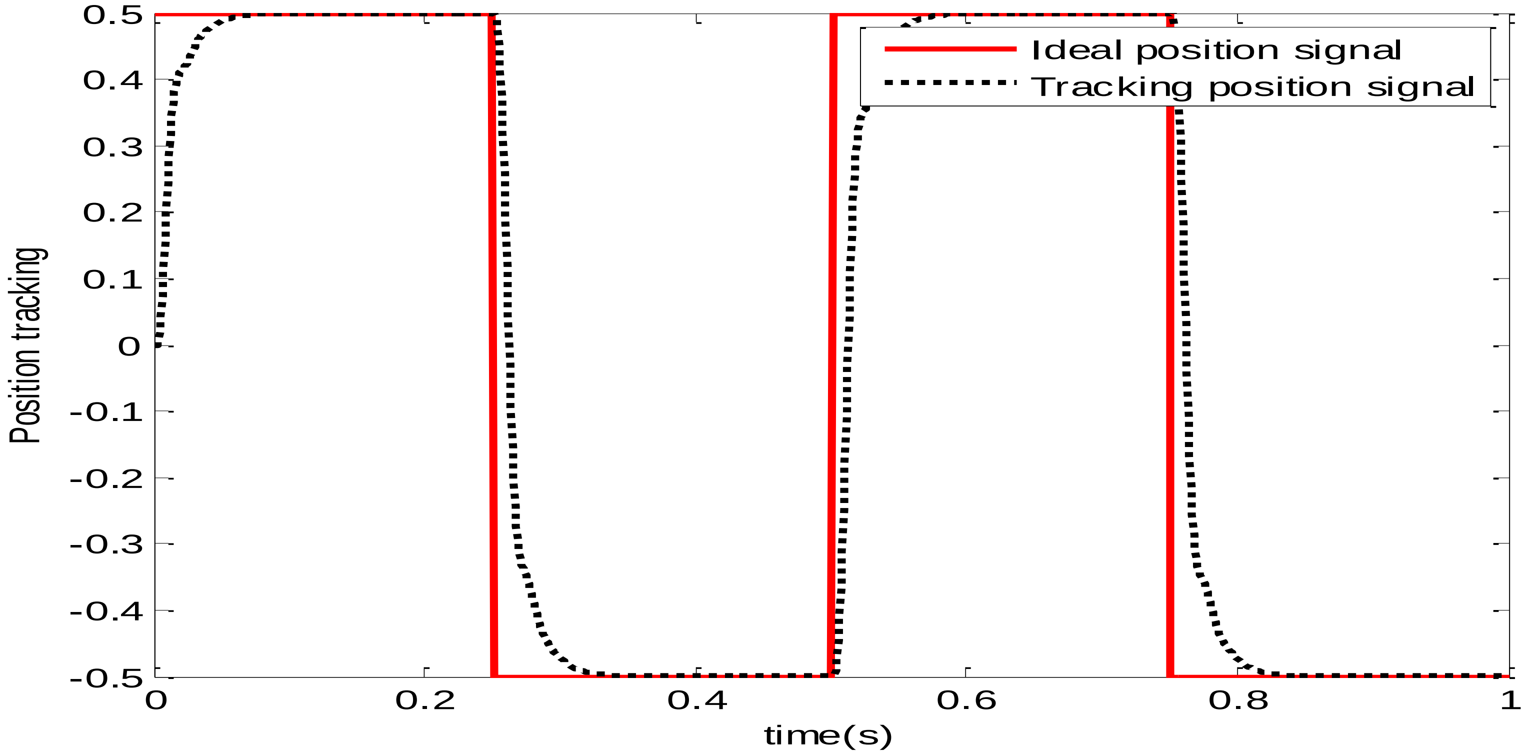
- 实例

$$G(s)=\frac{1000}{s^3+87.35s^2+10470s}$$

神经网络结构1-4-1

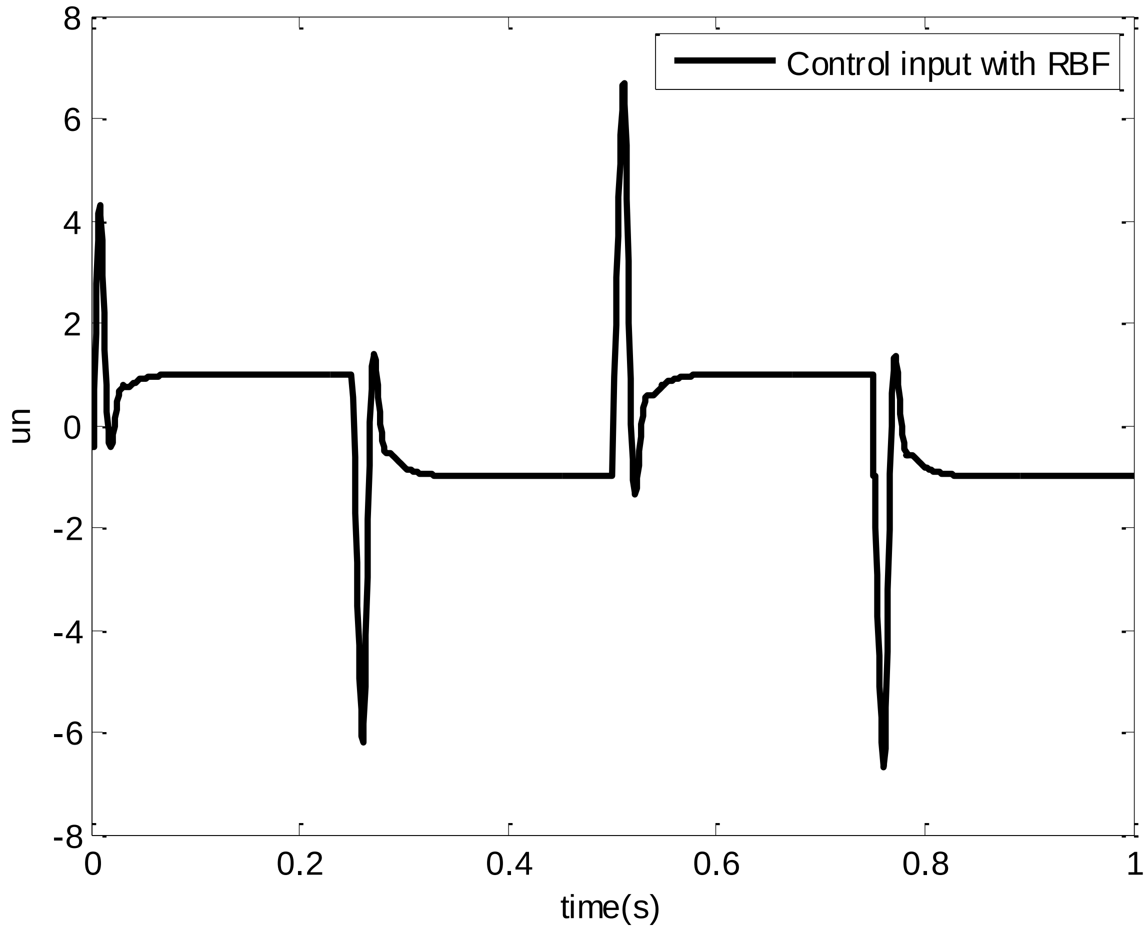
$$c=[-5\ \ -3\ 0\ 3\ 5]^T, \eta=0.30, \alpha=0.05$$

仿真实例


0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Position tracking
0
0.2
0.4
0.6
0.8
1
time(s)
Ideal position signal
Tracking position signal

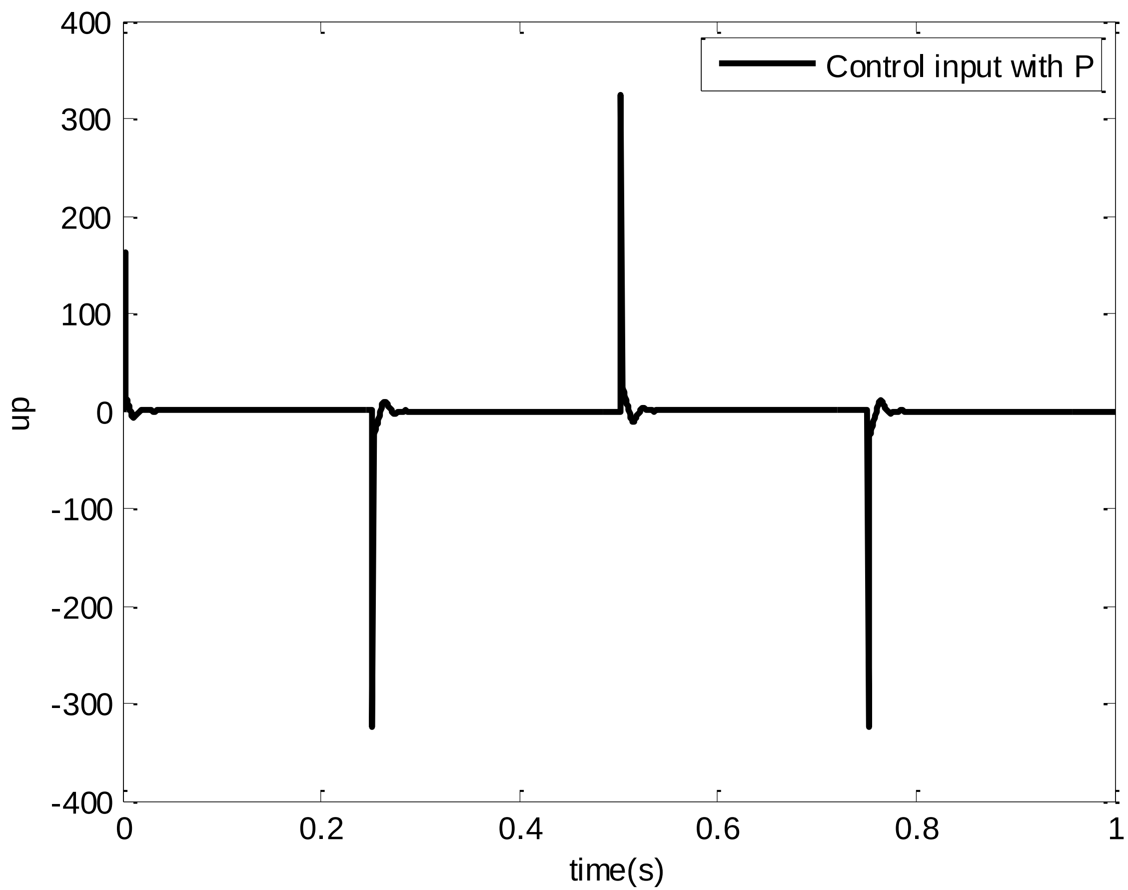
方波跟踪效果

仿真实例



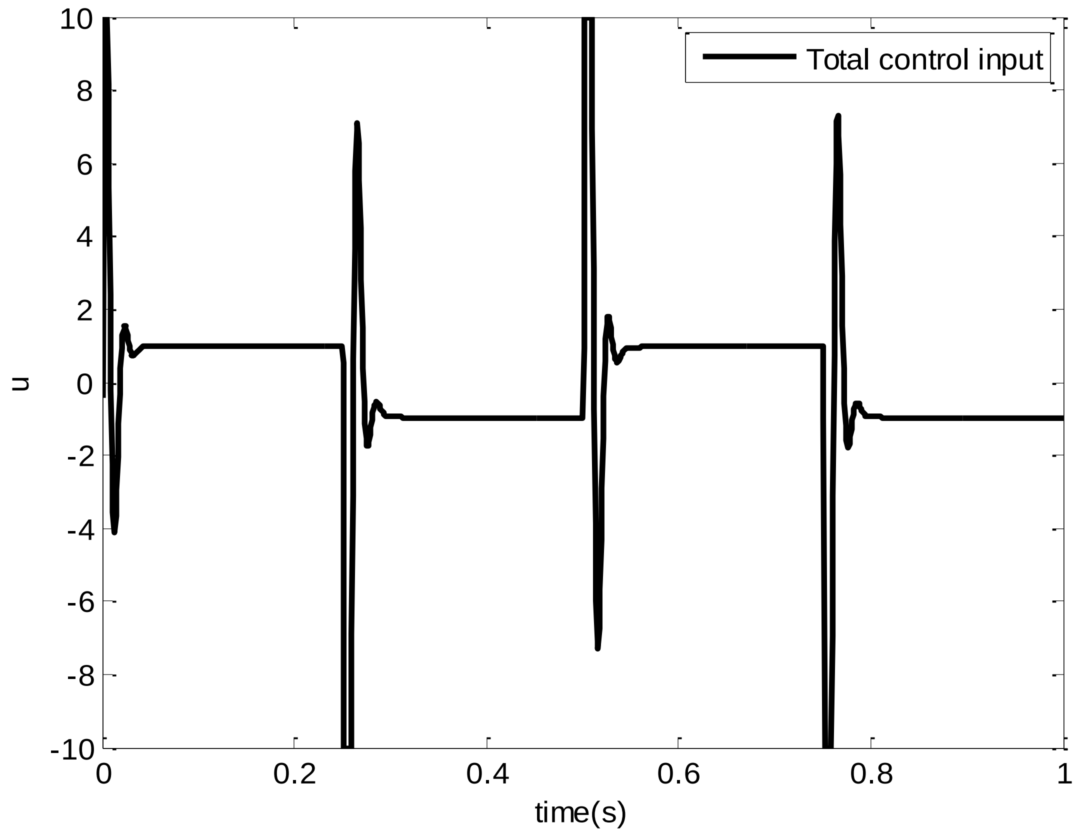
神经网络信号u_n

仿真实例



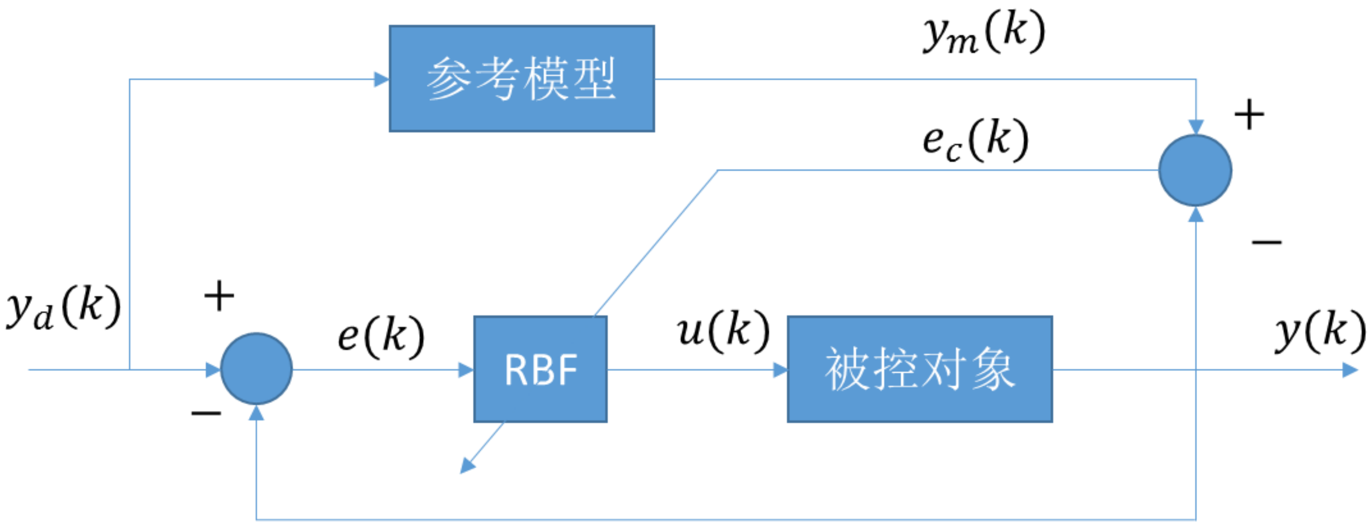
PD控制信号u_p

仿真实例



总控制信号u

RBF神经网络的模型参考自适应控制



直接模型参考自适应

RBF神经网络的模型参考自适应控制

$$e_c(k) = y_m(k) - y(k)$$

$$E = \frac{1}{2} e_c(k)^2$$

$$u(k) = h_1 w_1 + \cdots + h_m w_m$$

$$\Delta w_j(k) = -\eta \frac{\partial E(k)}{\partial w_j} = \eta e_c(k) \frac{\partial y(k)}{\partial u(k)} h_j$$

$$w_j(k) = w_j(k-1) + \Delta w_j(k) + \alpha\big(w(k-1) - w(k-2)\big)$$

$$\Delta b_j(k) = -\eta \frac{\partial E}{\partial b_j} = \eta e_c(k) \frac{\partial y(k)}{\partial u(k)} w_j h_j \frac{\left\| x - c_j \right\|^2}{b_j^3}$$

$$b_j(k) = b_j(k-1) + \Delta b_j(k) + \alpha\big(b_j(k-1) - b_j(k-2)\big)$$

$$\Delta c_{ji}(k) = -\eta \frac{\partial E}{\partial c_{ji}} = \eta e_c(k) \frac{\partial y(k)}{\partial u(k)} w_j h_j \frac{x_j - c_{ji}}{b_j^2}$$

$$c_{ji}(k) = c_{ji}(k-1) + \Delta c_{ji}(k) + \alpha\big(c_{ji}(k-1) - c_{ji}(k-2)\big)$$

$\frac{\partial y(k)}{\partial u(k)}$ 为Jacobo矩阵，表征系统输出对控制输入的灵敏度

仿真实例

离散被控对象

$$y(k)=\left(-0.10y(k-1)+u(k-1)\right)+\frac{u(k-1)}{1+y^2(k-1)}$$

采样周期Ts=1ms，参考模型

$$y_m(k)=0.6y_m(k-1)+y_d(k)$$

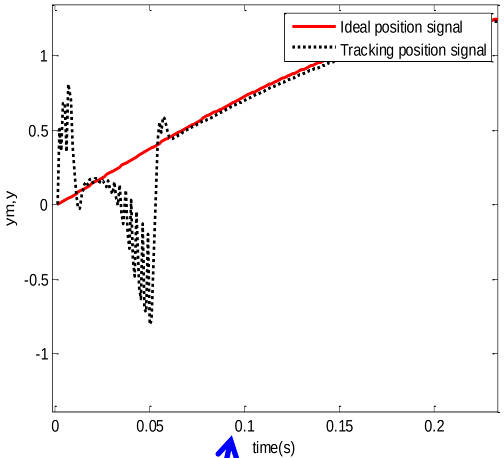
$$y_d(k)=0.5\sin(2\pi k*Ts)$$

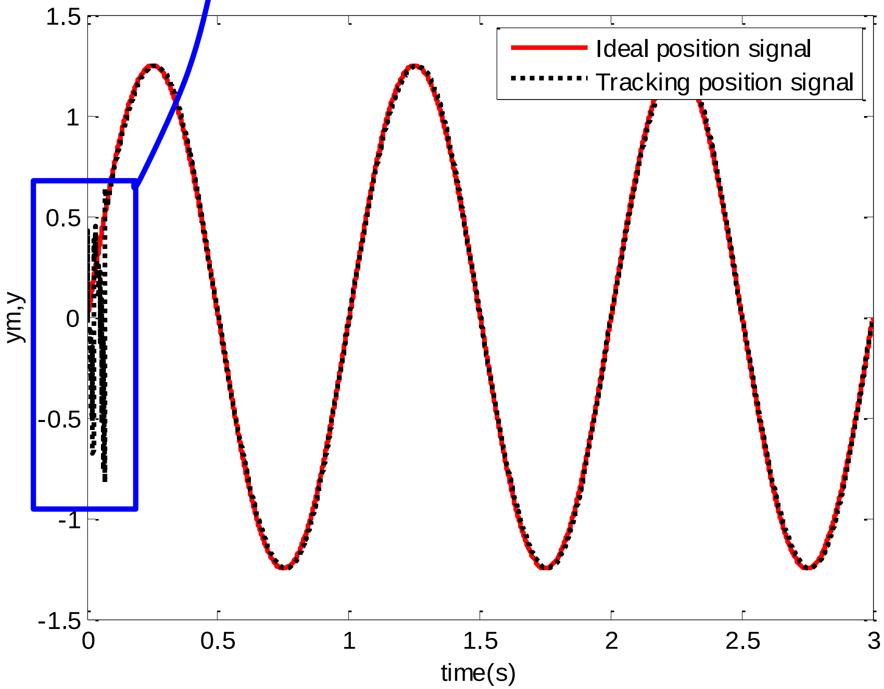
学习率$\eta=0.35$，动量因子$\alpha=0.05$

$$c=\begin{bmatrix}-3 & -2 & -1 & 1 & 2 & 3\\ -3 & -2 & -1 & 1 & 2 & 3\\ -3 & -2 & -1 & 1 & 2 & 3\end{bmatrix}^T$$

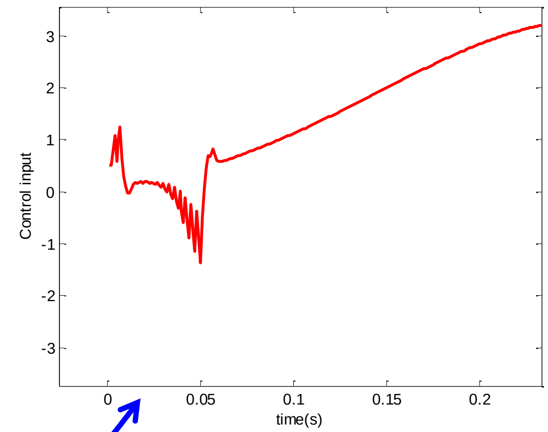
$$b=[2,2,2,2,2,2]^T$$

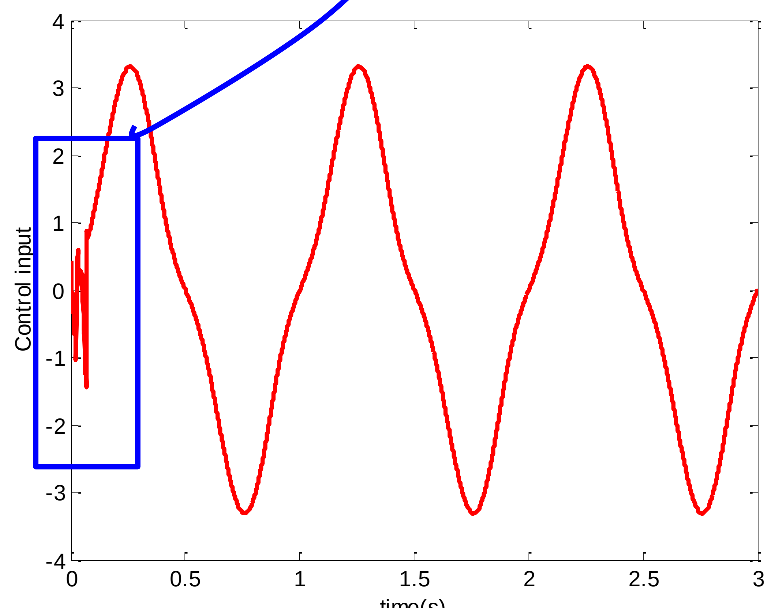
仿真实例





正弦跟踪





控制量

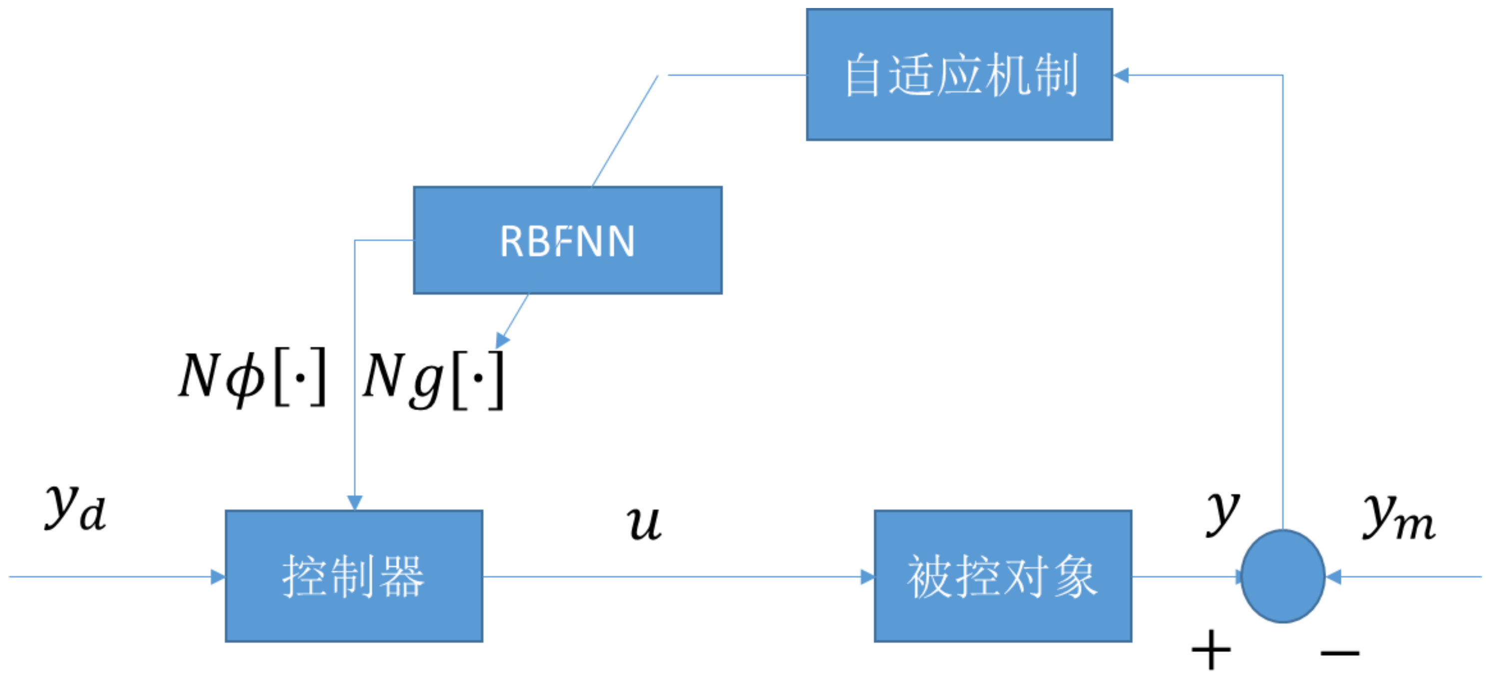
RBF自校正控制

考虑被控对象$y(k+1)=g[y(k)]+\phi[y(k)]u(k)$

- 若$g[\cdot]$和$\phi[\cdot]$已知，设计自校正控制器$u(k)=\frac{-g[\cdot]}{\phi[\cdot]}+\frac{y_d(k+1)}{\phi[\cdot]}$
- 若$g[\cdot]$和$\phi[\cdot]$未知，利用RBF神经网络逼近 $g[\cdot]$和$\phi[\cdot]$，得到估计值，记为$Ng[\cdot]$和$N\phi[\cdot]$，则自校正控制器可以设计为

