国家精品课程/ 国家精品资源共享课程/ 国家级精品教材
国家级十一(二)五规划教材/ 教育部自动化专业教学指导委员会牵头规划系列教材

控制系统计算机辅助设计

第10章: 自适应与智能控制系统设计

主讲：修贤超

自适应与智能控制系统设计

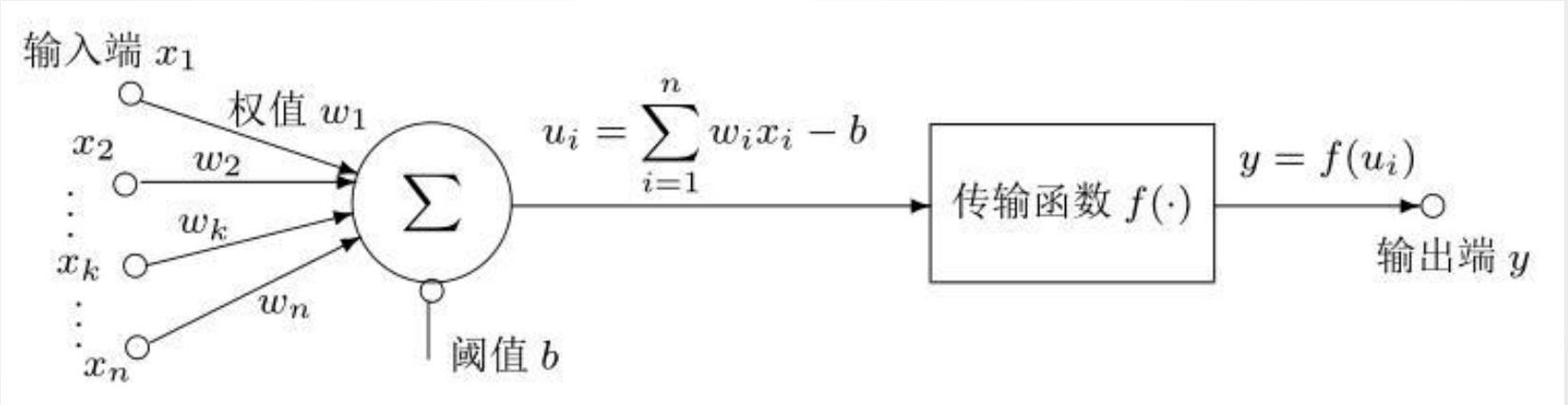
- 10.5 神经网络及神经网络控制器设计
- 10.7 全局最优控制设计

神经网络及神经网络控制器设计

- 人工神经网络是在对复杂的生物神经网络研究和理解的基础上发展起来的
- 本节主要内容
  - 神经网络简介
  - 基于单个神经元的 PID 控制器设计
  - 基于反向传播神经网络的 PID 控制器
  - 基于径向基函数的神经网络的 PID 控制器

神经网络简介

- 单个人工神经元的数学表示
  - 输入、输出、权值、阈值、激活函数

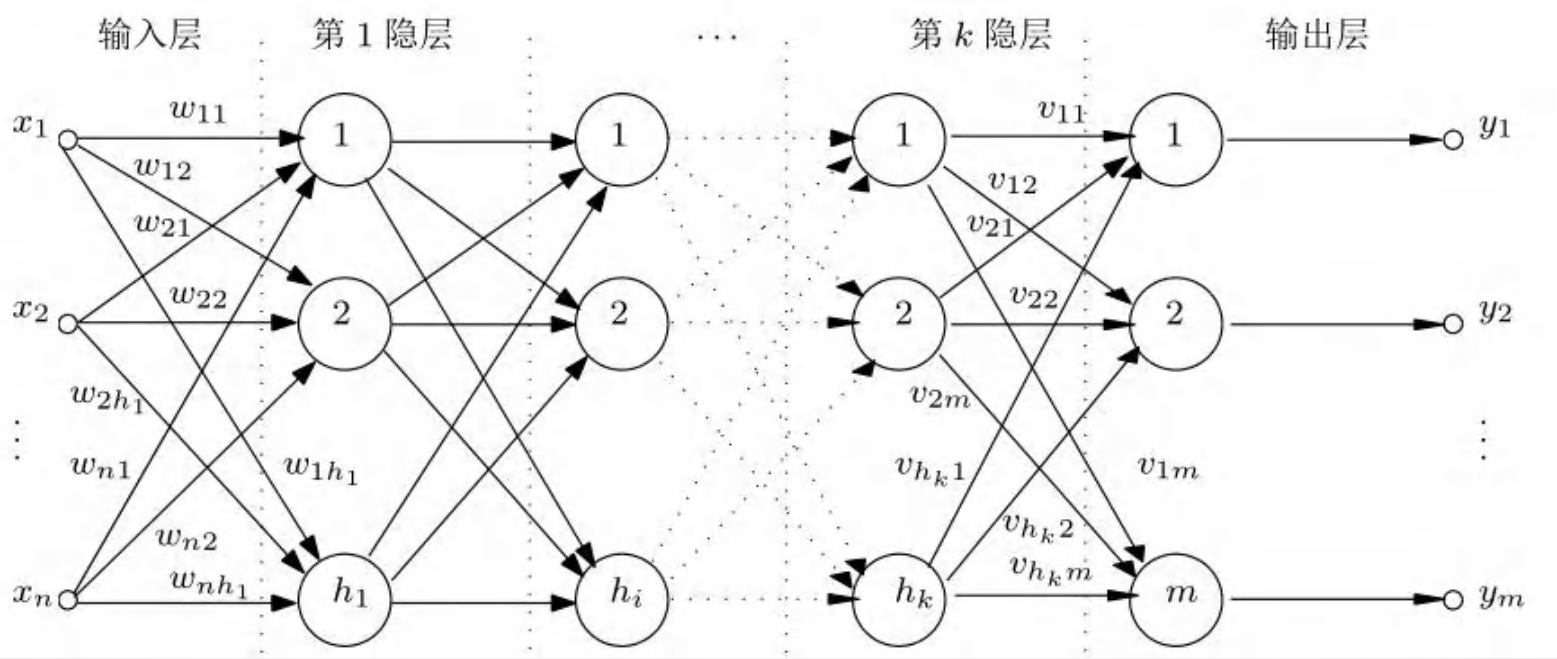


- 常用激活函数

$$f(x) = \frac{2}{1+\mathrm{e}^{-2x}} - 1 = \frac{1-\mathrm{e}^{-2x}}{1+\mathrm{e}^{-2x}}, \quad f(x) = \frac{1}{1+\mathrm{e}^{-x}}$$