$$u(k)=\frac{-Ng[\cdot]}{N\phi[\cdot]}+\frac{y_d(k+1)}{N\phi[\cdot]}$$

RBF自校正控制



基于RBF逼近的自适应控制——间接自校正控制

RBF自校正控制

- 分别用两个RBF神经网络逼近$g[\cdot]$和$\phi[\cdot]$，W和V分别是两个神经网络的权值，取$y(k)$为网络输入，径向基函数取为高斯函数

$$h_j = exp\left(-\frac{\left\|y(k)-c_j\right\|^2}{2b_j^2}\right)$$

两个网络的输出分别为

$$Ng(k) = h_1 w_1 + \cdots + h_m w_m$$
$$N\phi(k) = h_1 v_1 + \cdots + h_m v_m$$

基于RBF神经网络逼近的输出为

$$y_m(k) = Ng[y(k-1); W(k)] + N\phi[y(k-1); V(k)]u(k-1)$$

RBF自校正控制

$$E(k)=\frac{1}{2}(y(k)-y_m(k))^2$$

学习算法为

$$\Delta w_j(k)=-\eta_w\frac{\partial E(k)}{\partial w_j(k)}=-\eta_w\big(y(k)-y_m(k)\big)h_j(k)$$

$$\Delta v_j(k)=-\eta_v\frac{\partial E(k)}{\partial v_j(k)}=-\eta_v\big(y(k)-y_m(k)\big)h_j(k)u(k-1)$$

$$W(k)=W(k-1)+\Delta W(k)+\alpha(W(k-1)-W(k-2))$$

$$V(k)=V(k-1)+\Delta V(k)+\alpha(V(k-1)-V(k-2))$$

η_w,η_v为学习率,α为动量因子

仿真实例

被控系统

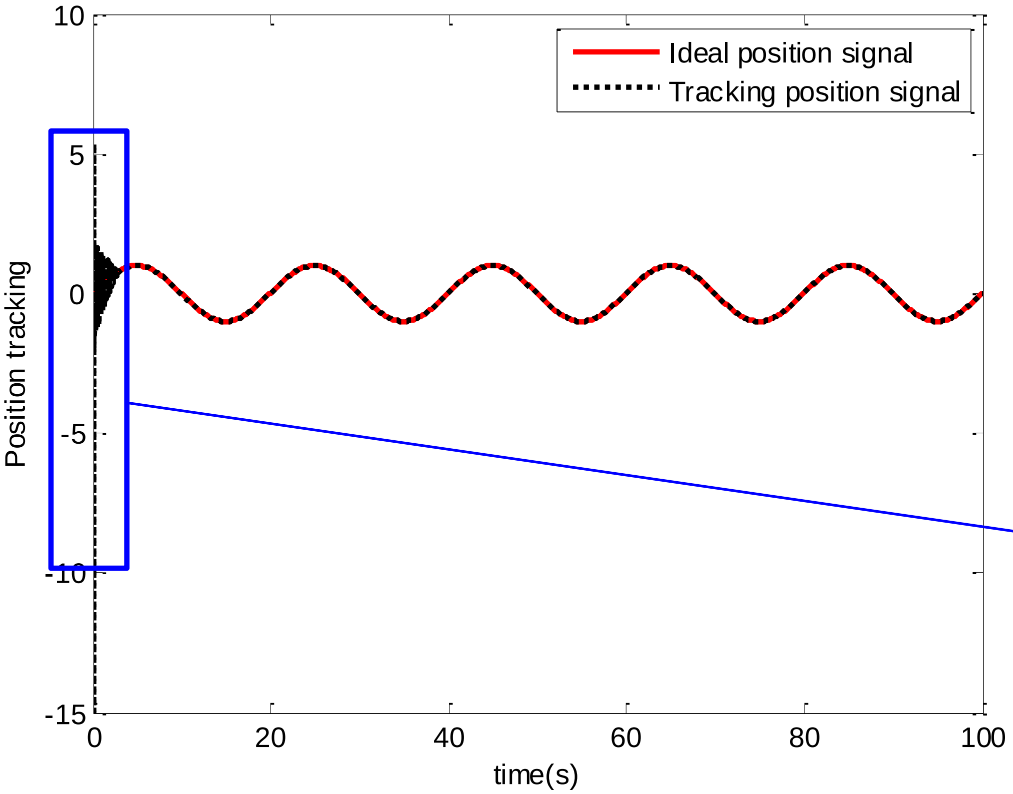
$$y(k)=0.8\sin\big(y(k-1)\big)+15u(k-1)$$

其中 $g[y(k)]=0.8\sin\big(y(k-1)\big),\phi[y(k)]=15$

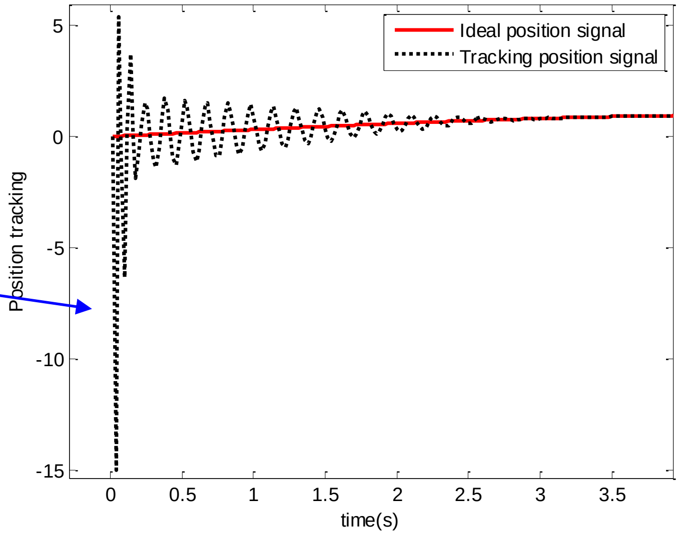
指令$y_d(t)=2\sin(0.1\pi t)$

- 1-6-1网络结构
- 初始权值和参数为
- $W=V=[0.5,0.5,0.5,0.5,0.5,0.5]^T$
- $c_j=[0.5,0.5,0.5,0.5,0.5,0.5]^T,b=[5,5,5,5,5,5]^T$
- $\eta_1=0.15,\eta_2=0.50,\alpha=0.05$

仿真实例

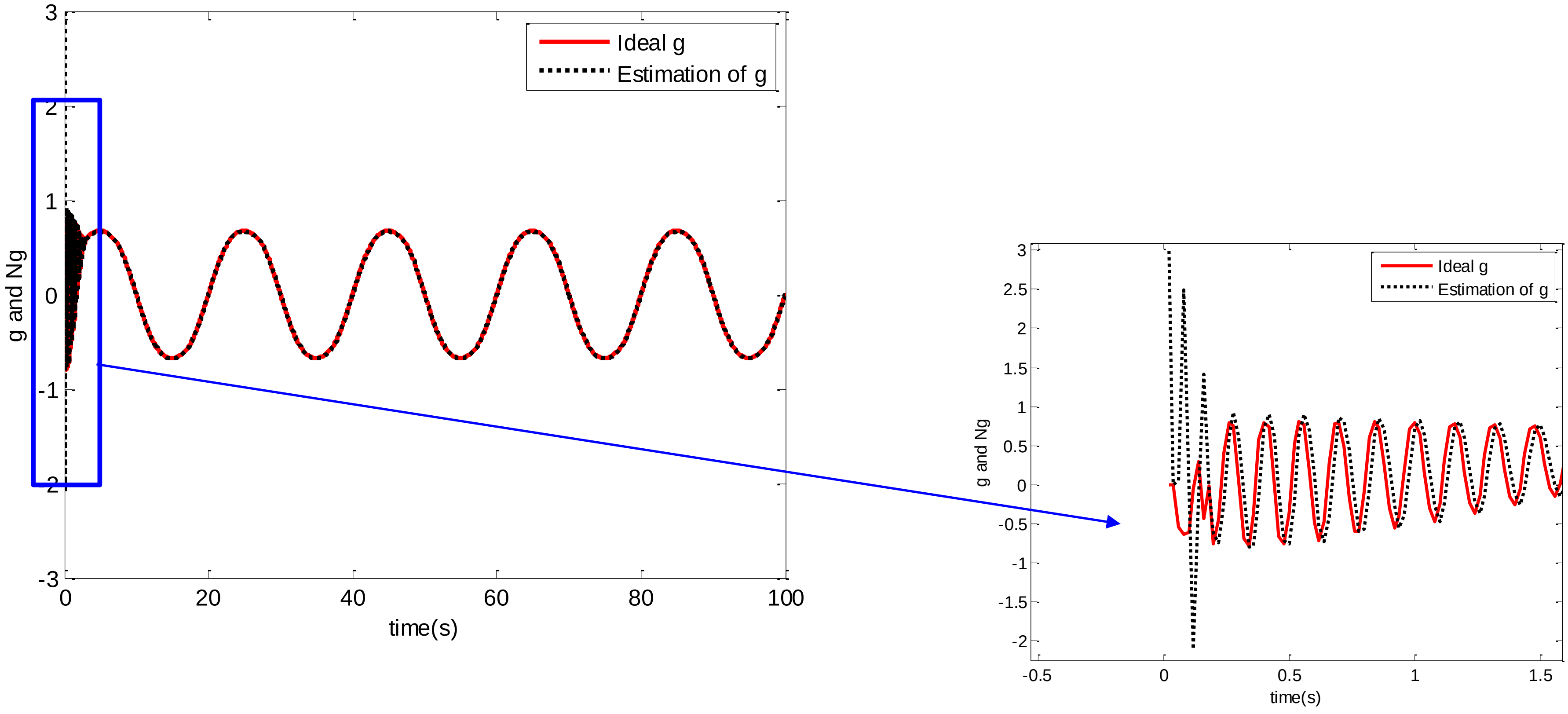


ts=0.02;



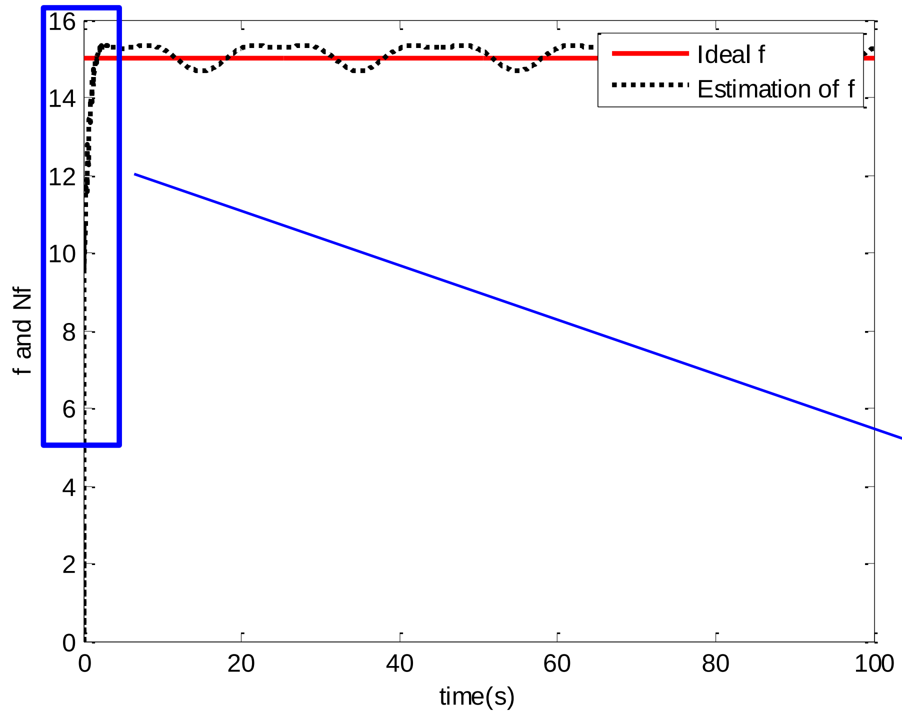
正弦指令跟踪

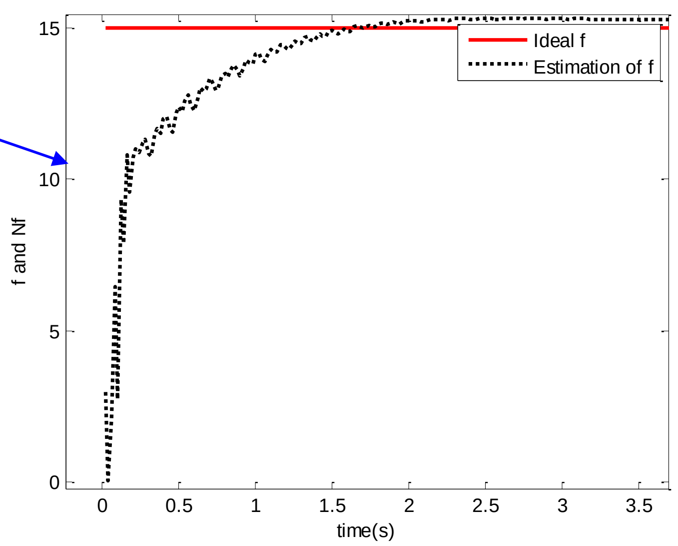
仿真实例



$g(x,t)$及其估计值$\hat{g}(x,t)$

仿真实例

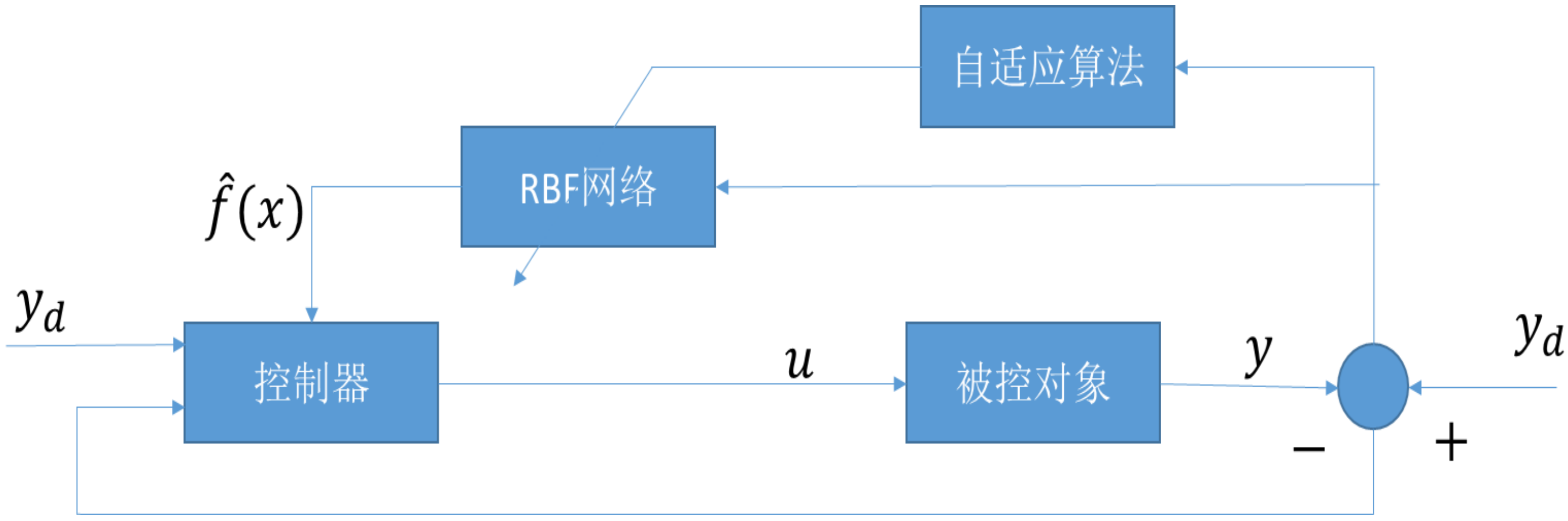




$f(x,t)$及其估计值$\hat{f}(x,t)$

自适应RBF神经网络控制

- 采用梯度下降法调整神经网络权值，易于陷入局部最优，且不能保证闭环系统稳定性
- 采用基于Lyapunov稳定性分析的在线自适应神经网络控制



RBF神经网络自适应控制系统

自适应RBF神经网络控制

考虑二阶非线性系统

$$\ddot{x} = f(x, \dot{x}) + g(x, \dot{x})u$$

令$x_1 = x, x_2 = \dot{x}, y = x_1$

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x_1, x_2) + g(x_1, x_2)u \\ y = x_1 \end{cases}$$

理想跟踪指令为y_d，则误差为

$$e = y_d - y = y_d - x_1$$

自适应RBF神经网络控制

设计理想控制律

$$u^* = \frac{1}{g(x)}\left[-f(x) + \ddot{y}_d + K^T E\right] \qquad K = \left(k_p, k_d\right)^T$$

可得到误差系统

$$\ddot{e} + k_p e + k_d \dot{e} = 0$$

特征多项式

$$s^2 + k_d s + k_p = 0$$

根在左半复平面 $t \to \infty, e(t) \to 0, \dot{e}(t) \to 0$

自适应RBF神经网络控制

利用RBF神经网络逼近未知函数f

$$f = W^{*T}h(x) + \epsilon$$

ϵ为网络逼近误差， W^*为理想权值

RBF神经网络的输出为$\hat{f}(x) = \widehat{W}^T h(x), x = [e, \dot{e}]^T$

$\widehat{W}$为理想权值W^*的估计

$$u = \frac{1}{g(x)}[-\hat{f}(x) + \ddot{y}_d + K^T E]$$

确定参数学习方法： $\widehat{W} \rightarrow W^*$

自适应RBF神经网络控制

闭环系统方程

$$\ddot{e} = -K^T E + [\hat{f}(x) - f(x)]$$

令$\Lambda = \begin{bmatrix} 0 & 1 \\ -k_p & -k_d \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, E = [e, \dot{e}]^T$

$$\dot{E} = \Lambda E + B[\hat{f}(x) - f(x)]$$

$$W^* = \arg\min_{W \in \Omega}[\sup|\hat{f}(x) - f(x)|]$$

定义$\omega = \hat{f}(x|W^*) - f(x)$, 闭环方程

$$\dot{E} = \Lambda E + B\{\left(\widehat{W} - W^*\right)^T h(x) + \omega\}$$

自适应RBF神经网络控制

- 定义Lyapunov函数

$$V=\frac{1}{2}E^TPE+\frac{1}{2\gamma}\left(\widehat{W}-W^*\right)^T\left(\widehat{W}-W^*\right)$$

γ为正常数，矩阵P为对称正定的且满足如下方程

$$\Lambda^TP+P\Lambda=-Q$$

其中$Q\geq 0$

取$V_1=\frac{1}{2}E^TPE$, $V_2=\frac{1}{2\gamma}\left(\widehat{W}-W^*\right)^T\left(\widehat{W}-W^*\right)$, $M=B\left\{\left(\widehat{W}-W^*\right)^Th(x)+\omega\right\}$

自适应RBF神经网络控制

$$\dot{V}_1 = -\frac{1}{2}E^TQE + \left(\widehat{W} - W^*\right)^T E^TPBh(x) + E^TPB\omega$$

$$\dot{V}_2 = \frac{1}{\gamma}\left(\widehat{W} - W^*\right)^T \dot{\widehat{W}}$$

$$\dot{V} = \dot{V}_1 + \dot{V}_2$$

$$= -\frac{1}{2}E^TQE + E^TPB\omega + \frac{1}{\gamma}\left(\widehat{W} - W^*\right)^T \left[\dot{\widehat{W}} + \gamma E^TPBh(x)\right]$$

$$\dot{\widehat{W}} = -\gamma E^TPBh(x)$$

$$\dot{V} = -\frac{1}{2}E^TQE + E^TPB\omega$$

$$\dot{V} \le 0 \Longrightarrow \|E\| \ge \frac{\lambda_{max}(PB)\omega}{\frac{1}{2}\lambda_{min}(Q)}$$

设计RBF使得ω充分小

仿真实例

- 实例

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -25x_2 + 133 \end{cases}$$

$$x = [x_1, x_2]^T, f(x) = -25x_2, g(x) = 133$$

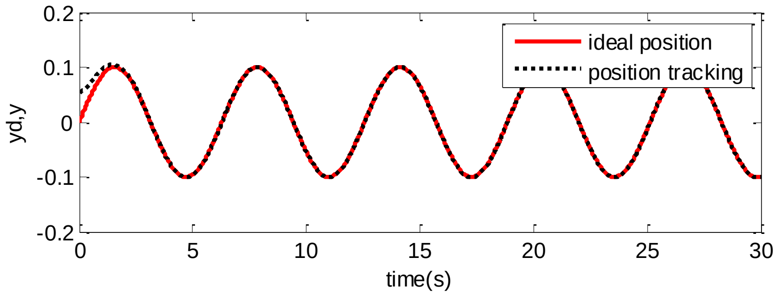
$$y_{1d}(t) = 0.1\sin(t), x_1(0) = \frac{\pi}{60}, x_2(0) = 0$$

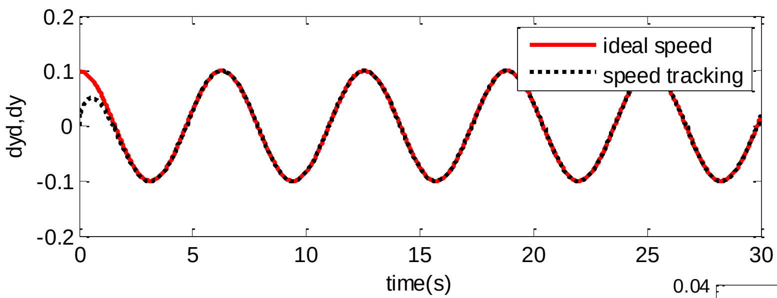
2-5-1结构 RBF, c= [-2 -1 0 1 2;

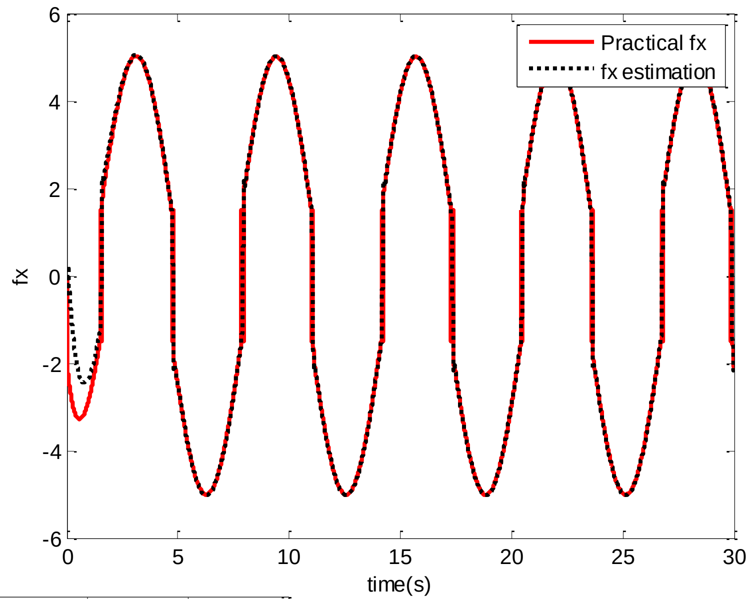
-2 -1 0 1 2];

- b=0.20; Q=[500 0;0 500];
- $k_d = 50, k_p = 30, \gamma = 1200$

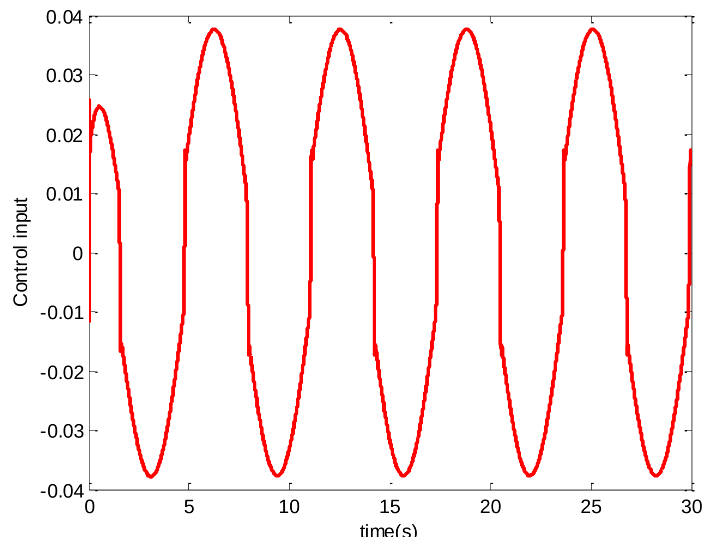
仿真实例1







Ts=0.01s



仿真实例2

- 单级倒立摆

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = f(x) + g(x)u$$

$$f(x) = \frac{g\sin x_1 - mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l(4/3 - m\cos^2 x_1/(m_c+m))}$$

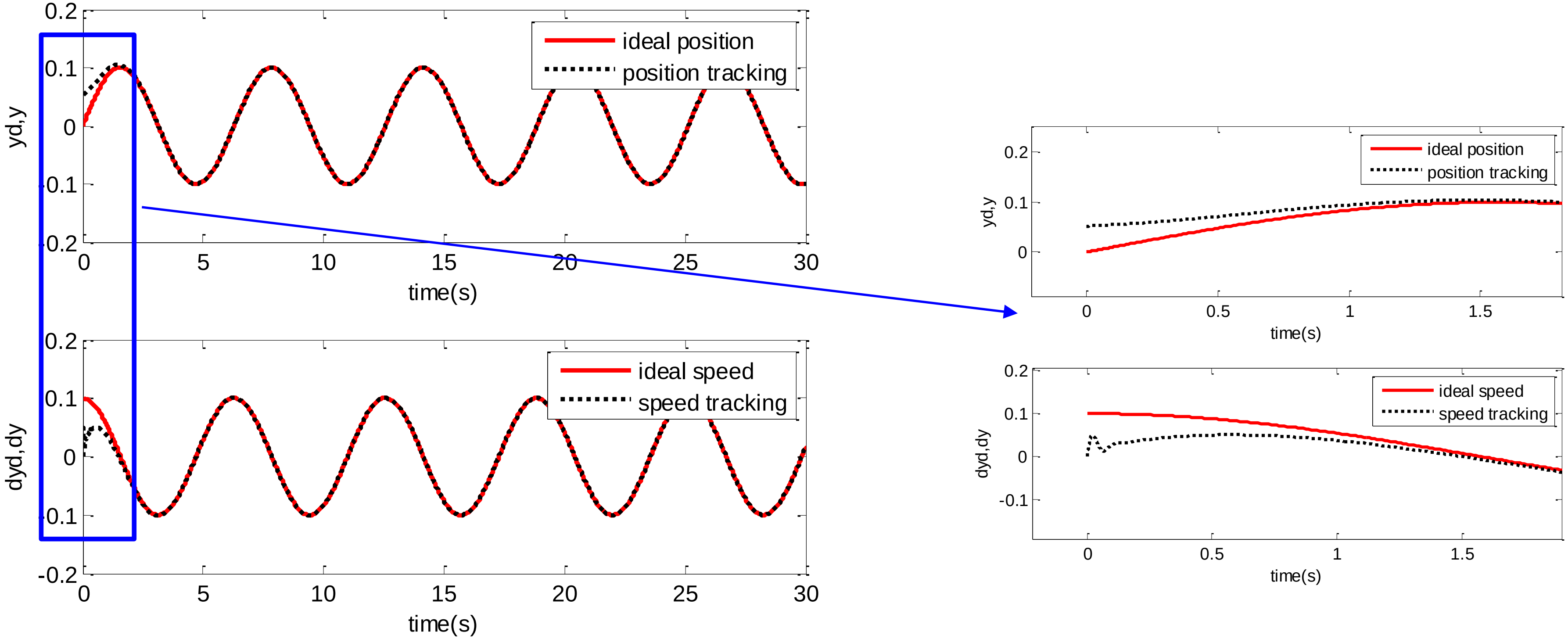
$$g(x) = \frac{\cos x_1/\,(m_c+m)}{l(4/3 - m\cos^2 x_1/(m_c+m))}$$

x_1和x_2分别是摆角和摆速； $g = 9.8m/s^2, m_c = 1kg$为小车质量, $m = 0.1kg$为摆质量； $l = 0.5m$为摆长一半； u为控制输入

$$y_{1d}(t) = 0.1\sin(t)$$

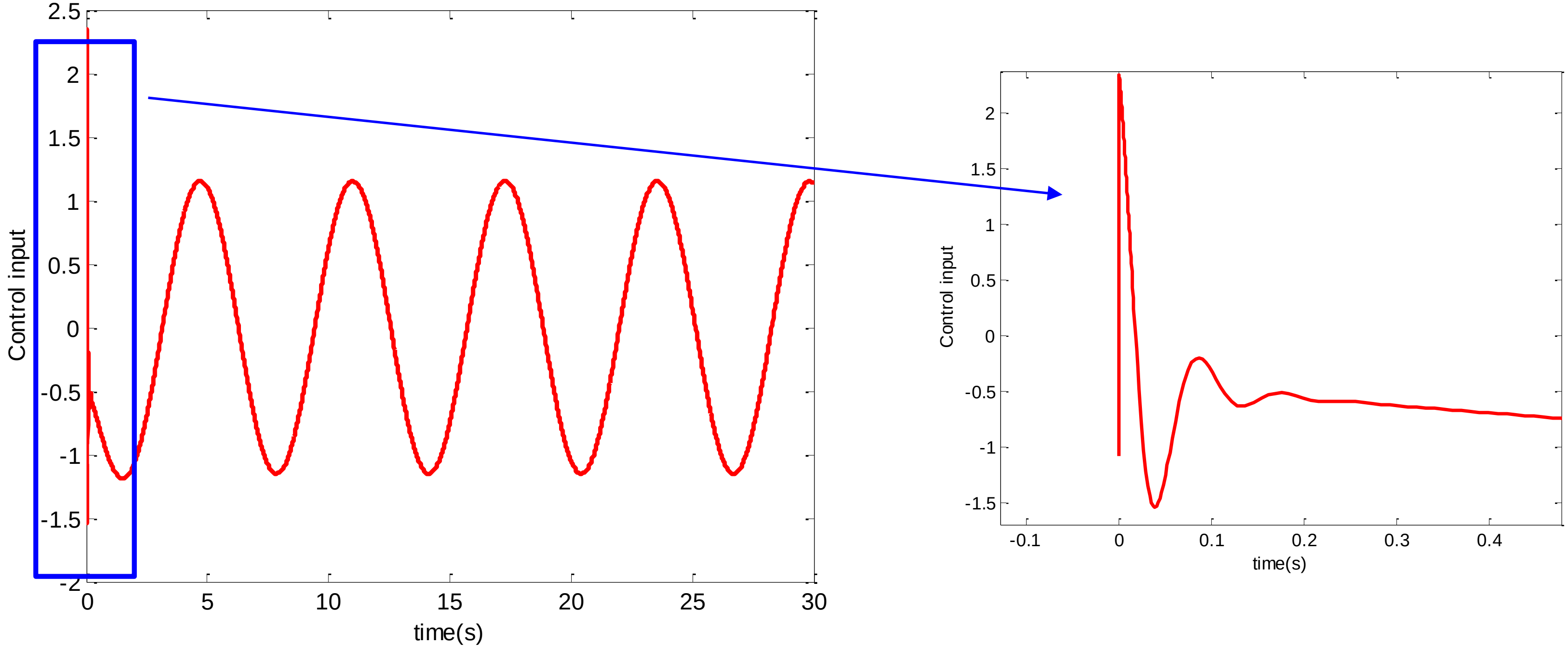
仿真实例2

- 结构参数同实例1



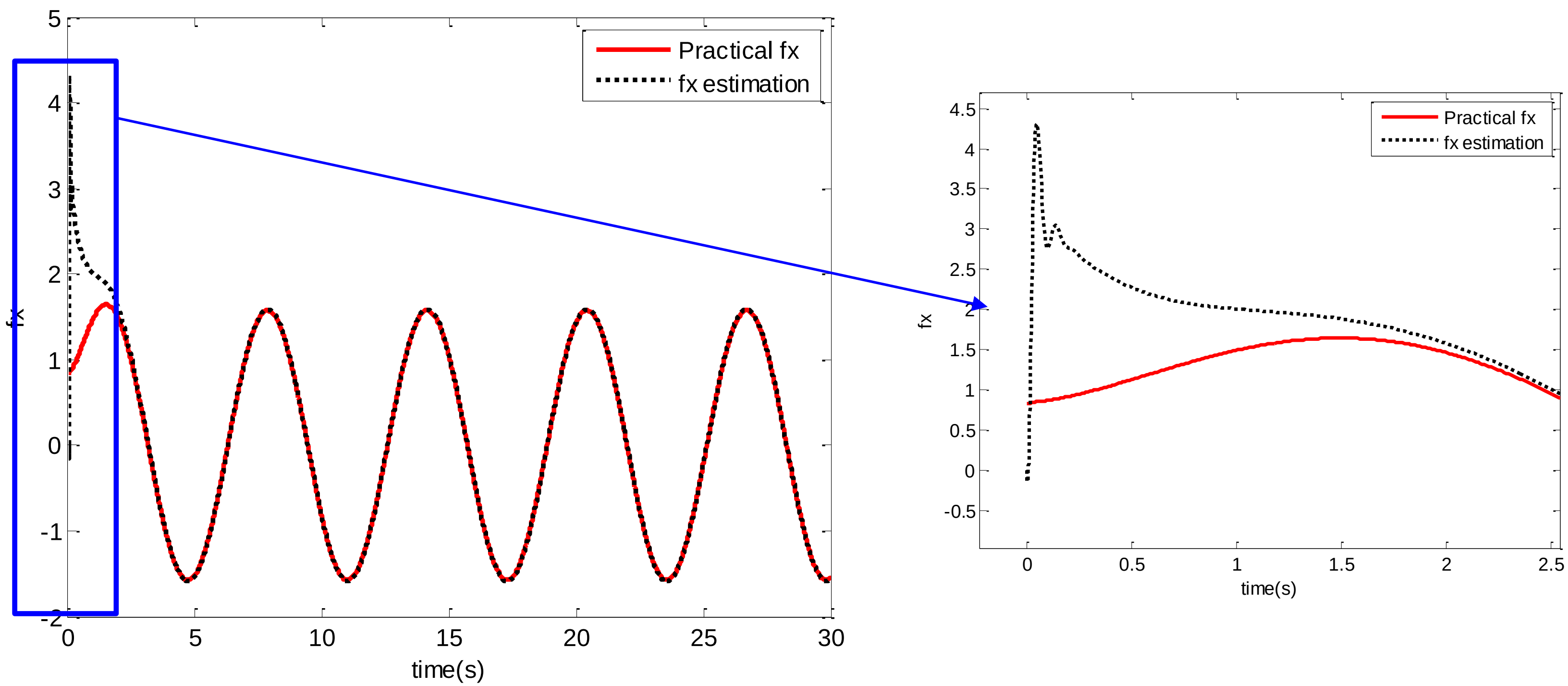
摆角摆速跟踪

仿真实例2



控制量

仿真实例2



$$f(x)\text{和}\hat{f}(x)$$

基于RBF神经网络补偿的机器人自适应控制

n关节机械臂，动态方程

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=\tau+d$$

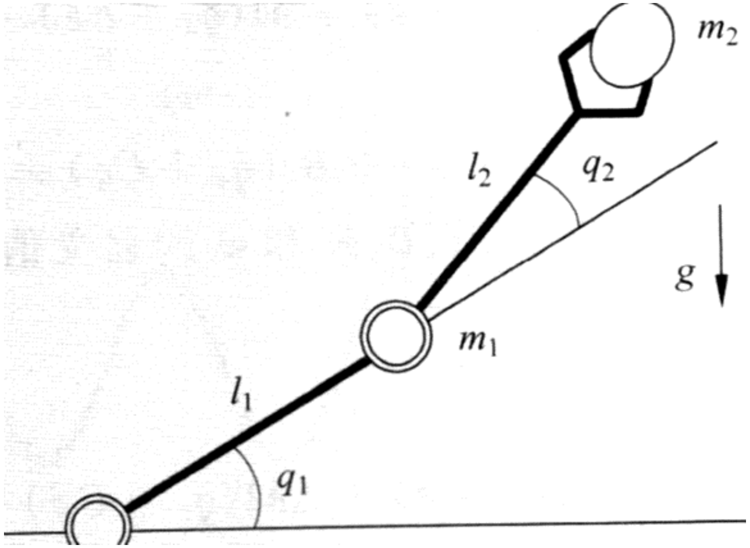
$M(q)$:正定惯性矩阵

$C(q,\dot{q})$：离心力和哥氏力项

$G(q)$：重力项

d:外界扰动

τ：控制输入



基于RBF神经网络补偿的机器人自适应控制

标称模型： $M_0(q), C_0(q,\dot{q}), G_0(q)$

$$M_0(q)\ddot{q} + C_0(q,\dot{q})\dot{q} + G_0(q) = \tau + f(q,\dot{q},\ddot{q})$$

$$\Delta M = M_0 - M, \Delta C = C_0 - C, \Delta G = G_0 - G$$

$$f(q,\dot{q},\ddot{q}) = \Delta M\ddot{q} + \Delta\dot{q} + \Delta G + d$$

若f已知，可设计控制律

$$\tau = M_0(q)\left(\ddot{q}_d - k_v\dot{e} - k_p e\right) + C_0(q,\dot{q})\dot{q} + G_0 - f(q,\dot{q},\ddot{q})$$

$$k_p = \begin{bmatrix} \alpha^2 & 0 \\ 0 & \alpha^2 \end{bmatrix}, k_v = \begin{bmatrix} 2\alpha & 0 \\ 0 & 2\beta \end{bmatrix}, \alpha > 0$$

闭环动态方程

$$\ddot{e} + k_v\dot{e} + k_p e = 0$$

q_d理想角度指令，$e = q - q_d, \dot{e} = \dot{q} - \dot{q}_d$

基于RBF神经网络补偿的机器人自适应控制

- 在线辨识机器人模型误差
- 保证闭环系统稳定性

利用RBF估计$f(\cdot)$，存在理想权值向量w^*

$$max\|f-\hat{f}^*\| \le \epsilon_0$$

$$w^* = \arg\min_{w\in\Omega}\{\sup_x\|f-\hat{f}^*\|\}$$

定义

$$\eta = f - \hat{f}^*$$

假定$\eta_0 = \sup\|f-\hat{f}^*\|$有界

$$\hat{f}^* = w^{*T}h(x)$$

基于RBF神经网络补偿的机器人自适应控制

设计如下控制器

$$\tau = M_0(q)\left(\ddot{q}_d - k_v\dot{e} - k_p e\right) + C_0(q,\dot{q})\dot{q} + G_0 - \hat{f}$$

$$\hat{f} = \widehat{w}^T h(x), \|w^*\|_F \leq w_{max}$$

$\widehat{w}$估计w^*

误差动态方程

$$\ddot{e} + k_v\dot{e} + k_p e = M_0^{-1}(q)(f(\cdot) - \hat{f}(\cdot))$$

基于RBF神经网络补偿的机器人自适应控制

$$\dot{x} = Ax + B\{f(\cdot) - \hat{f}(\cdot)\}$$

$$x = (e\ \dot{e})^T, A = \begin{bmatrix} 0 & I \\ -k_p & -k_v \end{bmatrix}, B = \begin{bmatrix} 0 \\ M_0^{-1}(q) \end{bmatrix}$$

$$f(\cdot) - \hat{f}(\cdot) = f(\cdot) - \hat{f}^*(\cdot) + \hat{f}^*(\cdot) - \hat{f}(\cdot)$$
$$= \eta + w^{*T}h - \widehat{w}^T h = \eta - \widetilde{w}^T h$$
$$\widetilde{w} = \widehat{w} - w^*, \eta = f(\cdot) - \hat{f}^*(\cdot)$$

闭环系统方程

$$\dot{x} = Ax + B(\eta - \widetilde{w}^T h)$$

基于RBF神经网络补偿的机器人自适应控制

定义Lyapunov函数

$$V=\frac{1}{2}x^TPx+\frac{1}{2\gamma}\|\tilde{w}\|^2$$

$$\gamma>0, PA+A^TP=-Q, Q\geq 0$$

定义$\|R\|^2=\sum_{i,j}|r_{ij}|^2=tr(RR^T)=tr(R^TR)$

$$\dot{V}=\frac{1}{2}[x^TP\dot{x}+\dot{x}^TPx]+\frac{1}{\gamma}tr\left(\dot{\tilde{w}}^T\tilde{w}\right)$$

$$=\frac{1}{2}x^TQx+\eta^TB^TPx-h^T\tilde{w}B^TPx+\frac{1}{\gamma}tr\left(\dot{\tilde{w}}^T\tilde{w}\right)$$

基于RBF神经网络补偿的机器人自适应控制

$$h^T\tilde{w}B^TPx = x^TPB\tilde{w}^Th, \eta^TB^TPx = x^TPB\eta$$

由于

$$h^T\tilde{w}B^TPx = \mathrm{tr}[B^TPxh^T\tilde{w}]$$

整理有

$$\dot{V} = -\frac{1}{2}x^TQx + \frac{1}{\gamma}tr\left(-\gamma B^TPxh^T\tilde{w} + \dot{\tilde{w}}^T\tilde{w}\right) + \eta^TB^TPx$$

基于RBF神经网络补偿的机器人自适应控制

■ 自适应律设计方法1

$$\dot{\hat{w}} = \gamma h x^T P B$$

$$\dot{\tilde{w}} = \dot{\hat{w}}$$

$$\Rightarrow \dot{V} = -\frac{1}{2}x^T Q x + \eta^T B^T P x$$

$$\dot{V} \leq -\frac{1}{2}\lambda_{min}(Q)\|x\|^2 + \|x\|\|\eta_0\|\left\|M_0^{-1}(q)\right\|\lambda_{max}(P)$$

$$= -\frac{1}{2}\|x\|\left[\lambda_{min}(Q)\|x\| - 2\|\eta_0\|\left\|M_0^{-1}(q)\right\|\lambda_{max}(P)\right]$$

$$\left(\|\eta^T\| \leq \|\eta_0\|, \|B\| = \left\|M_0^{-1}(q)\right\|\right)$$

基于RBF神经网络补偿的机器人自适应控制

- 自适应律设计方法1

$$\dot{V} \le 0 \Rightarrow \lambda_{min}(Q) \ge \frac{2\left\|M_0^{-1}(q)\right\|\lambda_{max}(P)}{\|x\|}\|\eta_0\|$$

$$\Rightarrow \|x\| \ge \frac{2\left\|M_0^{-1}(q)\right\|\lambda_{max}(P)}{\lambda_{min}(Q)}$$

基于RBF神经网络补偿的机器人自适应控制

■ 自适应律设计方法2

$$\dot{\hat{w}} = \gamma h x^T P B + k_1 \gamma \|x\| \hat{w}, k_1 > 0$$

$$\Rightarrow \dot{V} = -\frac{1}{2} x^T Q x + k_1 \gamma \|x\| tr(\hat{w}^T \tilde{w}) + \eta^T B^T P x$$

F范数性质 $\mathrm{tr}[\tilde{x}^T(x - \tilde{x})] \leq \|\tilde{x}\|_F \|x\|_F - \|\tilde{x}\|_F^2$

$$\mathrm{tr}(\hat{w}^T \tilde{w}) = \mathrm{tr}(\tilde{w}^T \hat{w}) = \mathrm{tr}[\tilde{w}^T(\tilde{w} + w^*)] \leq \|\tilde{w}\|_F \|W^*\|_F - \|\tilde{w}\|_F^2$$

基于RBF神经网络补偿的机器人自适应控制

$$\dot{V} \le -\|x\|\left(\frac{1}{2}\lambda_{min}(Q)\|x\| + k_1\left(\|\tilde{w}\|_F - \frac{\omega_{max}}{2}\right)^2 - \frac{k_1}{4}\omega_{max}^2 - \|\eta_0\|\lambda_{max}(P)\right)$$

为使$\dot{V} \le 0$，选择参数使

$$\omega_{max} = \|w^*\|_F$$

$$\frac{1}{2}\lambda_{min}(Q)\|x\| \ge \|\eta_0\|\lambda_{max}(P) + \frac{k_1}{4}\omega_{max}^2$$

或

$$k_1\left(\|\tilde{w}\|_F - \frac{\omega_{max}}{2}\right)^2 \ge \|\eta_0\|\lambda_{max}(P) + \frac{k_1}{4}\omega_{max}^2$$

从而$\|x\| \ge \frac{2}{\lambda_{min}(Q)}\left(\|\eta_0\|\lambda_{max}(P) + \frac{k_1}{4}\omega_{max}^2\right)$

或$\|\tilde{w}\|_F \ge \frac{\omega_{max}}{2} + \sqrt{\frac{1}{k_1}\left(\|\eta_0\|\lambda_{max}(P) + \frac{k_1}{4}\omega_{max}^2\right)}$

仿真实例

■ 双关节机械臂动力学模型

$$M(q)=\begin{pmatrix} v+q_{01}+2\gamma\cos q_2 & q_{02}+\cos q_2 \\ q_{02}+\cos q_2 & q_{02} \end{pmatrix}$$

$$C(q,\dot{q})=\begin{pmatrix} -q_{02}\dot{q}_2\sin q_2 & -q_{02}(\dot{q}_1+\dot{q}_2)\sin q_2 \\ q_{02}\dot{q}_1\sin q_2 & 0 \end{pmatrix}$$

$$G(q)=\begin{pmatrix} 15g\cos q_1+8.75g\cos(q_1+q_2) \\ 8.75g\cos(q_1+q_2) \end{pmatrix}$$

$$v=13.3, q_{01}=8.98, q_{02}=8.75, g=9.8$$

外界干扰 $d=d_1+d_2\|e\|+d_3\|\dot{e}\|, d_1=2, d_2=3, d_3=6$

仿真实例

关节角和角速度的期望跟踪指令

$$\begin{cases} q_{01} = 1 + 0.2\sin 0.5\pi\mathrm{t} \\ q_{02} = 1 - 0.2\cos 0.5\pi t \end{cases}$$

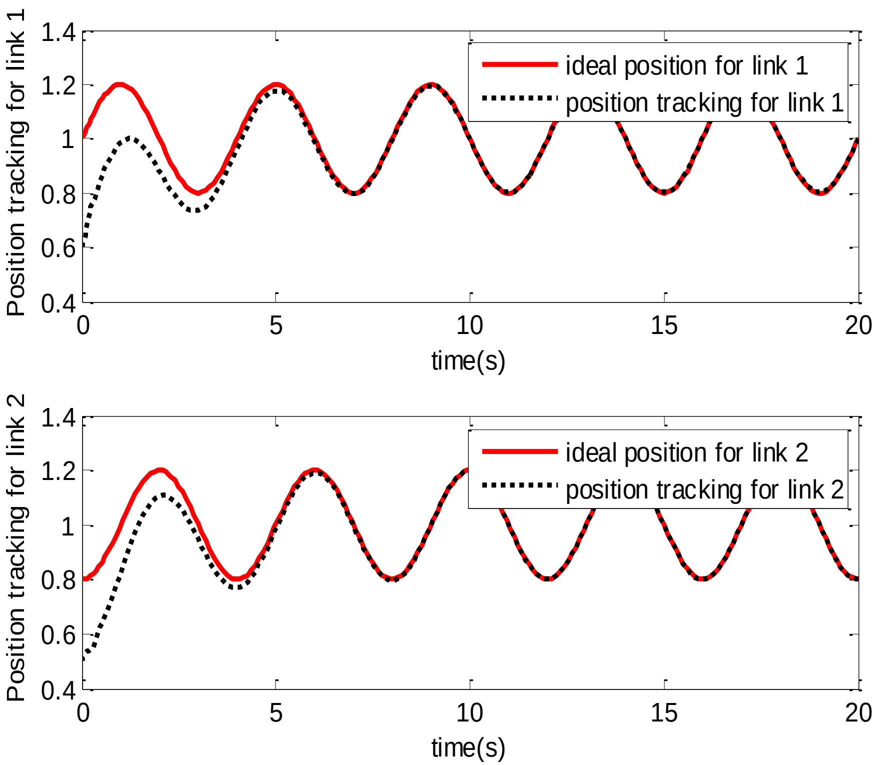
初始状态$[q_1, q_2, q_3, q_4]^T = [0.6,0.3,0.5,0.5\ ]^T$

$$\Delta M = 0.2M, \Delta C = 0.2C, \Delta G = 0.2G$$

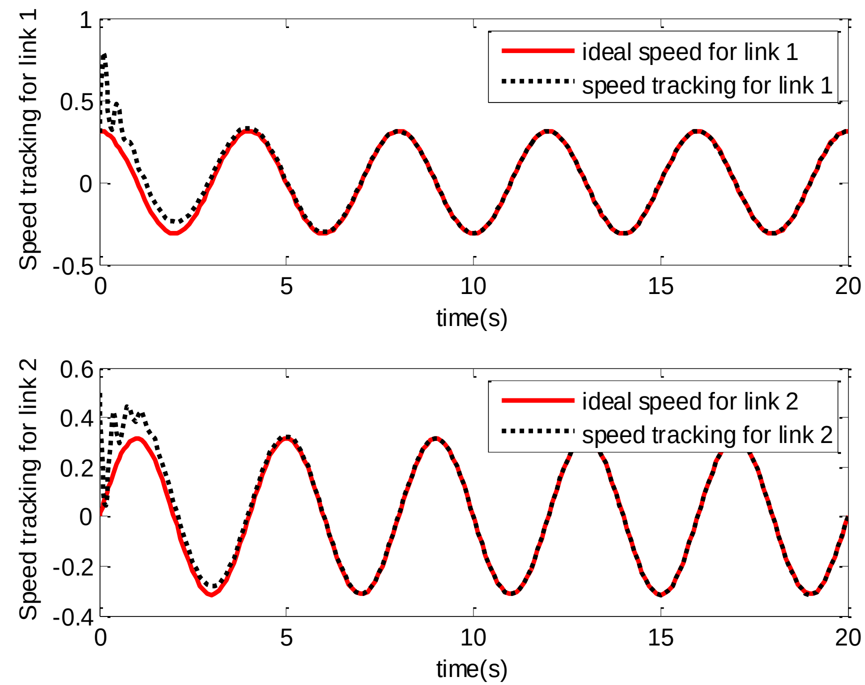
$$Q = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$$