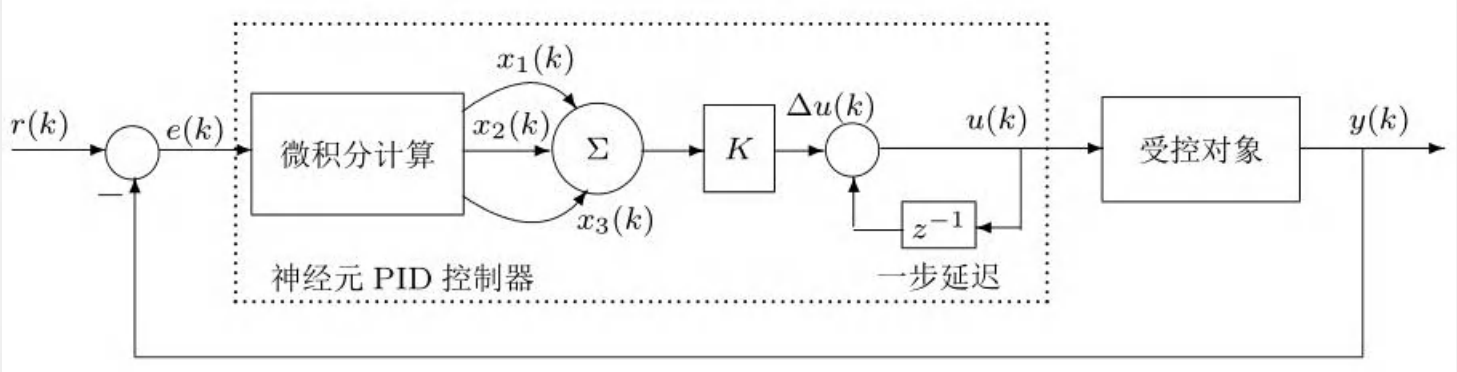
前馈神经网络

- 若干个神经元相互连接，则可以构成神经网络
- 反向传播（back propagation, BP）训练算法



基于单个神经元的PID控制器设计

- 神经元 PID 控制系统框图



- Hebb 学习速率

$$\begin{cases} w_1(k) = w_1(k-1) + \eta_{\mathrm{p}} e(k)u(k)\Big[e(k) - \Delta e(k)\Big] \\ w_2(k) = w_2(k-1) + \eta_{\mathrm{i}} e(k)u(k)\Big[e(k) - \Delta e(k)\Big] \\ w_3(k) = w_3(k-1) + \eta_{\mathrm{d}} e(k)u(k)\Big[e(k) - \Delta e(k)\Big] \end{cases}$$

神经元控制的数学公式

- 微积分信号

$$x_1(k) = e(k),\ x_2(k) = \Delta e(k) = e(k) - e(k-1),$$

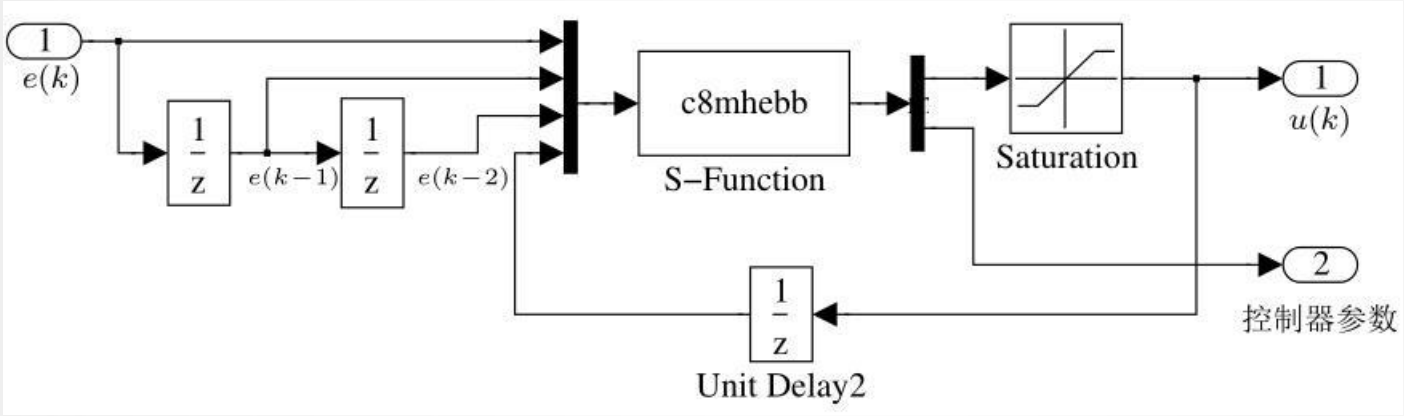
$$x_3(k) = \Delta^2 e(k) = e(k) - 2e(k-1) + e(k-2)$$

- 控制律 $u(k) = u(k-1) + K\sum_{i=1}^{3} w_i^0(k) x_i(k)$

- 归一化权值 $w_i^0(k) = \dfrac{w_i(k)}{\sum_{i=1}^{3} |w_3(k)|}$

控制器的Simulink实现

- 控制器模型 （c10shebb.mdl）




```
function [sys,x0,str,ts]=c10mhebb(t,x,u,flag,deltaK)
switch flag,
   case 0, [sys,x0,str,ts]=mdlInitializeSizes;
   case 2, sys=mdlUpdate(t,x,u,deltaK);
   case 3, sys = mdlOutputs(t,x,u);
   case {1, 4, 9}, sys = [];
   otherwise, error(['Unhandled flag = ',num2str(flag)]);
end;
```