$$\alpha = 3, \gamma = 20, k_1 = 0.001$$

仿真实例

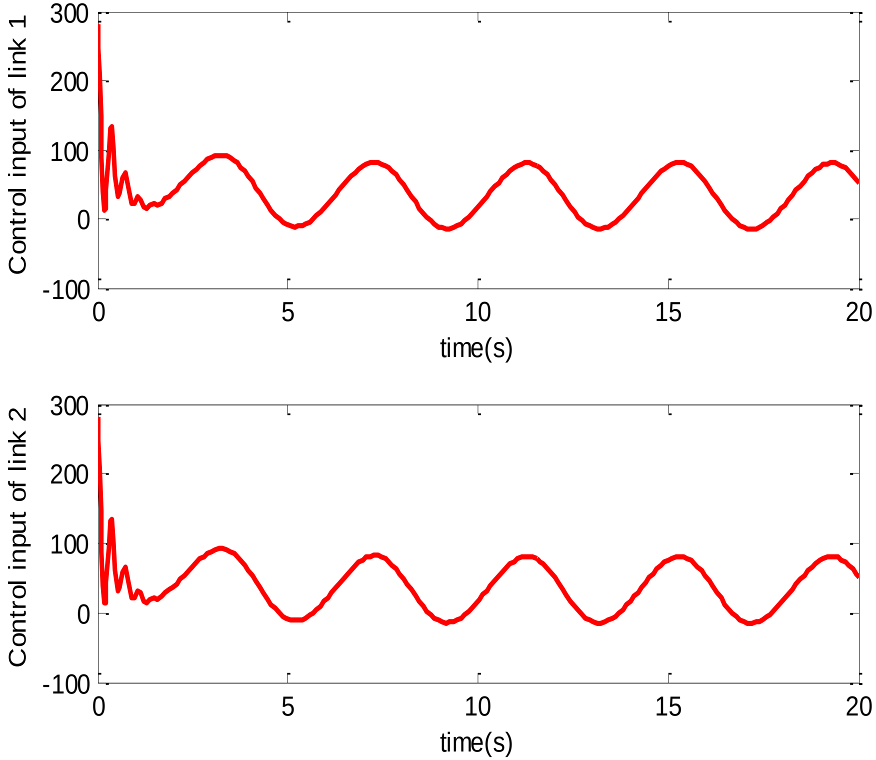


关节1、2角度跟踪

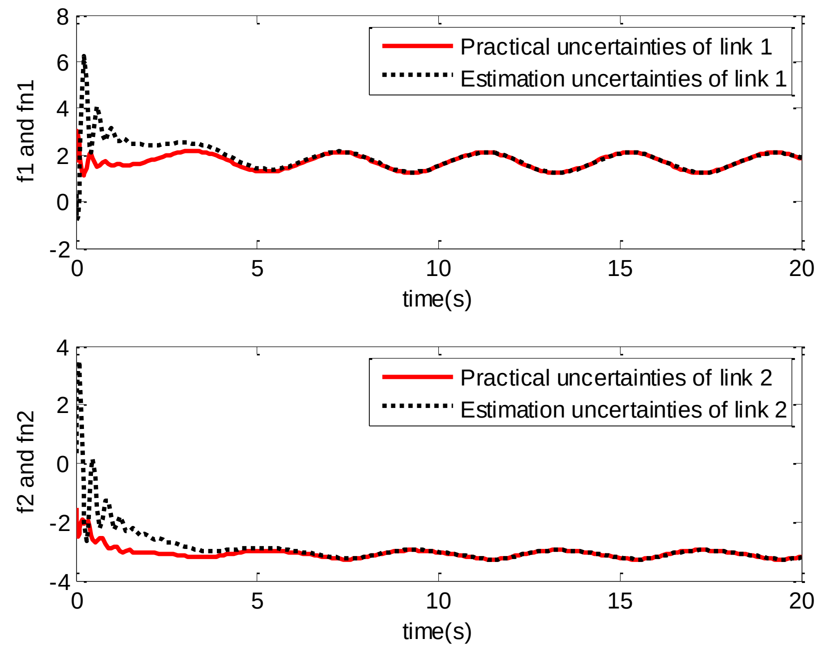


关节1、2角速度跟踪

仿真实例



关节1、2控制输入



关节1、2的$f(x)$及其逼近

离散系统的RBF网络控制

- 考虑离散非线性系统

$$y(k+1)=f(x(k))+u(k)$$

$$x(k)=[y(k)\quad y(k-1)\quad \cdots \quad y(k-n+1)]^T$$

控制任务 $y(k)$跟踪$y_d(k)$

$$e(k)=y(k)-y_d(k)$$

- 经典控制器设计

若$f(x(k))$已知，则设计控制律

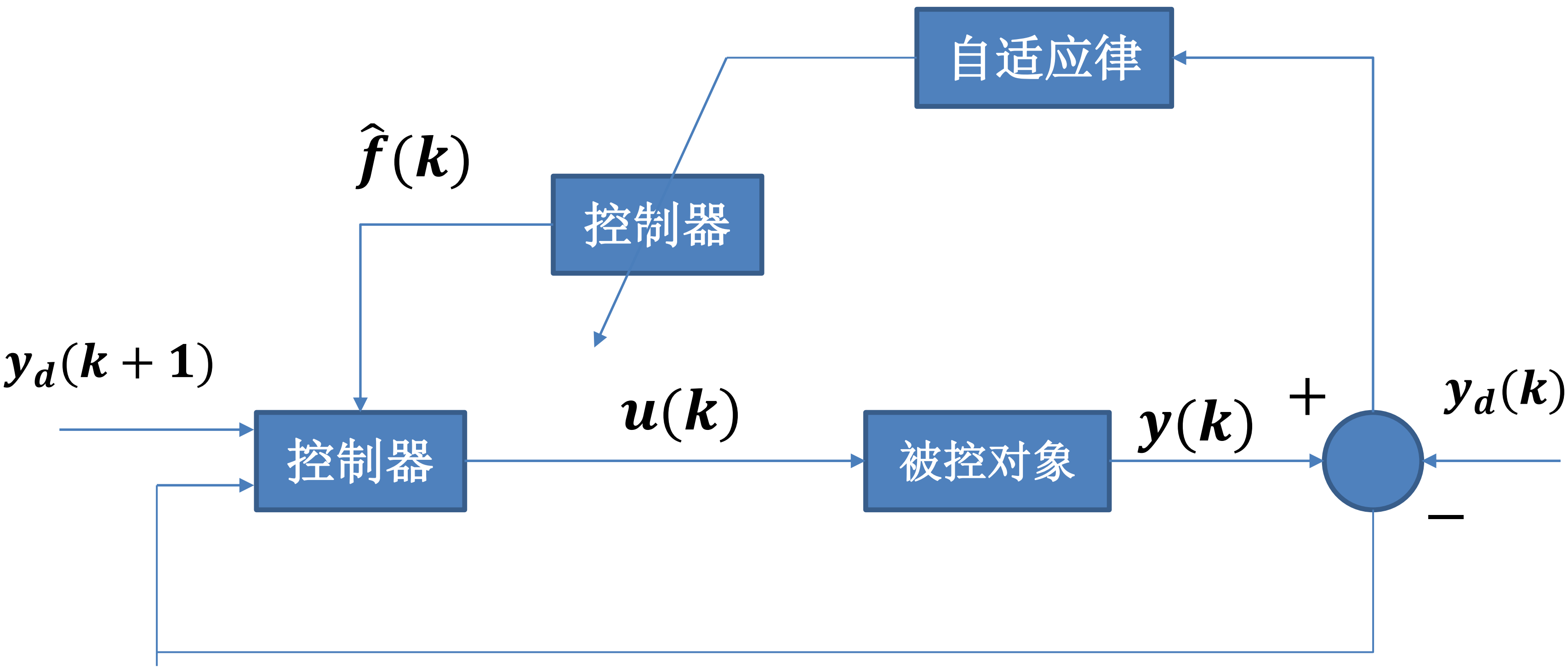
$$u(k)=y_d(k+1)-f\big(x(k)\big)-c_1e(k)$$

可得误差差分方程

$$e(k+1)+c_1e(k)=0$$

若$|c|<1, \mathrm{k}\to\infty$时，$e(k)\to 0$

离散系统的RBF网络控制



基于神经网络逼近的自适应控制

离散系统的RBF网络控制

- 自适应神经网络控制器设计

思想：利用RBF神经网络逼近$f(x(k))$

$$\hat{f}\big(x(k)\big)=\widehat{w}^T(k)h\big(x(k)\big)$$

根据逼近定理，存在最优权值向量w^*

$$f(x)=\hat{f}(x,w^*)-\Delta_f(x)$$

$\Delta_f(x)$为最优神经网络逼近误差，$\left|\Delta_f(x)\right|<\epsilon_f$

离散系统的RBF网络控制

神经网络逼近误差可写为

$$\tilde{f}\big(x(k)\big)=f\big(x(k)\big)-\hat{f}\big(x(k)\big)$$
$$=\hat{f}(x,w^*)-\Delta_f\big(x(k)\big)-\widehat{w}^T(k)h\big(x(k)\big)$$
$$=-\widetilde{w}^T(k)h\big(x(k)\big)-\Delta_f\big(x(k)\big)$$
$$\widetilde{w}(k)=\widehat{w}(k)-w^*$$

设计控制律

$$u(k)=y_d(k+1)-\hat{f}\big(x(k)\big)-c_1e(k)$$

离散系统的RBF网络控制

误差方程$e(k)+c_1e(k-1)=\tilde{f}\left(x(k-1)\right)$

$$e(k)=\Gamma^{-1}(z^{-1})\tilde{f}\left(x(k-1)\right)$$

$$\Gamma(z^{-1})=1+c_1z^{-1}$$

定义增广误差信号 (augmented error signal)

$$e_1(k)=\beta\left(e(k)-\Gamma^{-1}(z^{-1})v(k)\right),\beta>0$$

$$e_1(k)=\beta\frac{1}{1+c_1z^{-1}}\tilde{f}(x(k-1)-v(k))$$

$$e_1(k-1)=\frac{\beta\tilde{f}(x(k-1)-v(k)-e_1(k))}{c_1}$$

$v(k)$ 辅助信号 (auxiliary signal)

离散系统的RBF网络控制

定义离散时间Lyapunov函数

$$V(k)=e_1^2(k)+\gamma\tilde{w}^T(k)\tilde{w}(k)$$

$$\Delta V(k)=V(k)-V(k-1)$$

$$=-V_1$$

$$+2\hat{w}^T(k-1)\left(\gamma\Delta\hat{w}(k)-\frac{\beta}{c_1^2}h\big(x(k-1)\big)e_1(k)\right)$$

$$-\frac{2\beta}{c_1^2}\Big(\Delta_f\big(x(k-1)\big)+v(k)\Big)e_1(k)+\gamma\Delta\hat{w}^T(k)\Delta\hat{w}(k)$$

$$V_1=\frac{e_1^2(k)(1-c_1^2)}{c_1^2}+\frac{\beta^2\big(\tilde{f}\big(x(k-1)\big)-v(k)\big)^2}{c_1^2}\geq 0$$

离散系统的RBF网络控制

设计自适应律

$$\Delta\hat{w}(k)=\begin{cases}\dfrac{\beta}{\gamma c_1^2}h\big(x(k-1)\big)e_1(k) & |e_1(k)|>\dfrac{\varepsilon_f}{G}\\ 0 & |e_1(k)|\le\dfrac{\varepsilon_f}{G}\end{cases}$$

$$\Delta\hat{w}(k)=\hat{w}(k)-\hat{w}(k-1),\gamma>0,G>0$$

离散系统的RBF网络控制

$$\Delta V(k)=\begin{cases}-V_1-\frac{2\beta}{c_1^2}\left(\Delta_f\left(x(k-1)\right)+v(k)\right)e_1(k)+ \\ \left(\frac{\beta}{\sqrt{\gamma}c_1^2}\right)^2 h^T\left(x(k-1)\right)h\left(x(k-1)\right)e_1^2(k) & |e_1(k)|>\frac{\varepsilon_f}{G} \\ -V_1-\frac{2\beta}{c_1^2}\left[\left(\tilde{w}^T(k-1)h\left(x(k-1)\right)\right)+\right. \\ \left. v(k)+\Delta_f\left(x(k-1)\right)e_1(k)\right] & |e_1(k)|\le\frac{\varepsilon_f}{G}\end{cases}$$

设计辅助信号$v(k)$保证$e_1(k)\to 0$,从而$e(k)\to 0$

$$v(k)=v_1(k)+v_2(k)$$

$$v_1(k)=\frac{\beta}{2\gamma c_1^2}h^T\left(x(k-1)\right)h\left(x(k-1)\right)e_1(k) \quad v_2(k)=Ge_1(k)$$

离散系统的RBF网络控制

- 若$|e_1(k)| > \frac{\varepsilon_f}{G}$，则

$$\Delta V(k) \le -\frac{2\beta}{c_1^2}\Big(\Delta_f\big(x(k-1)\big) + Ge_1(k)\Big)e_1(k)$$

$$\left|\Delta_f(x)\right| < \epsilon_f, |e_1(k)| > \frac{\varepsilon_f}{G} \Rightarrow |e_1(k)| > \frac{\left|\Delta_f\big(x(k-1)\big)\right|}{G}, e_1^2(k)$$

$$> \frac{-\Delta_f\big(x(k-1)\big)e_1(k)}{G}$$

因此

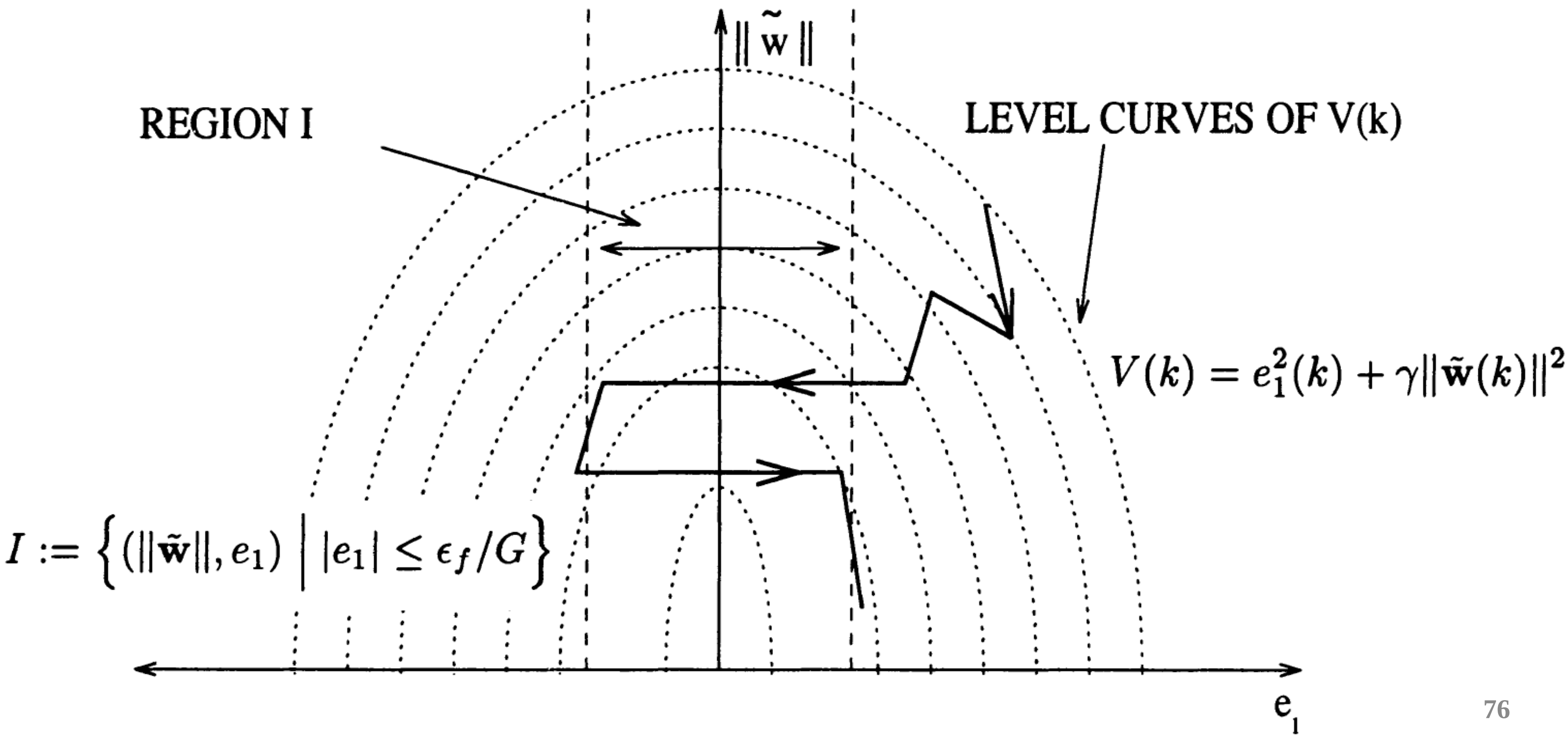
$$\Big(\Delta_f\big(x(k-1)\big) + Ge_1(k)\Big)e_1(k) > 0$$

故

$$\Delta V(k) < 0$$

离散系统的RBF网络控制

若$|e_1(k)| \le \dfrac{\varepsilon_f}{G}$，则可以保证跟踪性能，且$\Delta V(k)$可以任意小



离散系统的RBF网络控制

$$\lim_{k\to\infty}|e_1(k)| \le \left[1+\frac{1-|c_1|}{\beta G}+\frac{\beta\bar{h}}{2\gamma c_1^2 G}\right]\frac{\epsilon_f}{1-|c_1|}$$

$$\cong \frac{\epsilon_f}{1-|c_1|} \qquad\qquad if\ G \gg \frac{1-|c_1|}{\beta}+\frac{\beta\bar{h}}{2\gamma c_1^2}$$

$$\bar{h}=h^T\big(x(k-1)\big)h(x(k-1))$$

Simon G. Fabri, Visakan Kadirkamanathan. Functional Adaptive Control An Intelligent Systems Approach Springer Verlag London (2001), Chapter 5.

仿真实例

离散时间系统

$$y(k)=f(x(k-1))+u(k-1)$$

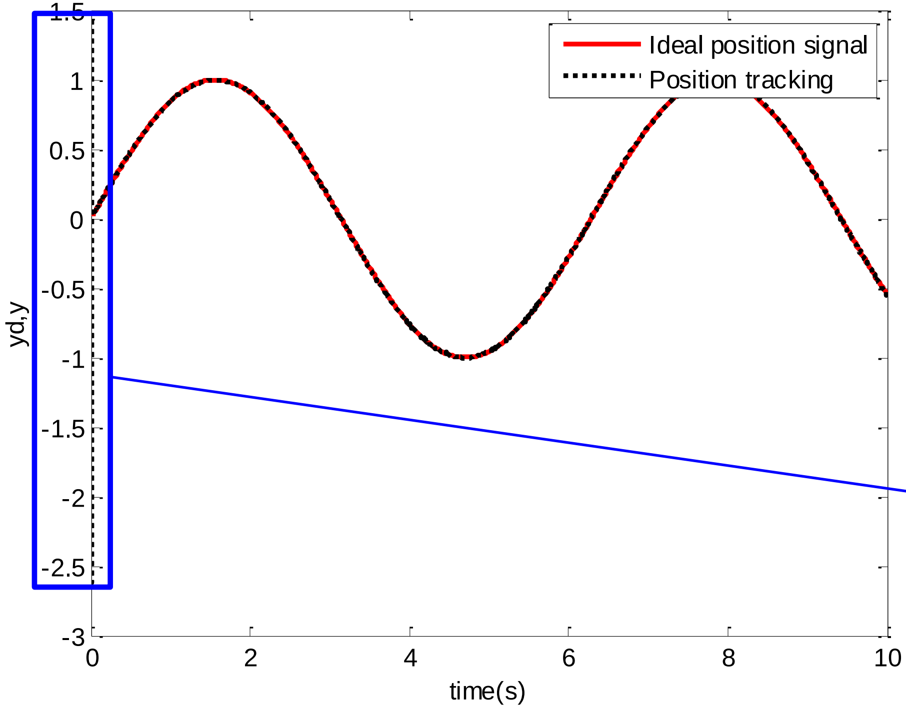
$$f\big(x(k-1)\big)=\frac{0.5y(k-1)(1-y(k-1))}{1+\exp(-0.25y(k-1))}$$

假定$f\big(x(k-1)\big)$未知，用RBF对其进行逼近，结构为1−9−1，网络输入取$y(k-1), c_i=[-2,-1.5\ -1.0\ -0.5\ 0\ 0.5\ 1.0\ 1.5\ 2], b_i=15,(i=1,j=1,2,\cdots,9)$，初始取值为$(0,1)$之间的随机数

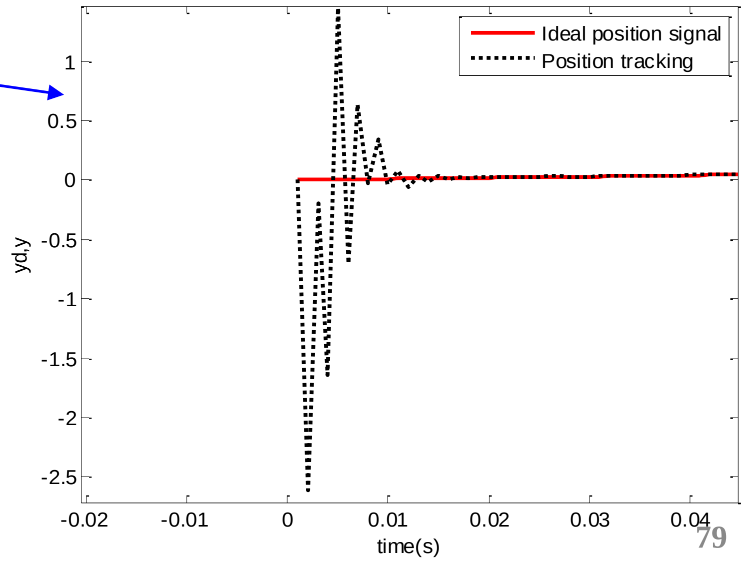
$$y_d=\sin t$$

$$c_1=-0.01,\beta=0.001,\gamma=0.001,G=50000,\epsilon_f=0.003$$

仿真实例

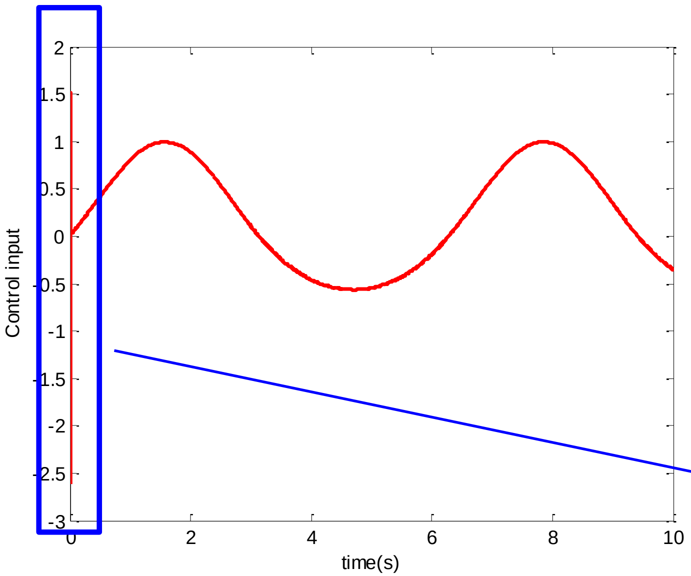


指令跟踪

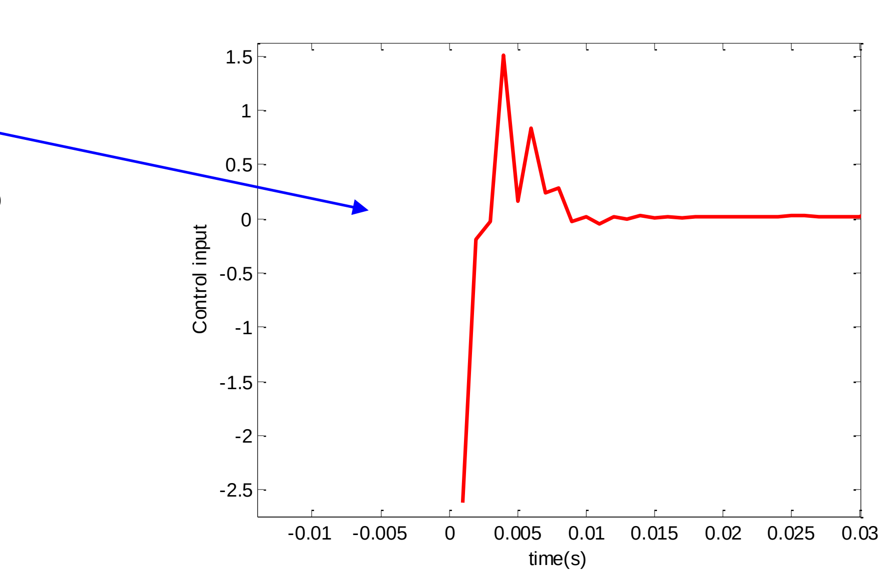


ts=0.001

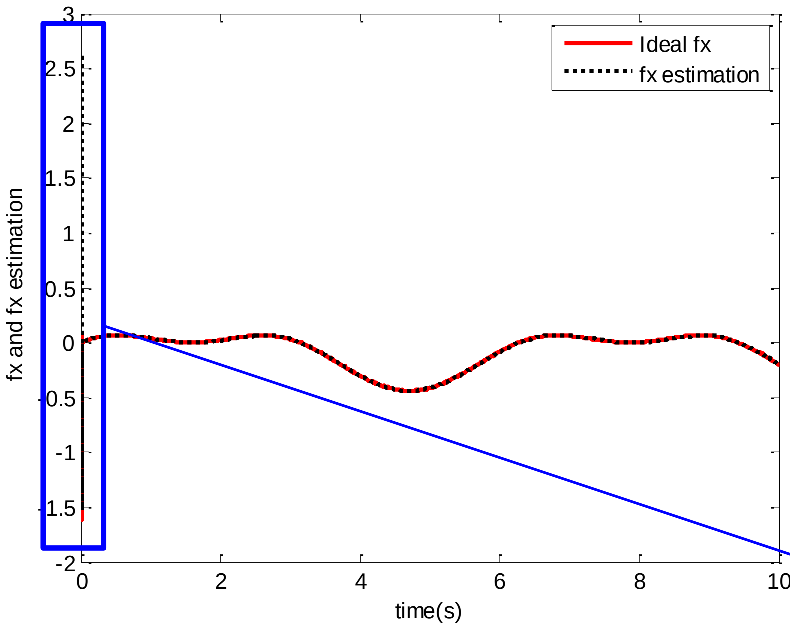
仿真实例


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
2
4
6
8
10
time(s)
Control input

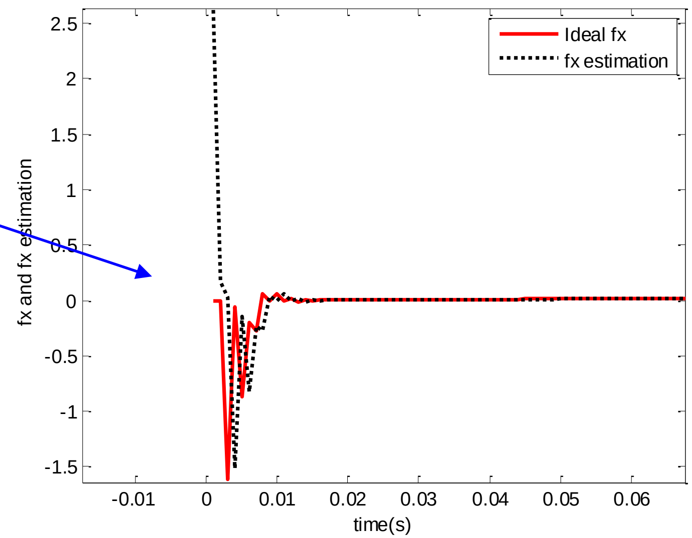
控制量


1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-0.01
-0.005
0
0.005
0.01
0.015
0.02
0.025
0.03
time(s)
Control input

仿真实例



$f(x(k-1))$及其估计



本讲的主要内容

一、人工神经元控制系统

二、RBF神经网络控制

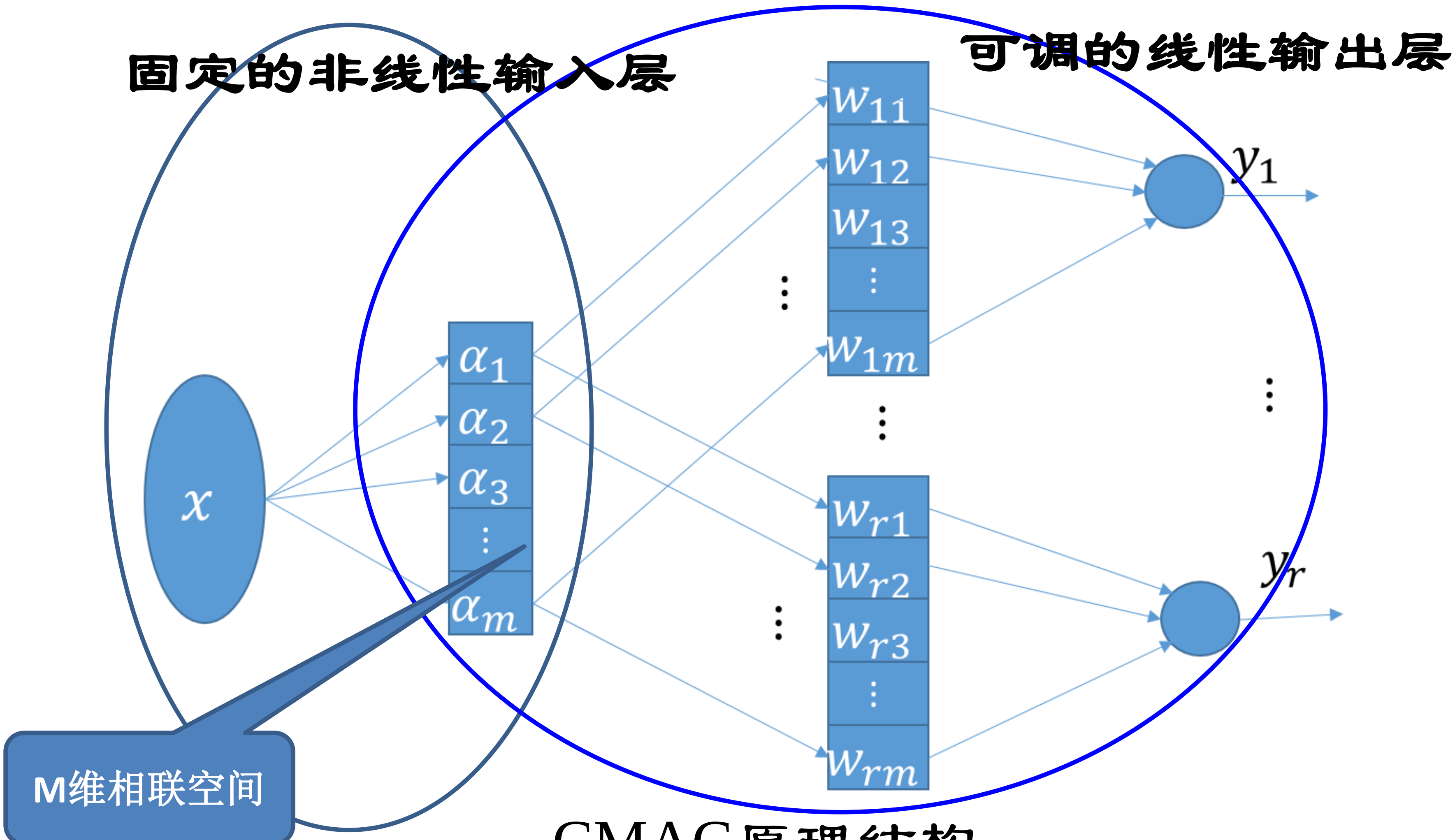
三、CMAC神经网络控制

四、Hopfield网络优化

CMAC神经网络

- 小脑模型关节控制器（Cerebellar Model Arculation Controller） 1975 Albus
- 仿照小脑如何控制肢体运动的原理而建立的神经网络模型，主要用来求解机械手的关节运动
- 推广至机器人控制、模式识别、自适应控制、信号处理等
- 局部逼近神经网络
- 一种联想网络，具有局部泛化能力，相似输入产生相似输出，远离的输入产生几乎独立地输出

CMAC神经网络



CMAC原理结构

CMAC神经网络数学表示

实现非线性映射

$$\boldsymbol{y}=f(\boldsymbol{x})\quad \boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}},\boldsymbol{y}=[y_1,x_2,\cdots,y_r]^{\mathrm{T}}$$

- （1） $S:\boldsymbol{x}\rightarrow A,\boldsymbol{\alpha}=S(\boldsymbol{x})$

$$\alpha=[\alpha_1,\alpha_2,\cdots,\alpha_m]^T\in A,\alpha_i=0\text{ or }1$$

A为m维相联空间； 固定数目c的$\alpha_i=1$, c:感受野（reception field）的大小，泛化参数

- （2） $P:A\rightarrow \boldsymbol{y},\boldsymbol{y}=P(\boldsymbol{\alpha})=W\boldsymbol{\alpha}$

$$W=\begin{bmatrix} w_{11} & \cdots & w_{1m}\\ \vdots & \ddots & \vdots\\ w_{r1} & \cdots & w_{rm}\end{bmatrix},\alpha=\begin{bmatrix}\alpha_1\\ \vdots\\ \alpha_m\end{bmatrix}$$

$$y_i=P_i(\alpha)=\sum_{j=1}^{m} w_{ij}\alpha_i$$

CMAC神经网络学习算法

连接权学习算法

标量形式 $$w_{ij}(k+1)=\frac{w_{ij}(k)+\beta(y_{di}-y_i)\alpha_j}{\boldsymbol{\alpha}^T\boldsymbol{\alpha}}$$

矢量形式 $$\boldsymbol{w_i}(k+1)=\frac{\boldsymbol{w_i}(k)+\beta(y_{di}-y_i)\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T\boldsymbol{\alpha}}$$

$$0<\beta<2$$

CMAC神经网络学习算法

定义

$$e_i(k) = y_{di}(k) - y_i(k) = y_{di} - w_i(k)\alpha(x)$$

$$\begin{aligned}\Delta e_i(k) &= e_i(k+1) - e_i(k) \\ &= [y_{di} - y_i(k+1)] - [y_{di} - y_i(k)] \\ &= -[y_i(k+1) - y_i(k)] \\ &= -[w_i(k+1) - w_i(k)]\alpha(x)\end{aligned}$$

$$\Delta w_i(k) = \frac{\beta e_i(k)\alpha(x)}{\alpha^T(x)\alpha(x)}$$

$$\Delta e_i(k) = -\beta e_i(k)\alpha^T(x)\alpha(x)/\alpha^T(x)\alpha(x) = -\beta e_i(k)$$

$$e_i(k+1) = (1-\beta)e_i(k)$$

CMAC神经网络

$$\lim_{k\to\infty} e_i(k) = 0$$

$$|1-\beta| < 1 \Rightarrow 0 < \beta < 2$$

CMAC网络具有如下特性：

- 可以实现从输入到输出的任意映射，输入各向量分量的量化精度愈高，分块愈细，逼近任意函数的精度就愈高
- 具有局部扩展功能，即在局部空间中靠近的向量，对应的输出也是靠近的
- 采用自适应LMS自适应算法，可得到全局最小值
- 由于相联空间中只有少数几个元素为1，其余均为0，因此在一次训练中只有少数的连接权需要调整，计算量比BP网络要小

CMAC输入层非线性映射实现

输入向量$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}\longrightarrow$相联空间A中向量$\alpha$

$$S:\boldsymbol{x}\rightarrow A$$

输入较近⇒输出较近　　　　输入较远⇒输出较远

输入空间距离（海明距离，Hamming Distance）

$$H_{ij}=\sum_{k=1}^{n}\left|x_{ik}-x_{jk}\right|$$

x_i与x_j距离较近　　交集$A_i^*\wedge A_j^*$应较大

x_i与x_j距离较远　　交集$A_i^*\wedge A_j^*$应较小

CMAC输入层非线性映射实现

A^*表示A中非零元素的集合

要求：相联空间的元素个数$|A_p|$远远大于A^*中元素的个数$|A^*|$，通常选$|A_p| = 100|A^*|$

保证输入空间的每一点都存在唯一的映射$x \rightarrow A$

假定向量x的每个分量可以取q个不同的值，则输入共有q^n个不同模式。

假定$u = |A^*|, v = {|A_p|}/{|A^*|}$，则$A^*$的组合数为

$$C_{uv}^{u} = \frac{(uv)!}{u!(uv-v)!} > \frac{(uv-u)^u}{u!} > \frac{(uv-u)^u}{u^u} = (v-1)^u$$

$$v = 100 \Rightarrow q^n < 99^{|A^*|}$$

CMAC输入层非线性映射实现

非线性映射分解成两步

$$x \longrightarrow M, M \longrightarrow A$$

1) $x \longrightarrow M$

- 将输入向量x的每个分量x_i转成二进制变量m_i,
- m_i只有一个区间段的位值均为1，其余为0
- 在任何一个m_i中位值为1的个数$|m^*|$均等于$|A^*|$

CMAC输入层非线性映射实现

$$x \longrightarrow m, c = 4$$

	m											
x	μ_a	μ_b	μ_c	μ_d	μ_e	μ_f	μ_g	μ_h	μ_i	μ_j	μ_k	μ_l
1	1	1	1	1	0	0	0	0	0	0	0	0
2	0	1	1	1	1	0	0	0	0	0	0	0
3	0	0	1	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	1	0	0	0	0	0
5	0	0	0	0	1	1	1	1	0	0	0	0
6	0	0	0	0	0	1	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1	1	0	0
8	0	0	0	0	0	0	0	1	1	1	1	0
9	0	0	0	0	0	0	0	0	1	1	1	1

CMAC输入层非线性映射实现

$x \longrightarrow m$简化形式

x	m^*
1	a, b, c, d
2	e, b, c, d
3	e, f, c, d
4	e, f, g, d
5	e, f, g, h
6	i, f, g, h
7	i, j, g, h
8	i, j, k, h
9	i, j, k, l

$\boldsymbol{H_{ij}} < |\boldsymbol{A_i^*}|$时,

$$\boldsymbol{H_{ij}} = |\boldsymbol{A_i^*}| - \left|A_i^* \wedge A_j^*\right|$$

$\boldsymbol{x_1 = 1, x_2 = 3}$时,

$$A_1^* \wedge A_2^* = \{\boldsymbol{c}, \boldsymbol{d}\}, |\boldsymbol{A_i^*}| = \boldsymbol{4}$$

$$\boldsymbol{H_{12}} = |\boldsymbol{A_1^*}| - |A_1^* \wedge A_2^*| = \boldsymbol{2}$$

CMAC输入层非线性映射实现

$$\boldsymbol{x}=[x_1,x_2]^T,1\le x_1\le 5,1\le x_2\le 7,|A_i^*|=4$$

$\boldsymbol{x}\longrightarrow \boldsymbol{m}$简化形式

x_1	m_1^*
1	A,B,C,D
2	E,B,C,D
3	E,F,C,D
4	E,F,G,D
5	E,F,G,H

$\boldsymbol{x}\longrightarrow \boldsymbol{m}$简化形式

x_2	m_2^*
1	a,b,c,d
2	e,b,c,d
3	e,f,c,d
4	e,f,g,d
5	e,f,g,h
6	i,f,g,h
7	i,j,g,h

CMAC输入层非线性映射实现

2) $M \longrightarrow A, c = 4$

A^* \ x_1 / x_2		**A,B,C,D**	**E,B,C,D**	**E,F,C,D**	**E,F,G,D**	**E,F,G,H**
		1	2	3	4	5
a,b,c,d	1	Aa,Bb,Cc,Dd	Ea,Bb,Cc,Dd	Ea,Fb,Cc,Dd	Ea,Fb,Gc,Dd	Ea,Fb,Gc,Hd
e,b,c,d	2	Ae,Bb,Cc,Dd	Ee,Bb,Cc,Dd	Ee,Fb,Cc,Dd	Ee,Fb,Gc,Dd	Ee,Fb,Gc,Hd
e,f,c,d	3	Ae,Bf,Cc,Dd	Ee,Bf,Cc,Dd	Ee,Ff,Cc,Dd	Ee,Ff,Gc,Dd	Ee,Ff,Gc,Hd
e,f,g,d	4	Ae,Bf,Cg,Dd	Ee,Bf,Cg,Dd	Ee,Ff,Cg,Dd	Ee,Ff,Gg,Dd	Ee,Ff,Gg,Hd
e,f,g,h	5	Ae,Bf,Cg,Dh	Ee,Bf,Cg,Dh	Ee,Ff,Cg,Dh	Ee,Ff,Gg,Dh	Ee,Ff,Gg,Hh
i,f,g,h	6	Ai,Bf,Cg,Dh	Ei,Bf,Cg,Dh	Ei,Ff,Cg,Dh	Ei,Ff,Gg,Dh	Ei,Ff,Gg,Hh
i,j,g,h	7	Ai,Bj,Cg,Dh	Ei,Bj,Cg,Dh	Ei,Fj,Cg,Dh	Ei,Fj,Gg,Dh	Ei,Fj,Gg,Hh

CMAC输入层非线性映射实现

$$\boldsymbol{x}_1 = [3,5]^T, \boldsymbol{x}_2 = [3,2]^T$$
$$\boldsymbol{H}_{12} = 3, \boldsymbol{A}_1^* \wedge \boldsymbol{A}_1^* = \{\boldsymbol{Ee}\}, |\boldsymbol{A}^*| = 4$$

$$\boldsymbol{H}_{12} = |\boldsymbol{A}_1^*| - |A_1^* \wedge A_2^*| = 3$$

对所有的组合并不总成立，但$\boldsymbol{x}_1, \boldsymbol{x}_2$距离增加时，$|A_1^* \wedge A_2^*|$不会增加

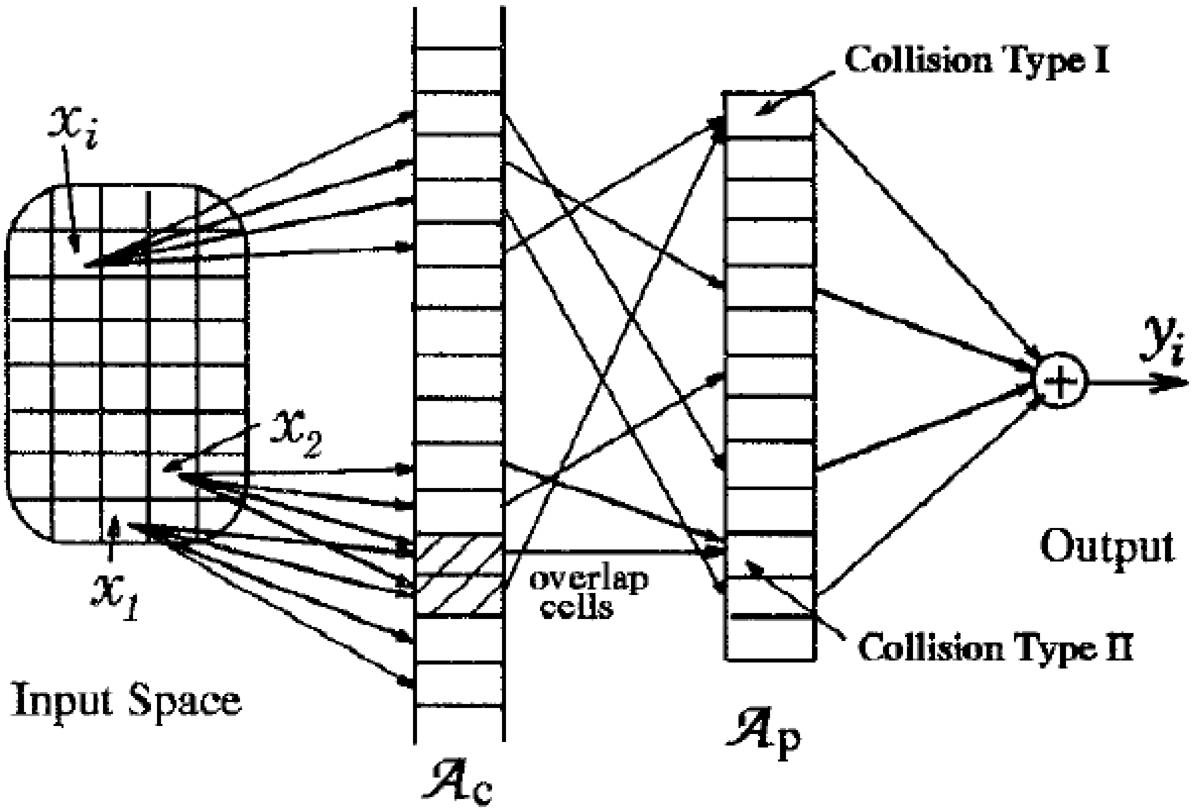
$S: \boldsymbol{x} \rightarrow A$ 使邻域内产生泛化，而不同邻域则产生分类

CMAC网络存储哈希编码

■ A到A_p的映射

- x的每个分量有q个值，则A的维数为q^n，如$q=50, n=10$,则需50^{10}个存储单元
- 哈希编码：计算机中数据压缩技术。当在一个大的存储区域稀疏地存储一些数据时，可以通过哈希编码将其压缩到一个很小的区域。

将稀疏地存于A中的A^*通过哈希编码将其压缩存储到小的存储区域A_p中。可以通过产生一个伪随机数的程序来实现，将大的存储区域A的地址作为该伪随机数产生程序的变量，产生的随机数限制在一个很小的范围内，该随机数便作为小存储区域A_p的地址。


x_i
x_2
x_1
Input Space
$\mathcal{A}_c$
overlap cells
Collision Type I
Collision Type II
$\mathcal{A}_p$
y_i
Output

CMAC网络存储哈希编码

- A到A_p的映射

①同一输入向量χ所对应的A^*的不同元素映射到同一地址的概率$\left(\left|A_p\right| = 2000, |A^*| = 20\right)$

$$\frac{1}{2000} + \frac{2}{2000} + \cdots + \frac{19}{2000} \approx 0.1$$

②相距较远向量x_1, x_2有$A_1^* \wedge A_2^* = \Phi$,但经过映射后

$$A_{1p}^* \wedge A_{2p}^* \neq \Phi$$

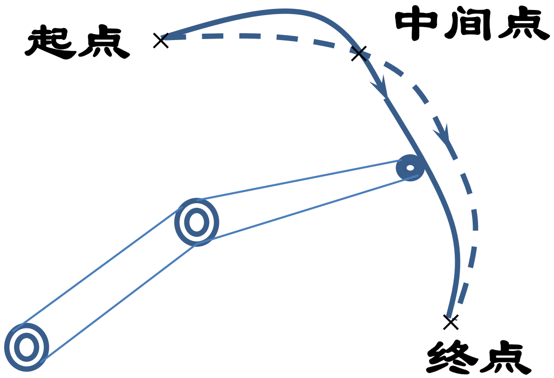
$$\mathrm{prob}(|A_1^* \wedge A_2^*| = 0) = 0.818$$

$$\mathrm{prob}(|A_1^* \wedge A_2^*| = 1) = 0.165$$

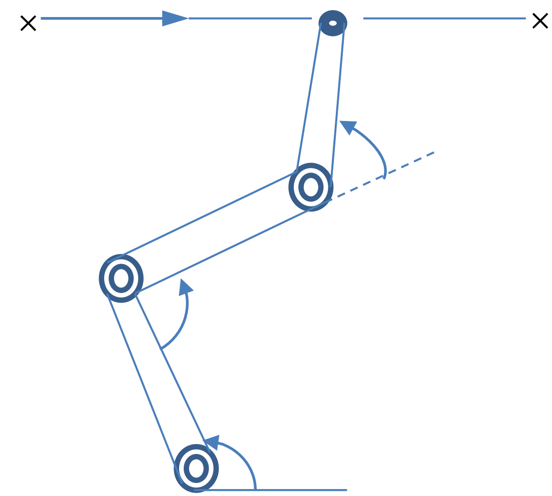
$$\mathrm{prob}(|A_1^* \wedge A_2^*| = 2) = 0.016$$

$$\mathrm{prob}(|A_1^* \wedge A_2^*| \geq 3) = 0.001$$

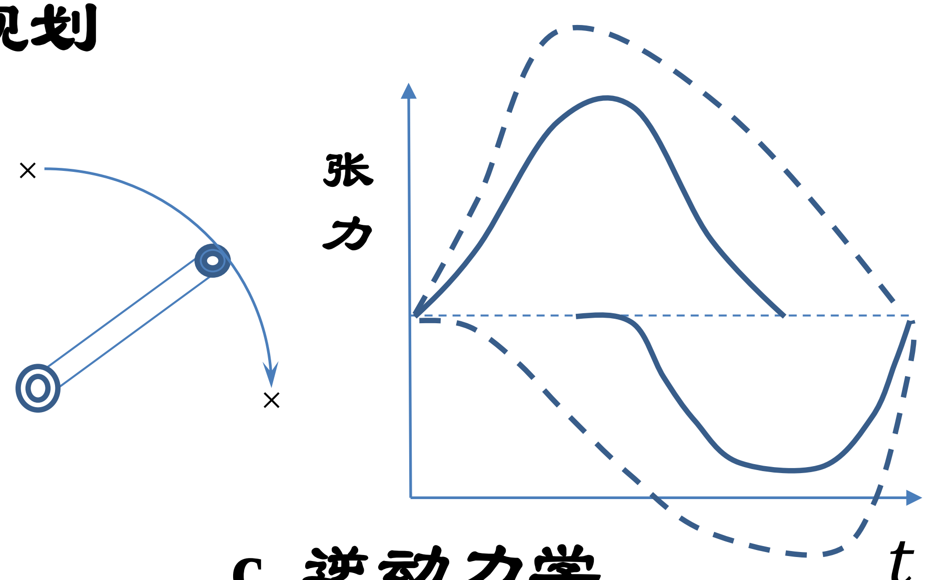
机械手控制问题



a. 轨迹规划



b. 逆运动学



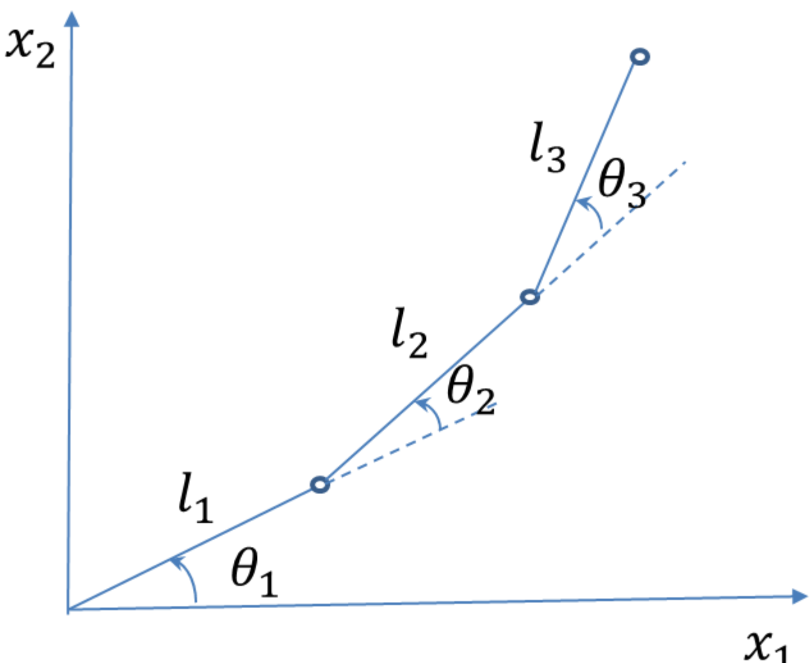
c. 逆动力学

机械手逆运动学

机械手运动方程（kinematic equation）静态方程

机械手臂长$l_1, l_2, l_3,$ **以及各关节角**$\theta_1,\ \theta_2, \theta_3,$ **则机械手终端在直角坐标系中的位置**

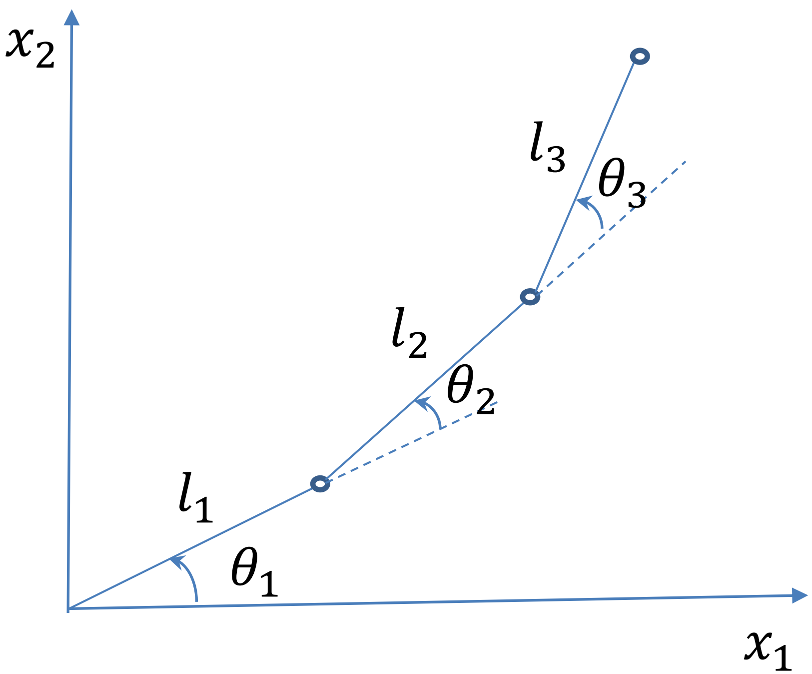
$$x_1 = l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3)$$
$$x_2 = l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) + l_3\sin(\theta_1+\theta_2+\theta_3)$$



写成一般形式

$$X = F(\Theta)$$
$$X = (x_1, x_2)^T, \Theta = (\theta_1,\ \theta_2, \theta_3)^T$$