S-函数（续）

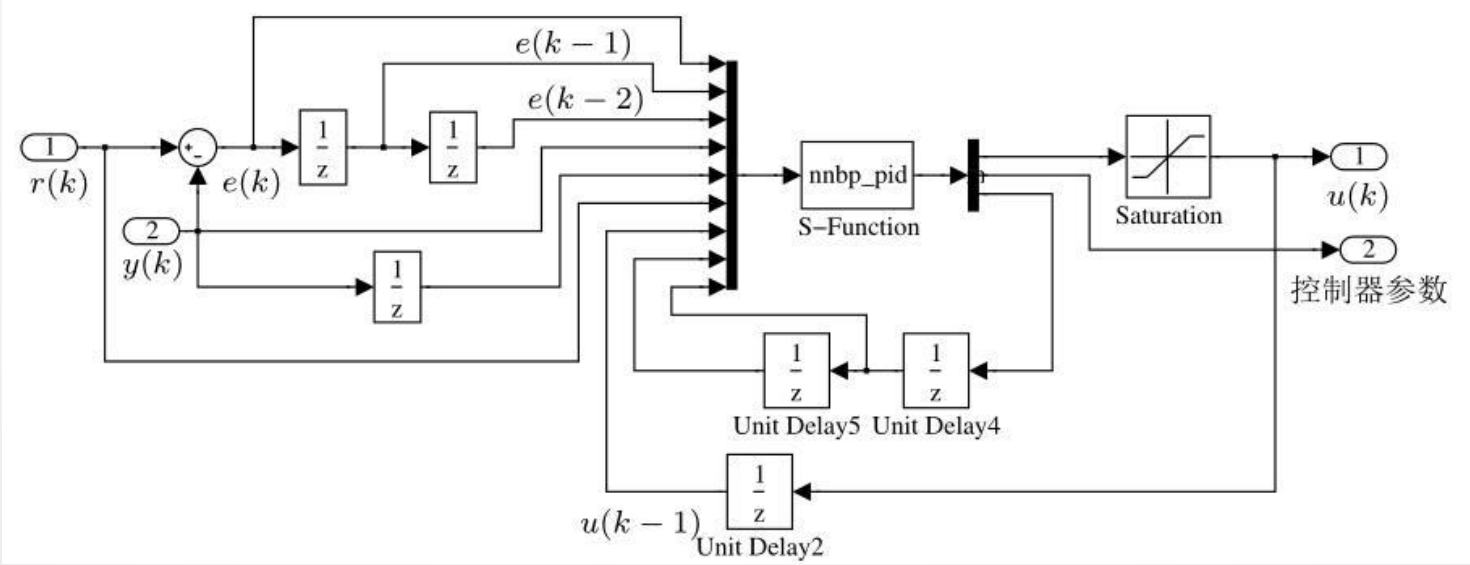
```
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0; sizes.NumDiscStates = 3;
sizes.NumOutputs = 4; sizes.NumInputs = 4;
sizes.DirFeedthrough = 1; sizes.NumSampleTimes = 1;
sys = simsizes(sizes); x0 = [0.3*rand(3,1)];
str = []; ts = [-1 0];
function sys = mdlUpdate(t,x,u,deltaK)
sys=x+deltaK*u(1)*u(4)*(2*u(1)-u(2));
function sys = mdlOutputs(t,x,u)
xx= [u(1)-u(2) u(1) u(1)+u(3)-2*u(2)];
sys=[u(4)+0.12*xx*x/sum(abs(x)); x/sum(abs(x))];
```

基于反向传播神经网络的PID控制器

- 增量式PID控制器

$$u(k)=u(k-1)+K_{\mathrm{p}}\left[e(k)-e(k-1)\right]$$
$$+K_{\mathrm{i}}e(k)+K_{\mathrm{d}}\left[e(k)+e(k-2)-2e(k-1)\right]$$