机械手逆运动学



平面上运动的三关节机械手

■ 基于位置的逆运动学控制

- 已知空间轨迹$X(t)$，求关节运动轨迹$\Theta(t)$
- 求逆有

$$\Theta = F^{-1}(X)$$

■ 基于速度的逆运动学控制

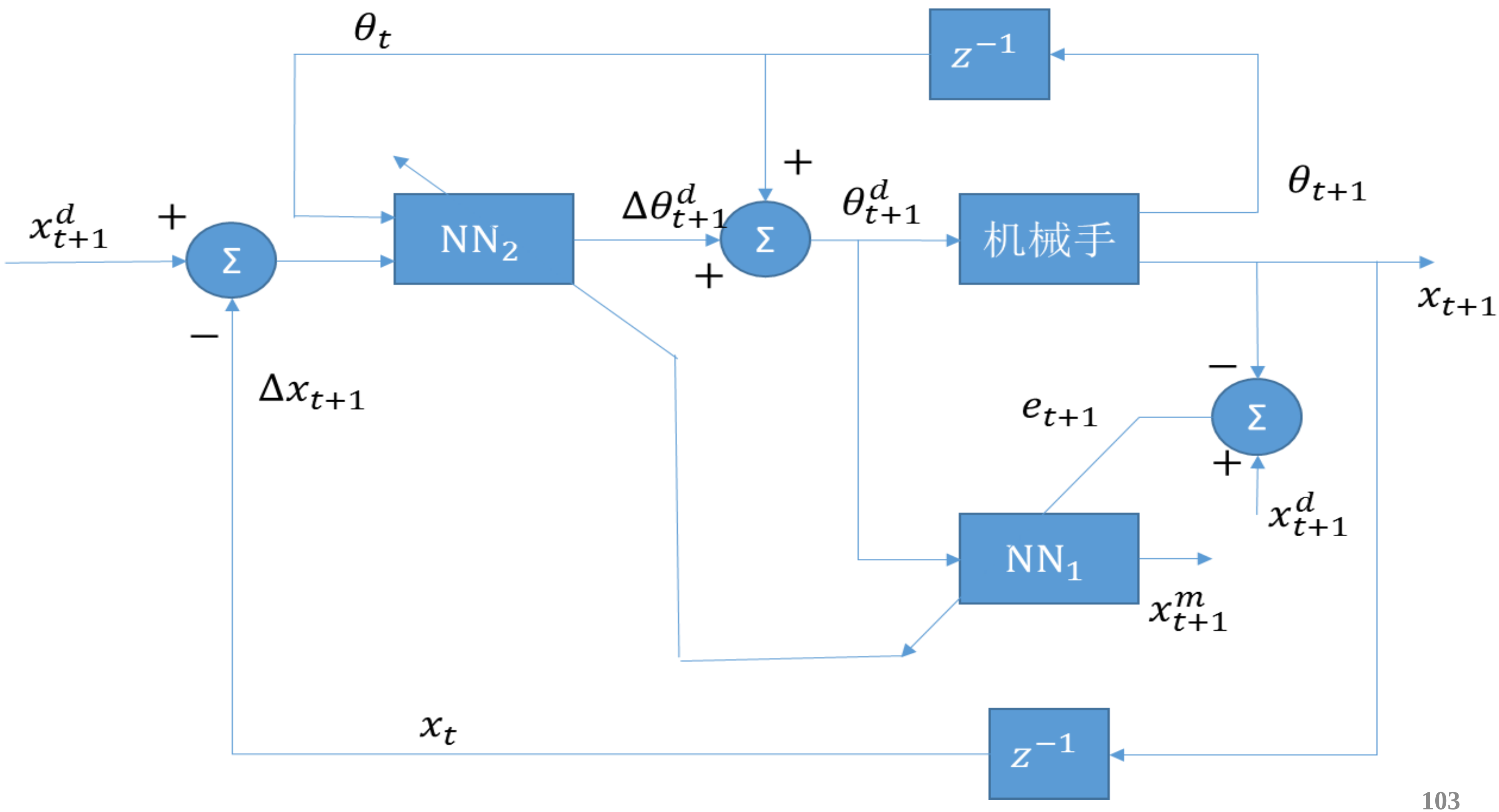
- 有时需要用到速度信息

$$\dot{X} = J(\Theta)\dot{\Theta}$$

$$\dot{\Theta} = J^{-1}(\Theta)\dot{X}$$

$J(\Theta)$机械手的Jacobian矩阵

机械手逆运动学神经网络控制



CMAC神经网络控制

- NN1是机械手正模型，NN2是机械手逆模型
- 在$t+1$时刻，由空间轨迹$X^d(t)$得到X_{t+1}^d作为给定信号，与t时刻真实位置作比较
- $\Delta X_{t+1} = X_{t+1}^d - X^d$
- $\Theta_{t+1}^d = \Theta_t + \Delta\Theta_{t+1}^d$
- $e_{t+1} = X_{t+1}^d - X_{t+1}$

机械手正模型NN1

- 机械手正模型

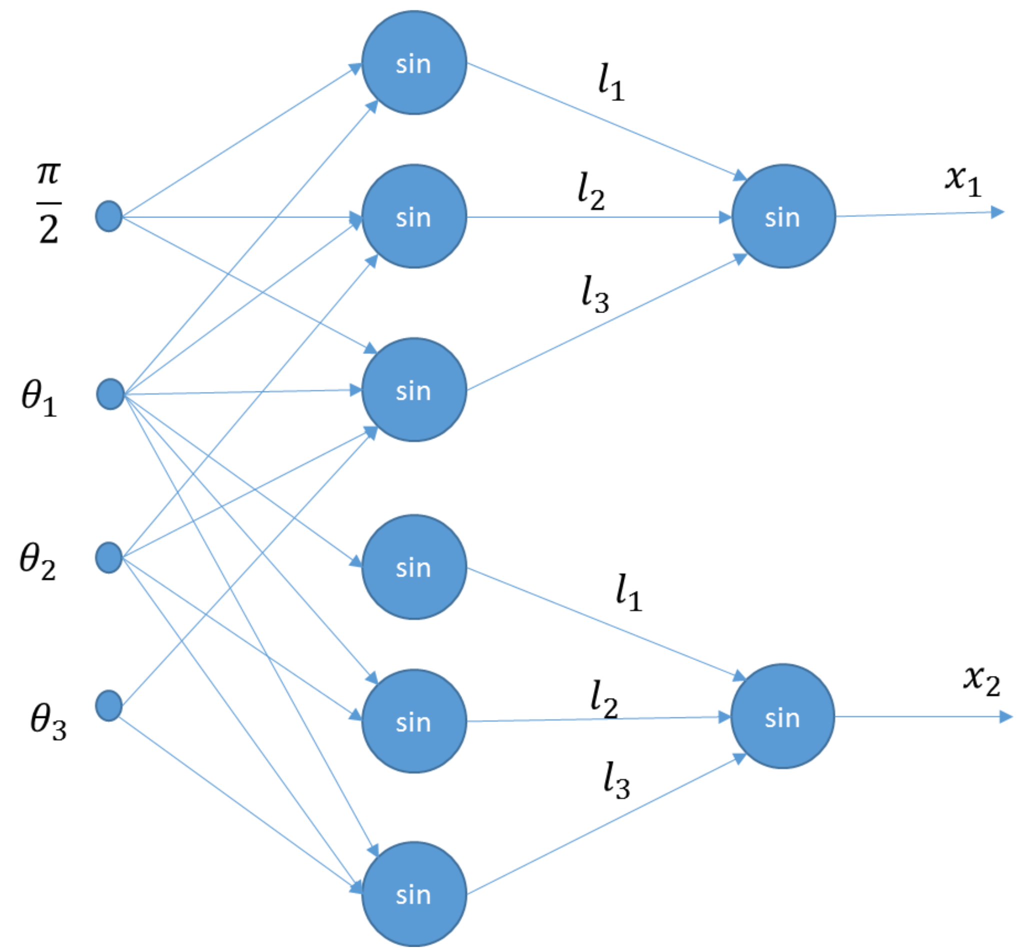
  实现由关节角Θ到空间位置X的变化

$$x_1 = l_1 \sin\left(\theta_1 + \frac{\pi}{2}\right) + l_2 \sin\left(\theta_1 + \theta_2 + \frac{\pi}{2}\right) + l_3 \sin\left(\theta_1 + \theta_2 + \theta_3 + \frac{\pi}{2}\right)$$

$$x_2 = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

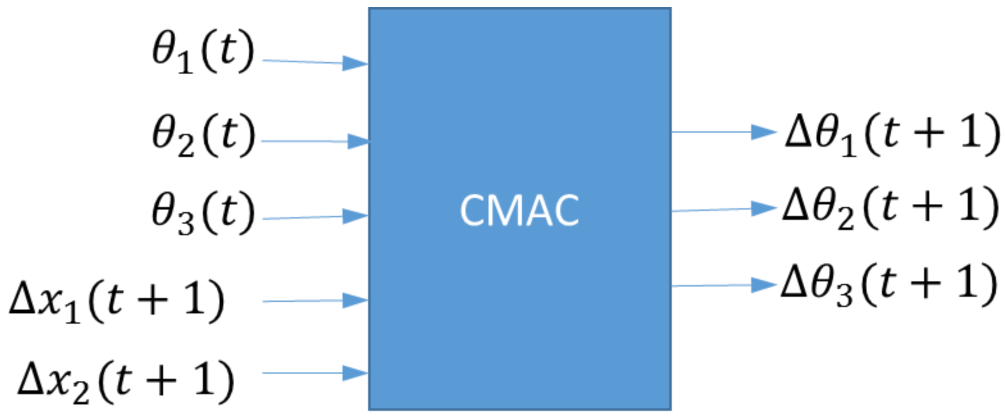
机械手正模型NN1

- **神经网络NN1结构**
- $J(\Theta) = \begin{bmatrix} \frac{\partial x_1}{\partial \theta_1} & \frac{\partial x_1}{\partial \theta_2} & \frac{\partial x_1}{\partial \theta_3} \\ \frac{\partial x_2}{\partial \theta_1} & \frac{\partial x_2}{\partial \theta_2} & \frac{\partial x_2}{\partial \theta_2} \end{bmatrix}$



机械手逆模型NN2

- 5个输入
- 每个输入量化级$R = 200$
- 感受野宽度$C = 40$
- 物理存储空间$Ap = 9240$
- 权值系数$w_{ij}, i = 1,2,3, j = 1,2,\cdots, C$



机械手逆模型NN2

CMAC网络输出

$$\Delta\theta_i(t+1)=\sum_{j=1}^{C} w_{ij}$$

- 求解$X\longrightarrow\theta$

指标函数（含关节角变化最小的约束条件）

$$\min U=\frac{\alpha}{2}\sum_{m=1}^{2} e_m^2(t+1)+\frac{\beta}{2}\sum_{i=1}^{3}\Delta\theta_i^2(t+1)$$

$$e_m(t+1)=x_m^d(t+1)-x_m(t+1)$$

$$U=\frac{\alpha}{2}e^T e+\frac{\beta}{2}\Delta\Theta^T\Delta\Theta$$

$e=[e_1(t+1),e_2(t+1)]^T \quad \Delta\Theta=[\Delta\theta_1(t+1),\Delta\theta_2(t+1),\Delta\theta_3(t+1)]^T$

机械手逆模型NN2

$$\frac{\partial U}{\partial \Delta\Theta} = \alpha e^T \frac{\partial e}{\partial \Delta\Theta} + \beta \Delta\Theta^T = \alpha e^T \left(\frac{-\partial X}{\partial \Delta\Theta}\right) + \beta \Delta\Theta^T$$
$$= -\alpha e^T J + \beta \Delta\Theta^T$$

$$\Delta\Theta^{k+1} = \Delta\Theta^k - \eta' \frac{\partial U}{\partial \Delta\Theta}$$

- CAMC学习，定义

$$V = \frac{1}{2} e^T e = \frac{1}{2} \sum_{m=1}^{2} e_m^2(t+1)$$

$$= \frac{1}{2} \sum_{m=1}^{2} [X_m^d(t+1) - X_m(t+1)]^2$$

机械手逆模型NN2

- w_{ij}修正算法

$$w_{ij\mathrm{new}} = w_{ij\mathrm{old}} - \eta \frac{\partial V}{\partial w_{ij}}, i = 1,2,3; j = 1,2,\cdots, C$$

$$\frac{\partial V}{\partial w_{ij}} = e^T \frac{\partial e}{\partial w_{ij}} = e^T \frac{\partial e}{\partial \Theta_{t+1}} \frac{\partial \Theta_{t+1}}{\partial w_{ij}} = e^T(-J_i)$$

仿真实验

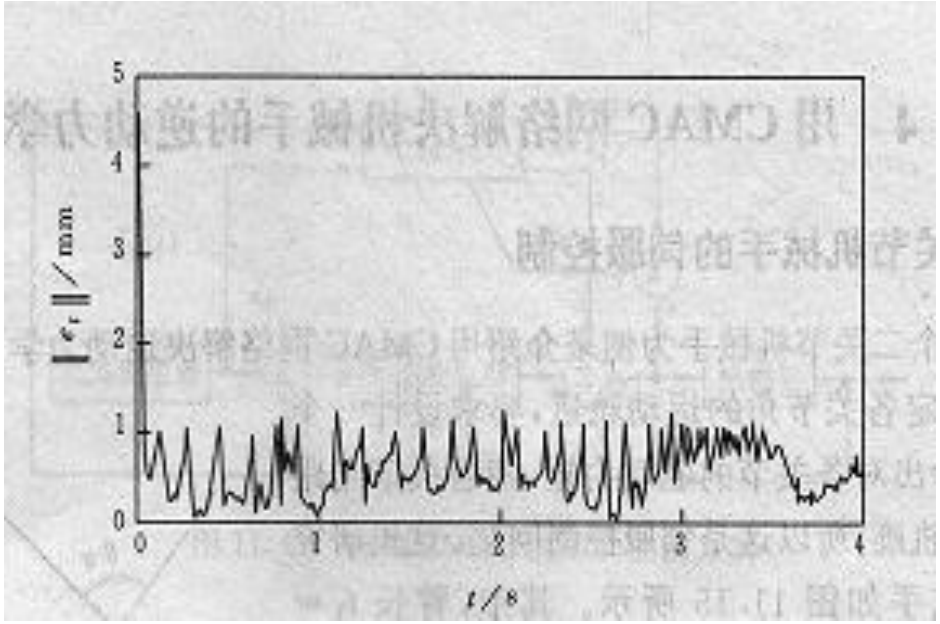
三关节机械手$l_1 = l_2 = l_3 = 0.5m$

跟踪平面上一个圆周，圆心$(0.5\mathrm{m}, 0.25\mathrm{m})$，半径$0.25m$

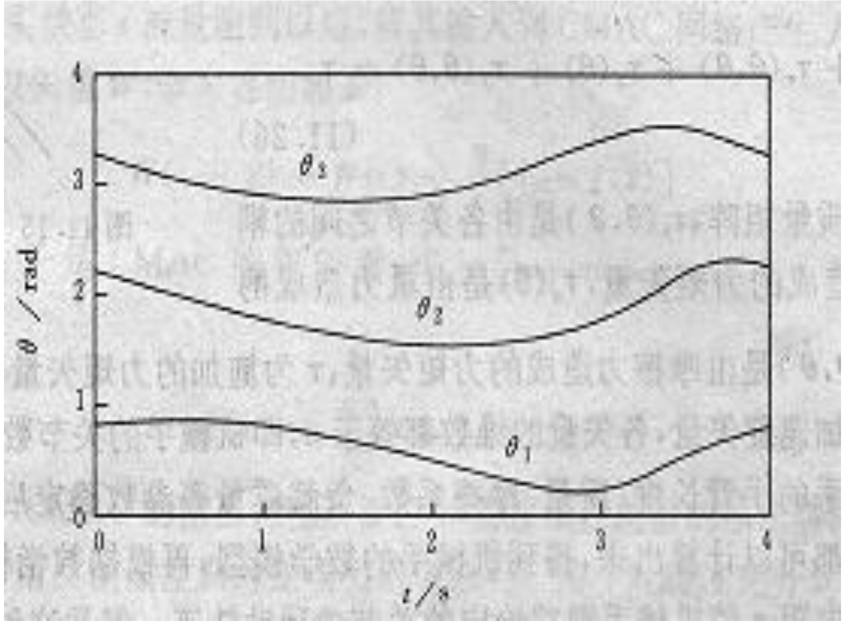
仿真实验

- ①设CMAC网络结构参数为：量化级$R=200$，感受野宽度$C=40$，物理存储空间$Mp=9240$，学习参数$\alpha=300, \beta=1, \eta=0.0001$，初始化CMAC的权值$w_{ij}=0, i=1,2,3; j=1,2,\cdots,C$. 设$t=0$
- ②由参考运动轨迹方程求出X_{t+1}^d,计算$\Delta X_{t+1}=X_{t+1}^d-X^d$, NN2产生$\Delta\Theta_{t+1}^d$，计算$\Theta_{t+1}^d=\Theta_t+\Delta\Theta_{t+1}^d$作为机械手关节的给定角
- ③根据机械手运动方程计算出t + 1时刻机械手空间位置X_{t+1}和关节角状态Θ_{t+1}，计算雅克比矩阵J
- ④求出位置跟踪误差$e_{t+1}=X_{t+1}^d-X_{t+1}$，修正NN2权值$w_{ij}$
- ⑤令$t=t+1$，转到步骤②

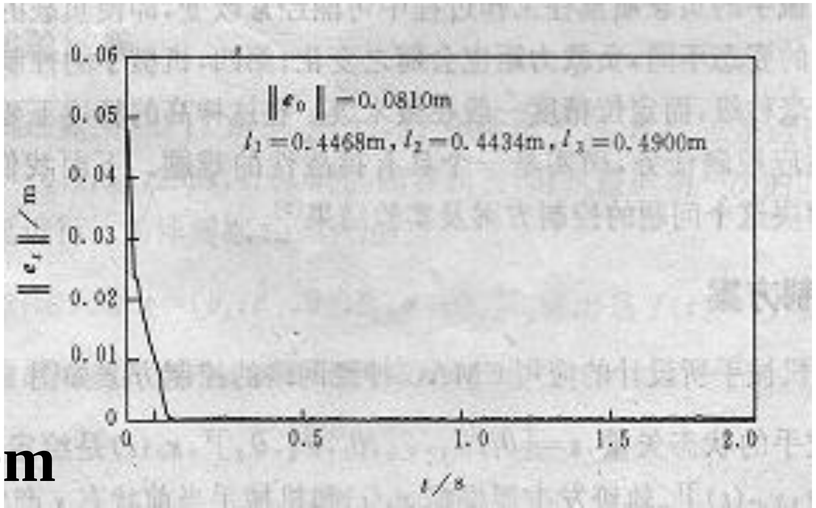
仿真实验



跟踪误差变化曲线



关节角运动轨迹



臂长变化后的跟踪误差变化曲线

l1=0.4468m，l2=0.4434m,l3=0.4900m

CMAC网络解机械手逆动力学问题

- n关节机械臂，动态方程

$$M(\theta)\ddot{\theta}+C(\theta,\dot{\theta})\dot{\theta}+G(\theta)+\tau_f(\theta,\dot{\theta})=\tau$$

$M(\theta)$：正定惯性矩阵

$C(\theta,\dot{\theta})$：离心力、哥氏力等矩项

$G(\theta)$：重力项

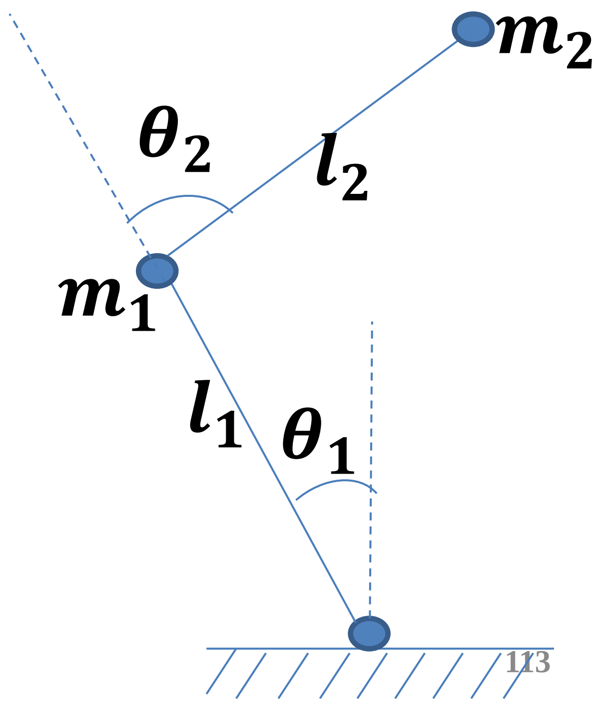
$\tau_f(\theta,\dot{\theta})$：摩擦力矩、干扰项

τ：施加的外力距，控制输入

θ：关节角度

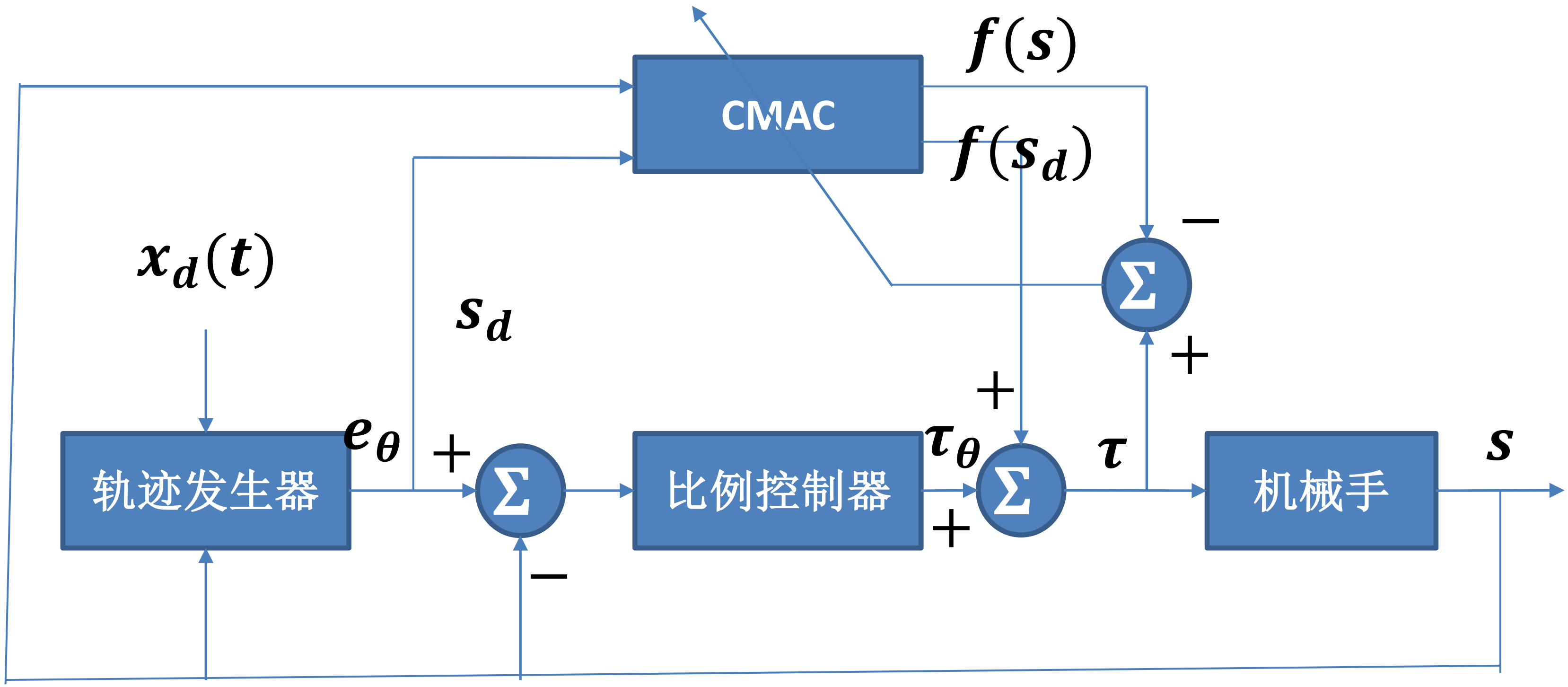
$\dot{\theta}$：关节角速度

$\ddot{\theta}$：关节角加速度



CMAC网络解机械手逆动力学问题

$$s = [\theta_1, \dot{\theta}_1, \ddot{\theta}_1, \theta_2, \dot{\theta}_2, \ddot{\theta}_2]$$



CMAC网络机械手逆动力学控制方案

$\tau = [\tau_1, \tau_2]^T$ $\quad \tau - f(s)$修正权值矢量W

CMAC网络解机械手逆动力学问题

$$W(t+1) = W(t) + \frac{\eta}{c}[\tau - f(s)]$$

$W = [w_1, w_2]^T, w_i = [w_{i1}, w_{i2}, \cdots, w_{ic}], c$为感受野宽度

$0 < \eta < 1$为学习率

$$f(s_i) = \sum_{j=1}^{c} w_{ij}$$

仿真实例

$$s = [\theta_1, \dot{\theta}_1, \ddot{\theta}_1, \theta_2, \dot{\theta}_2, \ddot{\theta}_2]^T, f(s) = [f(s_1), f(s_2)]^T$$

$$-4\text{rad} \le \theta \le 4\text{rad} \qquad -20\text{rad/s} \le \dot{\theta} \le 20\text{rad/s}$$

$$-100\text{rad}/s^2 \le \dot{\theta} \le 100\text{rad}/s^2$$

$$R = 400, \beta = 0.6, C = 80, Ap = 50000$$

定义均方根误差RMS

$$\text{RMS}= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\left(\theta_1^d(t) - \theta_1(t)\right)^2 + \left(\theta_2^d(t) - \theta_2(t)\right)^2\right]}$$

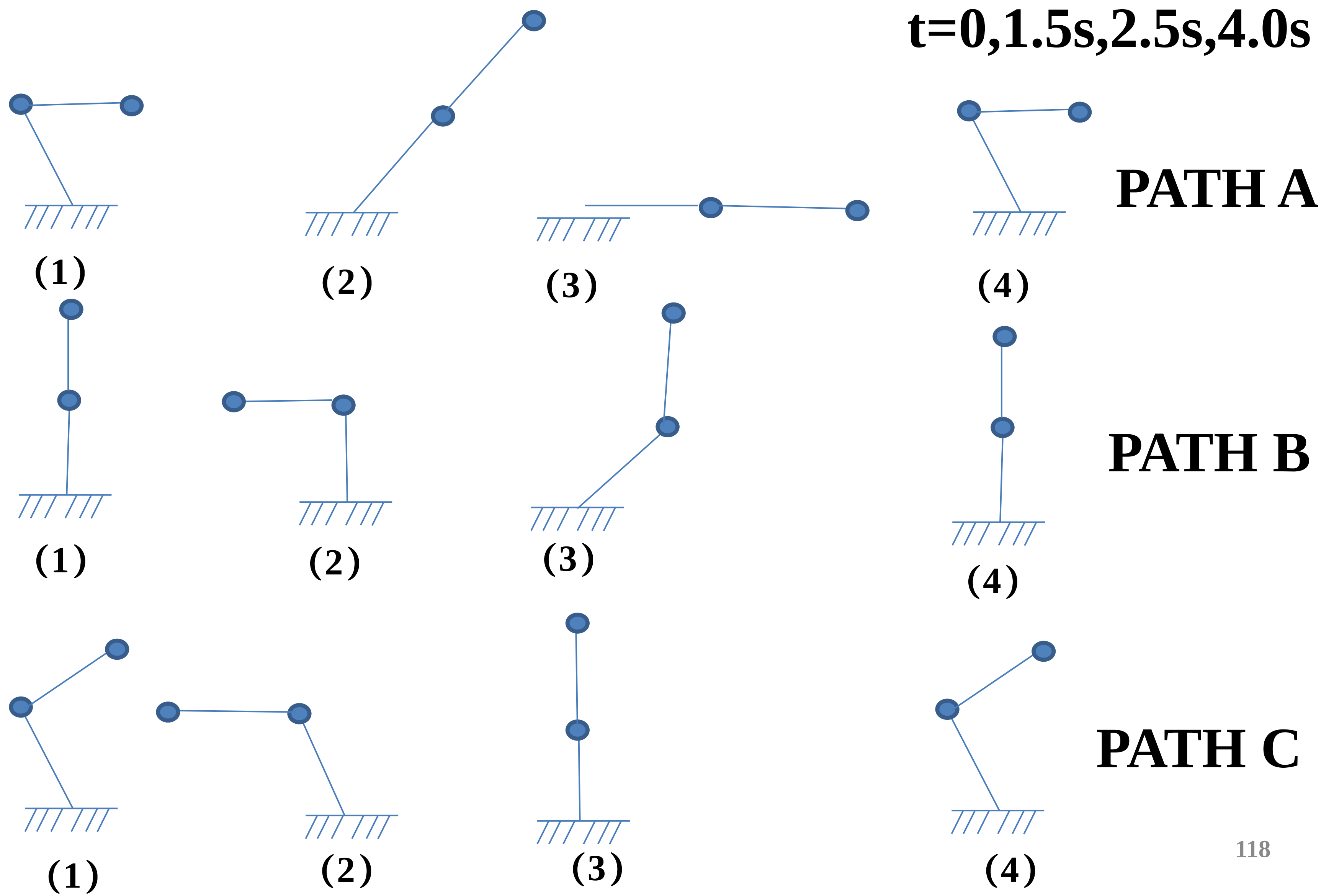
Miller W T, Glanz F H, Kraft L G. Application of a general learning algorithm to the Control of robotic manipulator, Int. J. Robotics Research, 1987,6(2):84~98

仿真实验

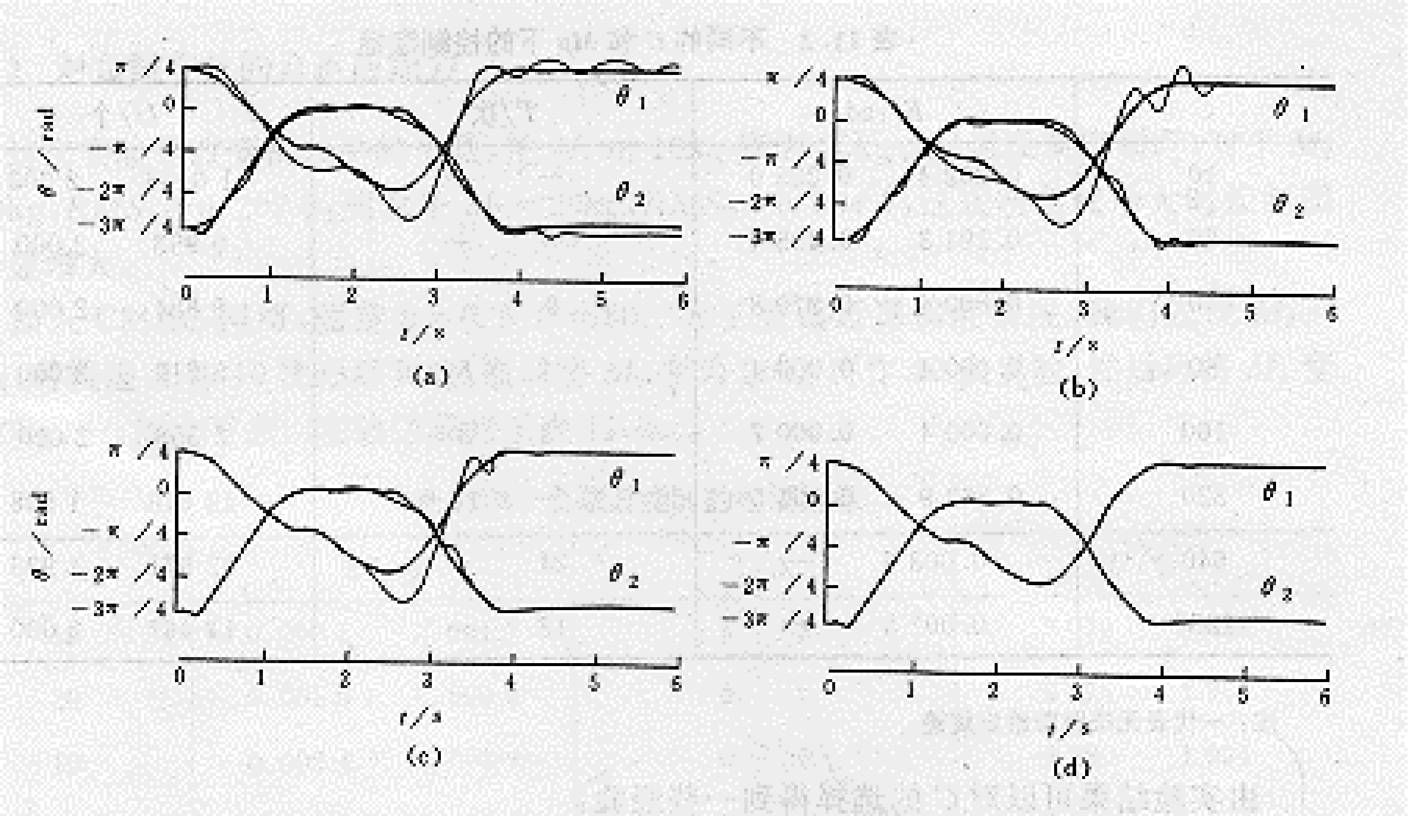
■ 性能指标

- 最终误差E：对每条轨迹作50次试验，最后五次试验的RMS作为E
- 试验次数T：令单独反馈控制器时的RMS为RMS_0，加上CMAC后重新做试验，当RMS小于10% RMS_0时的试验次数
- 内存使用数U：50次连续试验，Ap中实际被使用的内存单元

仿真实验



仿真实验



对轨迹A的学习跟踪曲线

(a)学习前跟踪结果;(b)第一次跟踪结果;
(c)第二次跟踪结果;(d)第三次跟踪结果

参数对控制性能影响

不同η和Ap下的控制指标

η	E/rad		T/次		U/个	
0.2	0.0006	0.0005	21	15	3544	1746
0.4	0.0003	0.0004	6	6	4203	1792
0.6	0.0004	0.0004	2	6	5216	2000
0.8	0.0003	0.0006	5	2	6546	1949
1.0	0.0004	0.0006	3	2	5948	1958

Ap=50000,2000

参数对控制性能影响

不同C和Ap下的控制指标

C	E/rad		T/次		U/个	
10	0.2024	0.2530	—	—	10106	2000
20	0.1943	0.4054	—	—	9953	2000
40	0.0004	0.2708	9	—	9884	2000
80	0.0004	0.0004	2	6	5216	2000
160	0.0004	0.0007	1	3	5558	2000
320	0.2629	0.0015	—	3	12618	1998
640	0.0039	—	24	—	12971	1998
1280	0.0075	—	13	—	14467	2000

C<=40泛化能力差

收敛好
控制性能好

C>=320映射冲突增多，可能不收敛

Ap=50000,2000

鲁棒性和自适应性—量测噪声影响

不同量测噪声下的控制指标（C=80,$\eta = 0.6$）

σ	E/rad	T/次	U/个
0.5	0.0018　0.0019	3　3	5111　1882
2.0	0.0095　0.0093	4　2	8787　1989
8.0	0.0714　0.2247	2　—	21340　2000

Ap=50000,2000

θ: 0.01σ rad

$\dot{\theta}$: 0.05σ rad/s

$\ddot{\theta}$: $0.25\sigma \mathrm{rad}/s^2$

噪声增大，跟踪误差增大，但对CMAC网络学习的收敛速度没多大影响

鲁棒性和自适应性—负载变化影响

负载改变时的控制指标（C=80,$\eta=0.6$）

M_2/kg	E/rad	T/次	U/个
10	0.0008　0.0017	3　3	5079　1954
20	0.0010　0.0008	0　0	6276　1954
10	0.0004　0.0007	0　0	6280　1954
20	0.0004　0.0007	0　0	6281　1954

Ap=50000,2000

只用反馈控制时，

M_2=10kg，RMS_0=0.2075

M_2=20kg，RMS_0=0.3947

鲁棒性和自适应性—变化轨迹跟踪

跟踪不同运动轨迹时控制指标（C=80, $\eta = 0.6$）

轨迹	E/rad	T/次	U/个
A	0.0012 0.0155	3 3	10308 2000
B	0.0011 0.1269	3 —	11737 2000
C	0.0007 0.0034	0 3	12140 2000
A	0.0007 0.0056	0 1	12142 2000
B	0.0005 0.0025	0 0	12143 2000
C	0.0004 0.0019	0 0	12143 2000

只用反馈控制时RMS_0

A:0.3410

B:0.2240

C:0.2878

本讲的主要内容

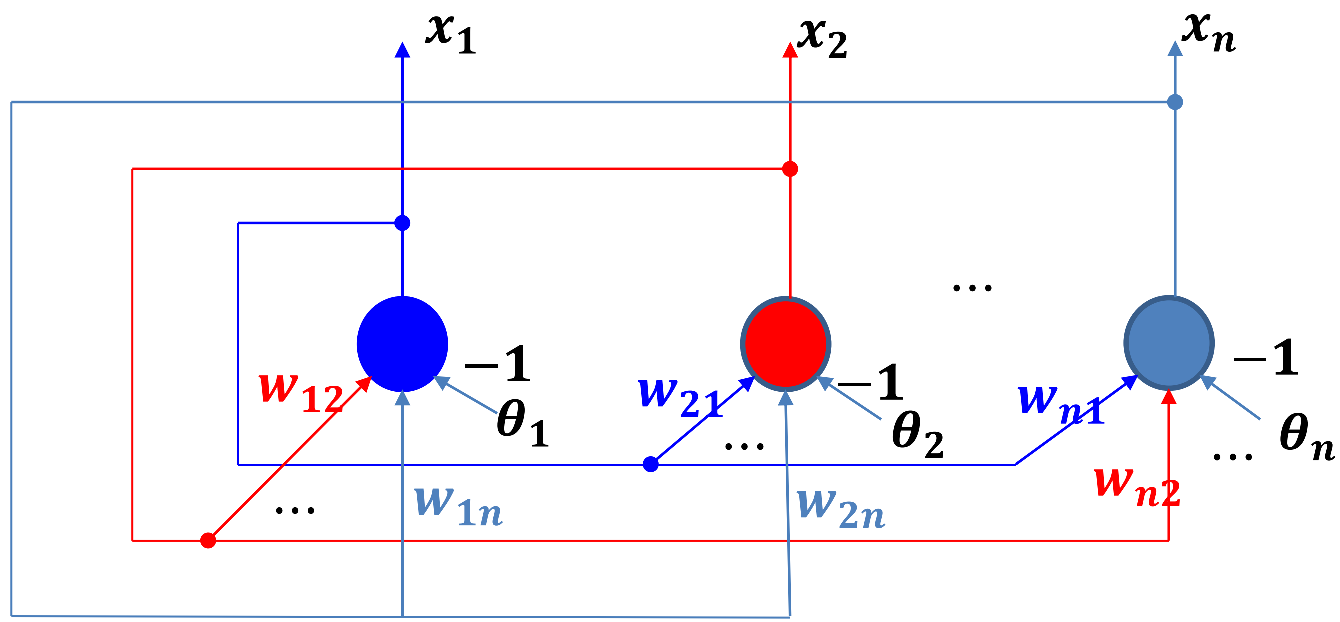
一、人工神经元控制系统

二、RBF神经网络控制

三、CMAC神经网络控制

四、Hopfield网络优化

Hopfield网络



$$\begin{cases} s_i = \sum_{j=1}^{n} w_{ij}\, x_j - \theta_i \\ x_i = f(s_i) \end{cases}$$

$$w_{ii} = 0$$

输入为网络初始状态 $x(0) = [x_1(0), x_2(0), \cdots, x_n(0)]^T$

输出为网络的稳定状态 $\lim_{k\to\infty} x(k)$

离散Hopfield网络（DHNN）

$$f(s)=\begin{cases}1, & s\geq 0\\ 0, & s<0\end{cases}\qquad f(s)=\begin{cases}1, & s\geq 0\\ -1, & s<0\end{cases}$$

网络更新方式

- **异步方式**

$$\begin{cases}x_i(k+1)=f\left(\sum_{j=1}^{n} w_{ij}x_j(k)-\theta_i\right)\\ x_j(k+1)=x_j(k), j\neq i\end{cases}$$

- **同步方式**

$$x_i(k+1)=f\left(\sum_{j=1}^{n} w_{ij}x_j(k)-\theta_i\right)$$

离散Hopfield网络的状态变换

•通常网络从某一初始状态开始经过多次更新后才可能达到某一稳态。使用异步状态更新策略有以下优点：

（1）算法实现容易，每个神经元节点有自己的状态更新时刻．不需要同步机制；

（2）以串行方式更新网络的状态可以限制网络的输出状态，避免不同稳态以等概率出现。

一旦给出HNN的权值和神经元的阈值，网络的状态转移序列就确定了。

离散Hopfield网络

由于反馈的存在，系统的演变可能出现如下几种状态

- （1）渐近稳定
- （2）极限环
- （3）混沌现象
- （4）状态轨迹发散

离散Hopfield网络

- 定义：网络状态$x = f(Wx - \theta)$，则称x为网络的**稳定点或吸引子**
- 若稳态为记忆样本，则收敛过程便是寻找记忆样本过程（联想记忆）
- 若稳态对应某种优化目标，并作为目标函数的极小点，收敛过程辨识优化计算过程

- 定理：对于离散Hopfield网络，若按异步方式调整，且**连接权值为对称阵**，对任意的初态，网络最终都收敛到一个**吸引子**。

稳定性分析

- 证明：定义网络能量函数

$$E(k) = -\frac{1}{2}x^T(k)Wx(k) + x^T(k)\theta$$

$$\Delta x(k) = x(k-1) - x(k)$$

$$\Delta E(k) = E(k+1) - E(k)$$

$$= -\Delta x^T(k)[Wx(k) - \theta] - \frac{1}{2}\Delta x^T(k)W\Delta x(k)$$

稳定性分析

■ 按异步方式

$$\Delta x(k)=[0,\cdots,0,\Delta x_i(k),0,\cdots,0]^T$$

- $\Delta E(k)=-\Delta x_i(k)\left[\sum_{j=1}^{n} w_{ij}x_j(k)-\theta_i\right]-\frac{1}{2}\Delta x_i^2 w_{ii}$
- $s_i(k)=\sum_{j=1}^{n} w_{ij}x_j(k)-\theta_i$
- $\Delta E(k)=-\Delta x_i(k)\left[s_i(k)+\frac{1}{2}\Delta x_i(k)w_{ii}\right]=$ $-\Delta x_i(k)s_i(k)$

假定节点取 -1和1两种状态，则

$$x_i(k+1)=f[s_i(k)]=\begin{cases}1 & s_i(k)\geq 0\\ -1 & s_i(k)<0\end{cases}$$

稳定性分析

- $x_i(k) = -1, x_i(k+1) = f[s_i(k)] = 1$
  $$\Delta x_i(k) = 2, s_i(k) \geq 0, \Rightarrow \Delta E(k) \leq 0$$
- $x_i(k) = 1, x_i(k+1) = f[s_i(k)] = -1$
  $$\Delta x_i(k) = -2, s_i(k) < 0, \Rightarrow \Delta E(k) \leq 0$$
- $x_i(k+1) = x_i(k) \Rightarrow \Delta E(k) = 0$
- $E(k)$收敛到常数，$\Delta E(k) = 0$

稳定性分析

- 吸引子分析 $\Delta E(k) = -\Delta x_i(k) s_i(k)$

$\Delta E(k) = 0$对应于

（a） $x_i(k+1) = x_i(k) = 1, or\ x_i(k+1) = x_i(k) = -1$

稳态

（b） $x_i(k) = -1, x_i(k+1) = 1, s_i(k) = 0$

稳态

$w_{ii} > 0$结论仍成立，而且收敛过程更快

离散Hopfield网络

■ 按同步方式

定理：对离散Hopfield网络，若按同步方式调整状态，且连接权值矩阵非负定对称，则对于任意初态，网络最终都将收敛到一个吸引子

$$\Delta E(k) = E(k+1) - E(k)$$

$$= -\Delta x^T(k)[Wx(k) - \theta] - \frac{1}{2}\Delta x^T(k) W \Delta x(k)$$

$$= -\Delta x^T(k) s(k) - \frac{1}{2}\Delta x^T(k) W \Delta x(k)$$

$$= -\sum_{i=1}^{n} \Delta x_i(k) s_i(k) - \frac{1}{2} x^T(k) W \Delta x(k)$$

$-\Delta x_i(k) s_i(k) \leq 0$ **W非负定** $\Rightarrow \boldsymbol{\Delta E(k) \leq 0}$

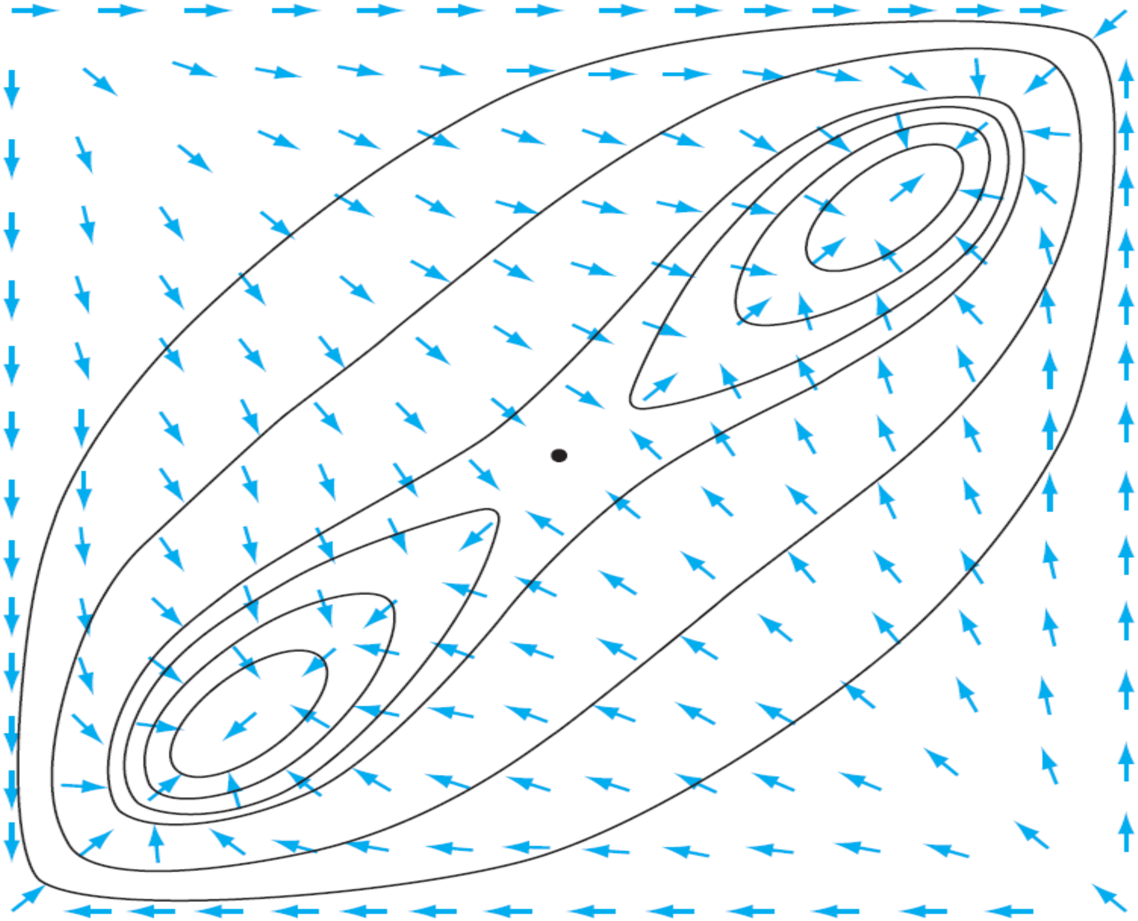
离散Hopfield网络

- 能量函数的物理意义：

在那些渐近稳定点的吸引域内，离吸引点越远的状态，所具有的能量越大，由于能量函数的单调下降特性，保证状态运动能从远离吸引点处，不断趋于吸引点，直到达到稳定点（局部能量极小点）。

注意：能量函数的选择，只是保证系统稳定和渐近稳定的充分条件，而不是必要条件，能量函数的选择也不是唯一的

离散Hopfield网络能量函数



离散Hopfield网络能量极小点的设计

- ■ 只有当网络的能量极小点可被选择和设定时，网络所具有的能力才能发挥作用。
- ■ 能量极小点的分布是由网络的连接权值和阈值所决定的。因此设计能量极小点的核心就是如何获取一组合适的参数值。
- ■ 有两种方法供选择：
- (1) 根据求解问题的要求直接设计出所需要的连接权值
- (2) 通过提供的附加机制来训练网络，使其自动调整连接权值，产生期望的能量极小点。
- 前者为静态学习方法，对于一个具体应用而言，权矩阵为定常矩阵、如TSP求解等。后者为动态学习方法，如联想记忆等。

离散Hopfield网络

■ 连接权设计　　Hebb规则

给定m个样本$x^{(k)}(k=1,2,\cdots,m), x\in\{0,1\}^n$

$$w_{ij}=\begin{cases}\sum_{k=1}^{m}\left(2x_i^{(k)}-1\right)\left(2x_j^{(k)}-1\right), i\neq j\\ 0, i=j\end{cases}$$

满足对称性要求

矩阵形式

$$W=\sum_{k=1}^{m}(2x^{(k)}-b)(2x^{(k)}-b)^T-mI$$

$$b=[1,1,\cdots,1]^T$$

离散Hopfield网络

■ **吸引子分析** $x \in \{-1,1\}^n$

● **若m个样本两两正交**

$$\begin{cases} x^{(i)T} x^{(j)} = 0, i \neq j \\ x^{(i)} x^{(i)} = n \end{cases}$$

$$Wx^{(k)} = \left(\sum_{i=1}^{m} x^{(i)} x^{(i)T} - mI \right) x^{(k)}$$

$$= \sum_{i=1}^{m} x^{(i)} x^{(i)T} x^{(k)} - m x^{(k)} = n x^{(k)} - m x^{(k)}$$

$$= (n-m) x^{(k)}$$

离散Hopfield网络

$$f(s)=\begin{cases}1, & s\geq 0\\ -1, & s<0\end{cases}$$

$$n-m>0$$
$$\Downarrow$$
$$f\left[Wx^{(k)}\right]=f\left[(n-m)x^{(k)}\right]=x^{(k)}$$

- 若m个样本$x^{(k)}(k=1,2,\cdots,m)$不是两两正交，且设内积为

$$x^{(i)T}x^{(i)}=\beta_{ij},(\beta_{ii}=0)$$

$$Wx^{(k)}=\sum_{i=1}^{m}x^{(i)}x^{(i)T}x^{(k)}-mx^{(k)}$$

$$=(n-m)x^{(k)}+\sum_{\substack{i=1\\ i\neq k}}^{m}x^{(i)}\beta_{ik}$$

离散Hopfield网络

取第j个元素

$$\left[Wx^{(k)}\right]_j = (n-m)x_j^{(k)} + \sum_{\substack{i=1\\i\neq k}}^{m} x_j^{(k)}\beta_{ik}$$

若能使$\forall j$有

$$n-m > \left|\sum_{\substack{i=1\\i\neq k}}^{m} x_j^{(k)}\beta_{ik}\right|$$

则$x^{(k)}$是网络吸引子。

$$\Updownarrow$$

$$\left|\sum_{\substack{i=1\\i\neq k}}^{m} x_j^{(k)}\beta_{ik}\right| \leq \left|\sum_{\substack{i=1\\i\neq k}}^{m} \beta_{ik}\right| \leq (m-1)\beta_m$$

$$\beta_m \triangleq \left|\beta_{ik}\right|_{max}$$

离散Hopfield网络

保证所有样本为吸引子的充分条件

$$n - m > (m-1)\beta_m$$

$$m < \frac{n+\beta_m}{1+\beta_m}$$

若m个样本满足

$$\alpha n \le d_H\left(x^{(i)}, x^{(j)}\right) \le (1-\alpha)n$$

$$i,j = 1,\cdots,m, i \ne j, 0 < \alpha < 0.5$$

$$\Downarrow$$

$$\left|\beta_{ij}\right| \le n - 2\alpha n = \beta_m$$

充分条件

$$m < \frac{2n(1-\alpha)}{1+2n(1-2\alpha)}$$

离散Hopfield网络

■ 记忆容量

- 给定网络结构参数下，保证联想功能正确，网络所能存储的最大样本数
- 样本本身不仅应为吸引子，而且具有一定的吸引域
- 不仅与节点数量有关，而且与连接权值设计有关
- 与样本本身的性质有关，输入样本正交，可获得最大容量
- $m \leq 0.15n$ Hopfield （n为网络节点数量）
- $n \to \infty \quad m \leq \dfrac{(1-2\alpha)^2 n}{2\ln n}(0 < \alpha < 0.5)$ 样本随机分布理论分析

离散Hopfield网络

■ 改进权值来提高记忆容量

m个样本$x^k(k=1,2,\cdots,m)$, 构成$n\times(m-1)$阶矩阵

$$A=\left[x^{(1)}-x^{(m)},x^{(2)}-x^{(m)},\cdots,x^{(m-1)}-x^{(m)}\right]$$

$$A=U\Sigma V^T$$

$$\Sigma=\begin{bmatrix}S & 0\\ 0 & 0\end{bmatrix}$$

$$S=\mathrm{diag}(\sigma_1,\sigma_2,\cdots,\sigma_r),U\in n\times n,V\in(m-1)\times(m-1)$$

离散Hopfield网络

设计权值W和阈值向量θ

$$W=\sum_{k=1}^{r}u_k u_k^T,\theta=Wx^{(m)}-x^{(m)}$$

$$U=[u_1,u_2,\cdots,u_r,u_{r+1},\cdots,u_n]$$

$u_1,u_2,\cdots,u_r$ 为A值域空间的正交基

$$x^{(k)}-x^{(m)}=\sum_{i=1}^{r}\alpha_i u_i$$

$$Wu_i=\sum_{k=1}^{r}u_k u_k^T u_i=u_i$$

$$W\left(x^{(k)}-x^{(m)}\right)=W\sum_{i=1}^{r}\alpha_i u_i=\sum_{i=1}^{r}\alpha_i(Wu_i)=\sum_{i=1}^{r}\alpha_i u_i$$

$$=x^{(k)}-x^{(m)}$$

离散Hopfield网络

对任一样本$x^{(k)}, (k = 1,2,\cdots,m-1)$

$$Wx^{(k)} - \theta = Wx^{(k)} - Wx^{(m)} + x^{(m)} = x^{(m)}$$

$$f\left(Wx^{(k)} - \theta\right) = f\left(x^{(k)}\right) = x^{(k)}$$

对第m个样本，有

$$Wx^{(m)} - \theta = Wx^{(m)} - Wx^{(m)} + x^{(m)} = x^{(m)}$$

$$f\left(Wx^{(m)} - \theta\right) = f\left(x^{(m)}\right) = x^{(m)}$$

所有样本都是吸引子，而并不要求两两正交

提高了网络记忆容量

离散Hopfield网络

例：

$$n=4,\theta_i=0(i=1,2,3,4),m=2$$

$$x^{(1)}=\begin{bmatrix}1\\1\\1\\1\end{bmatrix},x^{(2)}=\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}$$