- Simulink模型 （c10bp_pid.mdl）

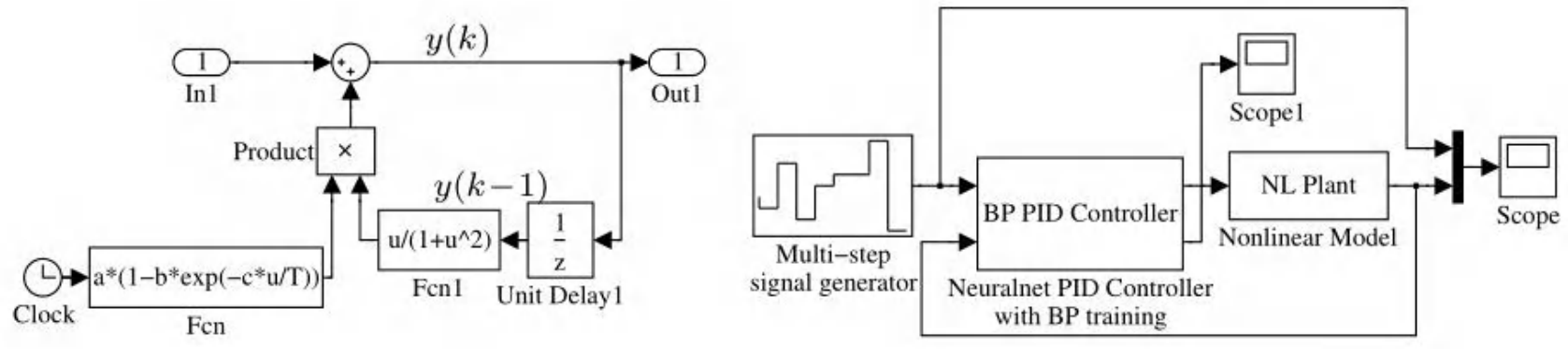


例10-9 BP PID控制系统仿真

- 非线性受控对象

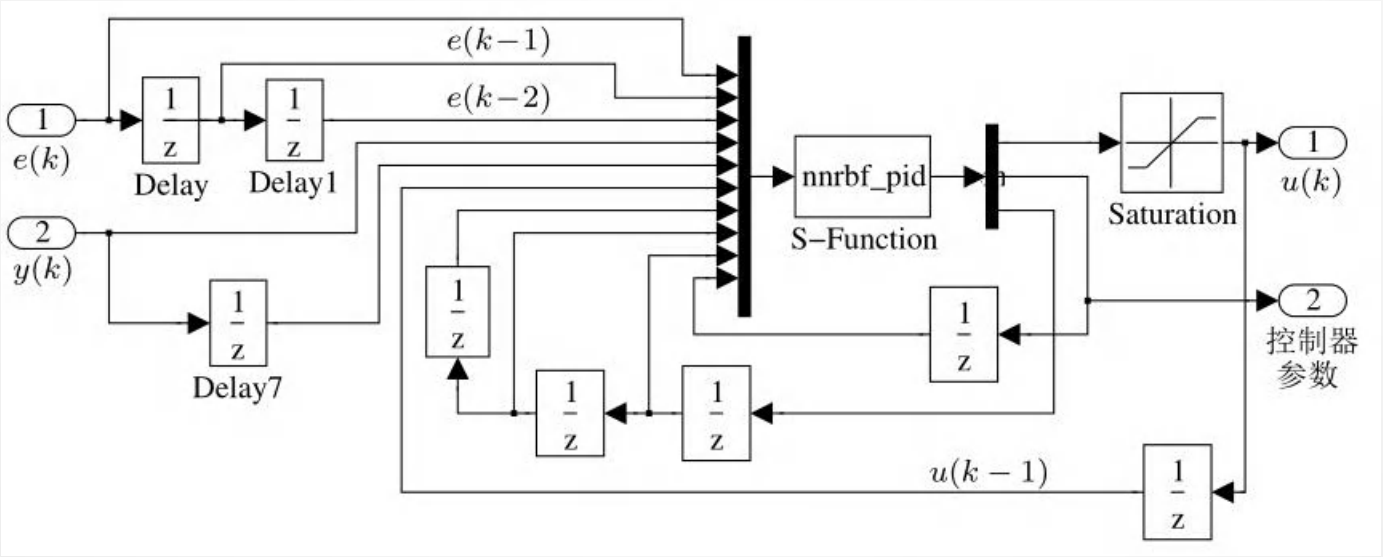
$$y(t)=\frac{a\left(1-b\mathrm{e}^{-ct/T}\right)y(t-1)}{1+y(t-1)^2}+u(t)$$