连接权矩阵（Hebb**规则**）

$$W=x^{(1)}x^{(1)T}+x^{(2)}x^{(2)T}-2I=\begin{bmatrix}0&2&2&2\\2&0&2&2\\2&2&0&2\\2&2&2&0\end{bmatrix}$$

$$d_H\left(x^{(1)},x^{(2)}\right)=4,\Leftrightarrow\alpha=0$$

离散Hopfield网络

$$f\left(Wx^{(1)}\right)=f\begin{bmatrix}6\\6\\6\\6\end{bmatrix}=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}=x^{(1)}$$ 吸引子

$$f\left(Wx^{(2)}\right)=f\begin{bmatrix}-6\\-6\\-6\\-6\end{bmatrix}=\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}=x^{(2)}$$ 吸引子

假定

$$x(0)=x^{(3)}=[-1\ 1\ 1\ 1]^T$$

异步方式$(1, 2, 3, 4)$调整演变网络

$$x_1(1)=f\left(\sum_{j=1}^{n} w_{1j}x_j(0)\right)=f(6)=1$$

$$x_2(1)=x_2(0)=1,\ x_3(1)=x_3(0)=1,\ x_4(1)=x_4(0)=1$$

$$x(1)=[1\ 1\ 1\ 1]^T=x^{(1)}$$ 收敛到$x^{(1)}$

离散Hopfield网络

假定

$$x(0) = x^{(4)} = [1\ -1\ -1-\ 1]^T$$

异步方式$(1, 2, 3, 4)$调整演变网络

$$x_1(1) = f\left(\sum_{j=1}^{n} w_{1j} x_j(0)\right) = f(-6) = -1$$

$$x_2(1) = x_2(0) = -1,\ x_3(1) = x_3(0) = -1,\ x_4(1) = x_4(0) = -1$$

$$x(1) = [-1\ -1\ -1\ -1]^T = x^{(2)}$$ 收敛到$x^{(2)}$

离散Hopfield网络

假定

$$x(0) = x^{(5)} = [1\ 1\ -1 -\ 1]^T$$

异步方式$(1, 2, 3, 4)$调整演变网络　　与两个吸引子的海明距离都为2

$$x_1(1) = f\left(\sum_{j=1}^{n} w_{1j} x_j(0)\right) = f(-2) = -1$$

$$x_i(1) = x_i(0), i = 2{,}3{,}4$$

$$x(1) = [-1\ 1\ -1\ -1]^T$$

$$x_2(2) = f\left(\sum_{j=1}^{n} w_{2j} x_j(1)\right) = f(-6) = -1$$

$$x_i(2) = x_i(1), i = 1{,}3{,}4$$

$$x(2) = [-1\ -1\ -1\ -1]^T = x^{(2)}$$

收敛到$x^{(2)}$

离散Hopfield网络

假定

$$x(0) = x^{(5)} = [1\ 1\ -1 -\ 1]^T$$

异步方式$(3, 4, 1, 2)$调整演变网络　　与两个吸引子的海明距离都为2

$$x_3(1) = f\left(\sum_{j=1}^{n} w_{3j} x_j(0)\right) = f(2) = 1$$

$$x_i(1) = x_i(0), i = 1,2,4$$

$$x(1) = [1\ 1\ 1\ -1]^T$$

$$x_4(2) = f\left(\sum_{j=1}^{n} w_{4j} x_j(1)\right) = f(6) = 1$$

$$x_i(2) = x_i(1), i = 1,3,4$$

$$x(2) = [1\ 1\ 1\ 1]^T = x^{(1)}$$ 收敛到$x^{(1)}$

离散Hopfield网络

按同步方式调整

$$x(0)=x^{(3)}=[-1\ 1\ 1\ 1]^T$$
$$x(1)=f[Wx(0)]=f\left(Wx^{(3)}\right)=[1\ 1\ 1\ 1]^T$$
$$x(2)=f[Wx(1)]=[1\ 1\ 1\ 1]^T$$

收敛到$x^{(1)}$

$$x(0)=x^{(4)}=[1\ -1\ -1-\ 1]^T$$
$$x(1)=f[Wx(0)]=f\left(Wx^{(3)}\right)=[-1\ -1\ -1\ -1]^T$$
$$x(2)=f[Wx(1)]=[-1\ -1\ -1\ -1]^T$$

收敛到$x^{(2)}$

$$x(0)=x^{(5)}=[1\ 1\ -1-\ 1]^T$$
$$x(1)=f[Wx(0)]=f\left(Wx^{(3)}\right)=[-1\ -1\ 1\ 1]^T$$
$$x(2)=f[Wx(1)]=[1\ 1\ -1\ -1]^T=x(0)$$

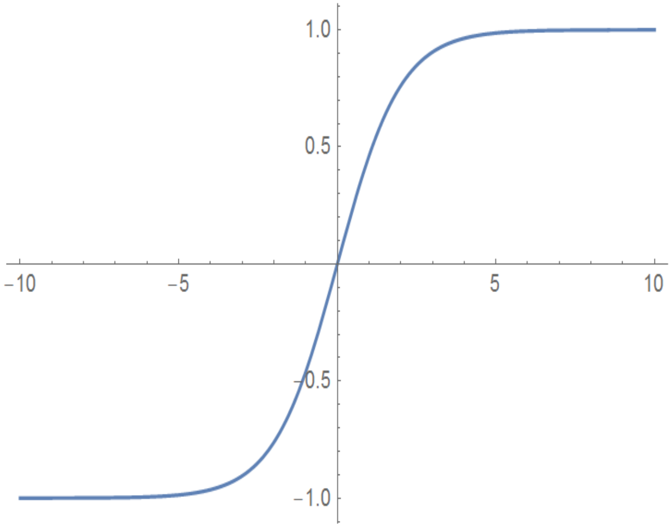
振荡

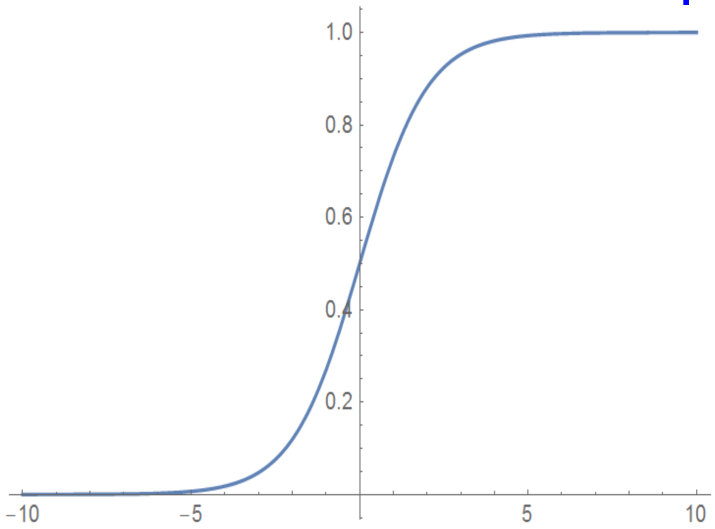
连续Hopfield网络

$$\begin{cases} s_i = \sum_{j=1}^{n} w_{ij} x_j - \theta_i \\ \dfrac{dy_i}{dt} = -\dfrac{1}{\tau} y_i + s_i \\ x_i = f(y_i) \end{cases}$$

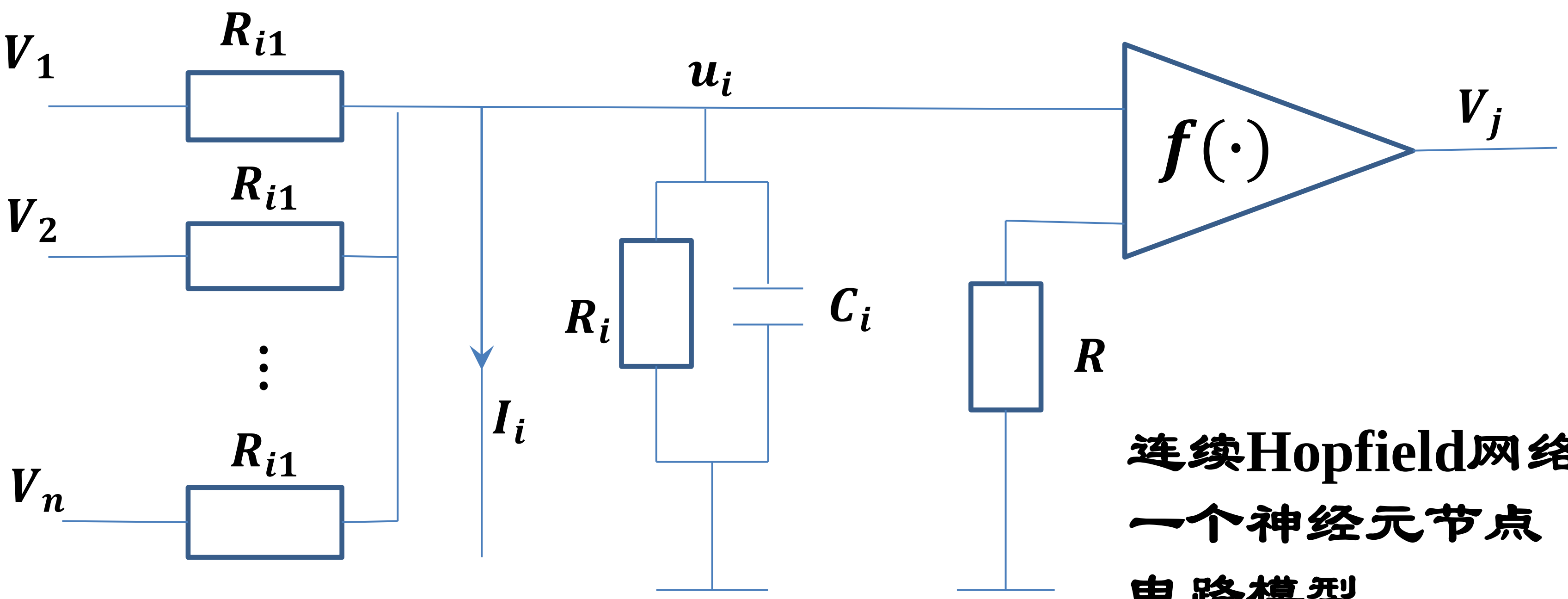
$x_i \in (-1,1)$　$x_i = f(y_i) = \dfrac{1 - e^{\mu y_i}}{1 + e^{\mu y_i}}$

$x_i \in (0,1)$　$x_i = f(y_i) = \dfrac{1}{1 + e^{\mu y_i}}$





连续Hopfield网络



连续Hopfield网络一个神经元节点电路模型

$$\begin{cases} C_i \dfrac{du_i}{dt} + \dfrac{u_i}{R_i} + I_i = \displaystyle\sum_{j=1}^{n} \dfrac{V_j - u_i}{R_{ij}} \\ V_i = f(u_i) \end{cases}$$

连续Hopfield网络

$$\begin{cases} \dfrac{du_i}{dt} = \dfrac{u_i}{R_i'C_i} - \dfrac{I_i}{C_i} + \sum_{j=1}^{n} \dfrac{V_j}{R_{ij}C_i} \\ V_i = f(u_i) \end{cases} \qquad \frac{1}{R_i'} = \frac{1}{R_i} + \sum_{j=1}^{n} \frac{1}{R_{ij}}$$

$$x_i = V_i, y_i = u_i, \tau_i = R_i'C_i, w_{ij} = \frac{1}{R_{ij}C_i}, \theta_i = \frac{I_i}{C_i}$$

$$\Rightarrow \begin{cases} s_i = \sum_{j=1}^{n} w_{ij}x_j - \theta_i \\ \dfrac{dy_i}{dt} = -\dfrac{1}{\tau_i} y_i + s_i \\ x_i = f(y_i) \end{cases}$$

连续Hopfield网络

■ 稳定性

定义连续Hopfield网络的能量函数

$$E=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_ix_j+\sum_{i=1}^{n}x_i\theta_i+\sum_{i=1}^{n}\frac{1}{\tau_i}\int_0^{x_i}f^{-1}(\eta)d\eta$$

$$\frac{dE}{dt}=\sum_{i=1}^{n}\frac{\partial E}{\partial x_i}\frac{dx_i}{dt}$$

$$\frac{\partial E}{\partial x_i}=-\sum_{j=1}^{n}w_{ij}x_j+\theta_i+\frac{1}{\tau_i}f^{-1}(x_i)=-\sum_{j=1}^{n}w_{ij}x_j+\theta_i+\frac{1}{\tau_i}y_i=-\frac{dy_i}{dt}$$

$$\frac{dE}{dt}=\sum_{i=1}^{n}\left(-\frac{dy_i}{dt}\frac{dx_i}{dt}\right)=-\sum_{i=1}^{n}\left(\frac{dy_i}{dx_i}\frac{dx_i}{dt}\frac{dx_i}{dt}\right)=-\sum_{i=1}^{n}\left(\frac{dy_i}{dx_i}\left(\frac{dx_i}{dt}\right)^2\right)$$

连续Hopfield网络

- 1）具有良好的收敛性；
- 2）具有有限个平衡点；
- 3）如果平衡点是稳定的，那么它也一定是渐进稳定的；
- 4）渐进稳定平衡点为其能量函数的局部极小点；
- 5）能将任意一组希望存储的正交化矢量综合为网络的渐进平衡点；
- 6）网络的存储信息表现为神经元之间互连的分布式动态存储；
- 7）网络以大规模、非线性、连续时间并行方式处理信息，其计算时间就是网络趋于平衡点的时间。

连续Hopfield网络

- 连续Hopfield网络用于组合优化计算

  把最优化问题的目标函数转换成网络的能量函数，把问题的变量对应于网络的状态，网络达到稳定状态时，就是它的能量函数达到最小的时候，对应于优化问题的解。

网络的计算量不会随维数的增加发生指数性巨增，对于优化问题的快速计算特别有效

连续Hopfield网络

■ 连续Hopfield网络用于优化计算一般步骤

（1）用罚函数法写出问题优化目标函数

（2）根据目标函数与Hopfield网络能量函数E确定连接权值系数

（3）写出网络动态方程

（4）选择合适初值，按网络动态演化直到收敛为止。

能量函数和网络动态关系：

$$\frac{\partial E}{\partial x_i}=-\sum_{j=1}^{n}w_{ij}x_j+\theta_i+\frac{1}{\tau_i}f^{-1}(x_i)=-\sum_{j=1}^{n}w_{ij}x_j+\theta_i+\frac{1}{\tau_i}y_i=-\frac{dy_i}{dt}$$

■ 旅行商问题（Traveling Salesman Problem, TSP）

- 已知：
  - N个城市间的相互距离，现有一推销员必须遍访N个城市，并且每个城市只能访问一次，最后又必须返回出发城市
- 问题：
  - 如何安排对这些城市的访问顺序，可使其旅行路线的总长度最短？

- 典型的组合优化问题，其可能的路径数目与城市数目N呈指数型增长的
- 很多实际应用问题，经过简化处理后，均可化为旅行商问题

■ 旅行商问题（Traveling Salesman Problem, TSP）

为将TSP问题映射成一个神经网络动态过程，Hopfield采取了换位矩阵表示法，用$N \times N$的矩阵表示商人访问过的N个城市，例如有四个城市A,B,C,D，访问路线是D→A→C→B→D, Hopfield网络输出所代表的有效解可用下表表示，其中1代表到达，0代表未到达

城市 \ 次序	1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	0	0	1	0
D	1	0	0	0

- V_{xi}表示神经元(x,i)的输出，相应的输入用U_{xi}表示，如果城市x在i位置上被访问$V_{xi}=1$,否则$V_{xi}=0$
- 针对TSP问题，Hopfield定义了如下形式的能量函数

$$
\begin{aligned}
E &= \frac{a}{2}\sum_{i=1}\sum_{x=1}\sum_{j=1,j\neq i} V_{xi}V_{xj} + \frac{b}{2}\sum_{i=1}\sum_{x=1}\sum_{y=1,y\neq x} V_{xi}V_{yi} \\
&+ \frac{c}{2}\left(\sum_{x=1}\sum_{i=1} V_{xi} - N\right)^2 \\
&+ \frac{d}{2}\sum_{x=1}\sum_{y=1,y\neq x}\sum_{i=1} d_{xy}V_{xi}\left(V_{y,i+1}+V_{y,i-1}\right)
\end{aligned}
$$

d_{xy}表示城市x,y之间的距离

Hopefield能量函数存在局部最小，不稳定等问题改进的TSP问题能量函数如下

$$E = \frac{a}{2}\sum_{x}\left(\sum_{i} V_{xi} - 1\right)^2 + \frac{a}{2}\sum_{i}\left(\sum_{x} V_{xi} - 1\right)^2 + \frac{b}{2}\sum_{x}\sum_{y}\sum_{i} V_{xi} d_{xy} V_{y,i+1}$$

孙守宇、郑君里. Hopfield 网络求解TSP的一种改进算法和理论证明. 电子学报，1995,23(1):73~78

根据稳定性分析

$$\frac{dU_{xi}}{dt} = -\frac{\partial E}{\partial V_{xi}}$$

$$= -a\left(\sum_i V_{xi} - 1\right) - a\left(\sum_y V_{yi} - 1\right) - b\sum_y d_{xy}V_{y,i+1}$$

$$T_{xi,yj} = -a\delta_{xy} - a\delta_{ij} - bd_{xy}\delta_{j,i+1}, I_{xi} = 2a$$

$$\frac{dU_{xi}}{dt} = \sum_y \sum_j T_{xi,yj} V_{xi} + I_{xi} (C_i = 1, R_i \to \infty)$$

$$\delta_{ij} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

$$\frac{dU_{xi}}{dt} = -a\left(\sum_i V_{xi} - 1\right) - a\left(\sum_y V_{yi} - 1\right) - b\sum_y d_{xy}V_{y,i+1} \quad (1)$$

- 利用Hopefield网络求解TSP问题的算法如下：
- 1）置初值$t = 0, a = 15, b = 1.0, \mu = 50$
- 2）计算N个城市之间的距离$d_{xy}(x, y = 1,2,\cdots, N)$
- 3）神经网络输入$U_{xi}(t)$的初始化在0附近产生
- 4）利用动态方程计算$\frac{dU_{xi}}{dt}$
- 5）根据一阶欧拉法离散化（1），求$U_{xi}(t+1)$

$$U_{xi}(t+1) = U_{xi}\ (t) + \frac{dU_{xi}}{dt}\Delta T$$

刘金琨,智能控制(第4版),电子工业出版社,2018

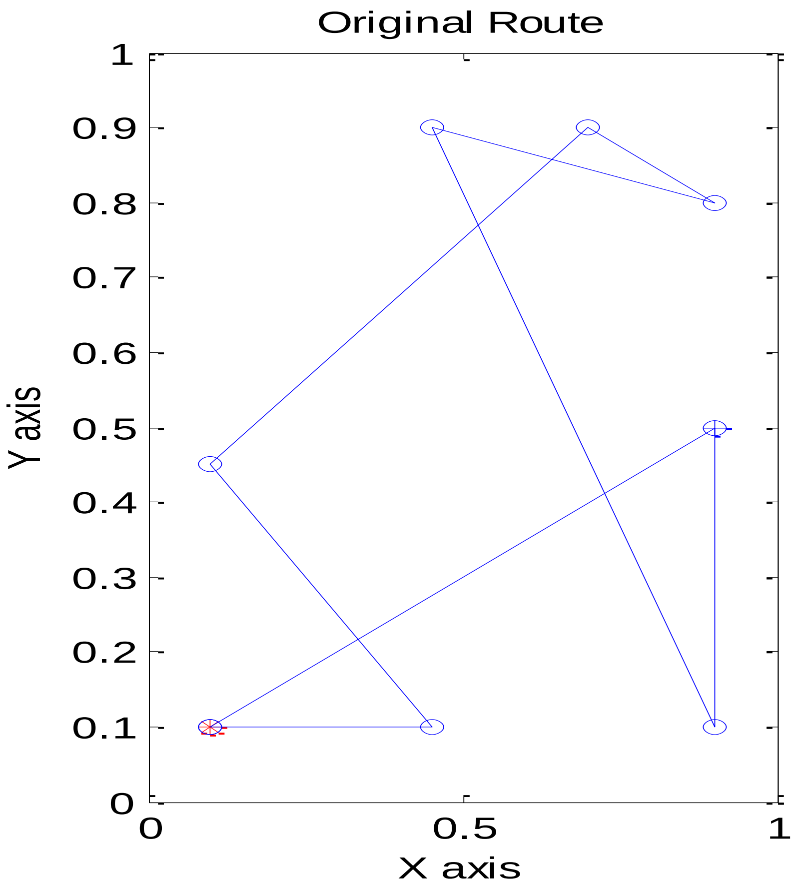
$$\frac{dU_{xi}}{dt}=-a\left(\sum_{i}V_{xi}-1\right)-a\left(\sum_{y}V_{yi}-1\right)-b\sum_{y}d_{xy}V_{y,i+1} \qquad (1)$$

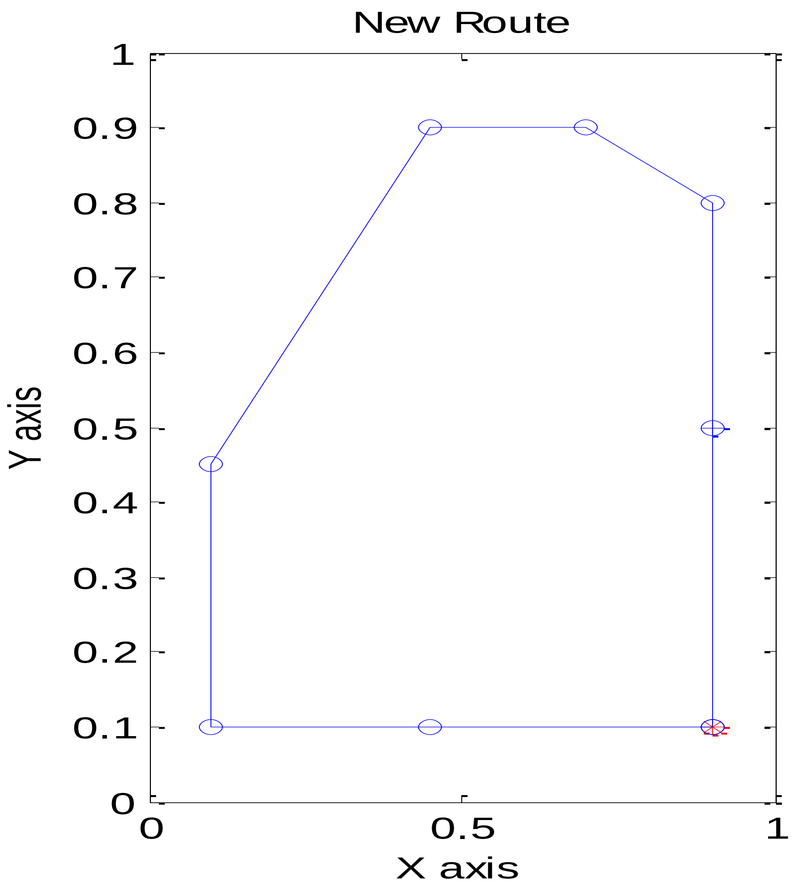
- 6）采用单调上升Sigmoid函数计算

$$V_{xi}(t)=\frac{1}{1+e^{-\mu U_{xi}(t)}}$$

- 7）计算能量函数E
- 8）检查路径合法性，判断迭代次数是否结束，如果结束，则终止，否则返回第4）步

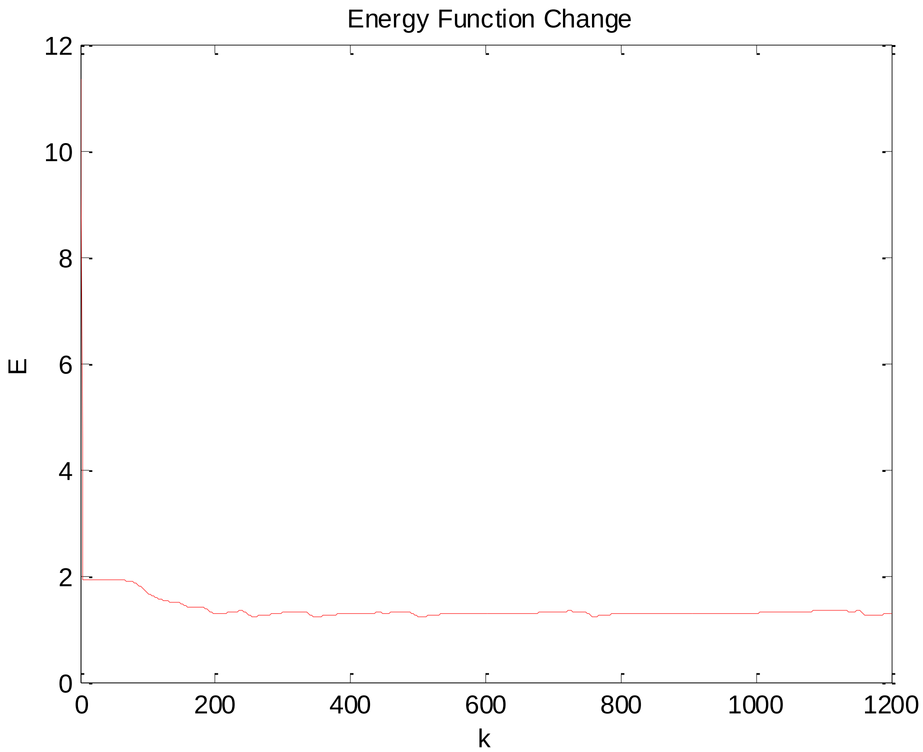
仿真实例





初始路径 **优化后的路径**

仿真实例



能量函数随迭代次数的变化

思考

- Lyapunov函数在控制中的应用？
- CAMC网络的思想及优缺点？
- 机械手控制难点？神经网络控制如何应对？
- 神经网络的容错性、联想记忆特性与网络结构的关系？
- Hopfield网络如何用于优化计算？