- 参数 $a=1.2, b=0.8, c=0.1, T=0.001$
- 仿真框图（c10bp_pid.mdl子模型）



基于径向基函数的神经网络的PID控制器

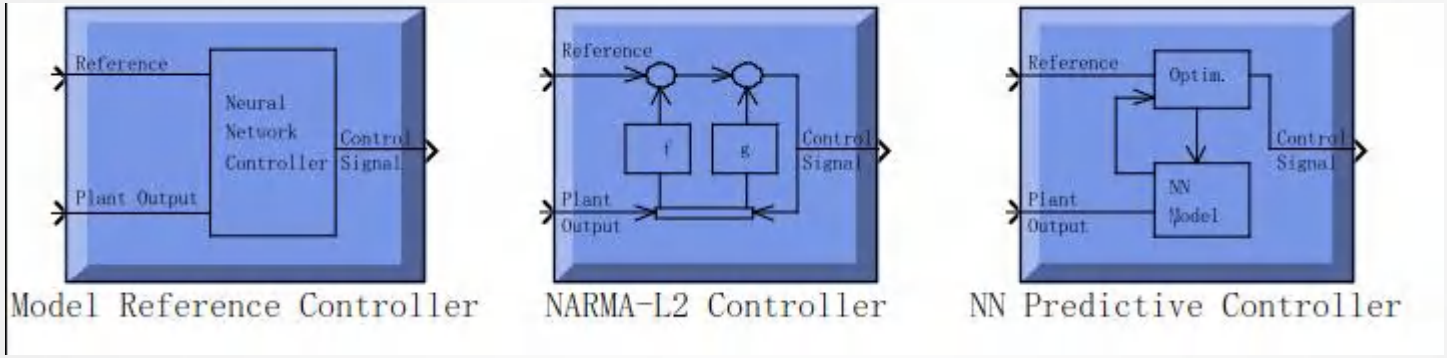
- 径向基函数（radial basis function，RBF）神经网络是一种采用局部接受域来进行函数映射的人工神经网络，是由一个隐含层和一个线性输出层构成的前向网络结构



c10mrbf

Simulink提供的控制器模块

- 打开模块



自适应与智能控制系统设计

- 10.5 神经网络及神经网络控制器设计
- 10.7 全局最优控制设计

进化算法及其在最优化问题中的应用

- 传统最优化方法的最大问题——局部最优
- 遗传算法
  - 遗传算法的基本概念与主要MATLAB工具
  - 遗传算法在求解最优化问题中的应用举例
  - 遗传算法在有约束最优化问题中的应用
- 其他全局优化算法
  - 粒子群优化、模式搜索方法、模拟退火方法
- 求取精确的全局最优解

遗传算法的基本概念

- 遗传算法是基于进化论，在计算机上模拟生命进化机制而发展起来的一门新技术
  - 它根据适者生存、优胜劣汰等自然进化规则搜索和计算问题的解，并行的搜索方法
  - 美国 Michigen 大学 John Holland 1975 年提出
  - 更可能获得全局最优解

简单遗传算法的一般步骤

1. 选择 n 个个体构成初始种群 $\boldsymbol{P}_0$，求出各个个体的函数值，$\boldsymbol{P}_0$ 可以随机生成
2. 设置代数为 $i=1$，即设置其为第一代。
3. 计算选择函数的值，所谓选择即通过概率的形式从种群中选择若干个个体的方式
4. 通过染色体个体基因的复制、交叉、变异等创造新的个体，构成新的种群 $\boldsymbol{P}_{i+1}$
5. $i=i+1$，若终止条件不满足，则继续进化

遗传算法和传统优化算法比较

- 传统最优化算法从初始点开始搜索，遗传算法从种群（若干个初值）开始进行并行搜索
- 遗传算法并不依赖于目标函数导数信息或其他辅助信息来进行最优解搜索
- 遗传算法采用的是概率型规则而不是确定性规则，所以每次得出的结果不一定完全相同，有时甚至可能会有较大的差异。

遗传算法的MATLAB函数

- MATLAB 的全局优化工具箱
  - 早期版本称为遗传算法与直接搜索工具箱
  - 大量其他寻优算法，包括粒子群优化算法、模拟退火算法与模式搜索算法等
  - 一些方法号称能够求解有约束最优化问题
- 英国Sheffield大学的遗传算法工具箱
- MATLAB Central 下有大量工具箱
  - GAOT：最早的遗传算法最优化工具箱

遗传算法求解函数

- GAOT 工具箱中的函数调用格式
  - 最大值计算、目标函数的不同格式

$$[a,\mathrm{b},\mathrm{c}]=\mathrm{gaopt}(\mathrm{bound},fun)$$

- MATLAB 全局优化工具箱

$$[x,f,\mathrm{flag},\mathrm{out}]=\mathrm{ga}(\mathrm{fun},n,\mathrm{opts})$$

粒子群优化算法与求解

- 粒子群优化算法是一种进化算法，该算法是受生物界鸟群觅食的启发而提出的搜索食物
- 假设某个区域内有一个全局最优点，和位于随机初始位置的粒子
  - 每一个粒子有到目前为止自己的个体最优值 $p_{i,\mathrm{b}}$
  - 整个粒子群有到目前为止群体的最优值 g_{b}

粒子群算法的求解函数

- MATLAB 2014b新的粒子群优化函数

$$[x, f_{\mathrm{m}}, \mathtt{key}] = \mathtt{particleswarm}(problem),$$

$$[x, f_{\mathrm{m}}, \mathtt{key}] = \mathtt{particleswarm}(f, n, x_{\mathrm{m}}, x_{\mathrm{M}}, opts)$$

  - 无约束最优化问题
  - 不能求解真正的有约束问题
  - 算法的效率比其他已知粒子群优化算法函数高

全局优化工具箱其他函数

- 无约束优化问题求解
  - 模拟退火方法 simulannealbnd

$$x = \texttt{simulannealbnd}(f, x_0, x_{\mathrm{m}}, x_{\mathrm{M}}, opts)$$

$$x = \texttt{simulannealbnd}(problem)$$

- 有约束最优化问题
  - 模式搜索方法 patternsearch

$$[x, f_{\mathrm{opt}}, \texttt{flag}, \texttt{c}] = \texttt{patternsearch}(problem)$$

$$[x, f_{\mathrm{opt}}, \texttt{flag}, \texttt{c}] = \texttt{patternsearch}(fun, x_0, A, B, A_{\mathrm{eq}}, B_{\mathrm{eq}}, \ldots x_{\mathrm{m}}, x_{\mathrm{M}}, CFun, \texttt{OPT}, p_1, p_2, \cdots)$$

例10-11 最优化问题求解

- 改进的Rastrigin函数

$$f(x_1,x_2)=20+(x_1/30-1)^2+(x_2/20-1)^2$$
$$-10[\cos(x_1/30-1)\pi+\cos(x_2/20-1)\pi],$$

- 目标函数曲面

```
>> f=@(x,y)20+(x/30-1).^2+(y/20-1).^2-...
        10*(cos(pi*(x/30-1))+cos(pi*(y/20-1)));
   ezsurf(f,[-100,100])
```

全局最优求解

- 目标函数描述

```
>> f=@(x)20+(x(1)/30-1)^2+(x(2)/20-1)^2-...
        10*(cos(pi*(x(1)/30-1))+cos(pi*(x(2)/20-1)));
```

- 遗传算法与粒子群方法

```
>> x=ga(f,2), [x g]=particleswarm(f,2)
```

- 模拟退火方法与模式搜索

```
>> x0=8*rand(2,1);  x=patternsearch(f,x0)
   [x g]=simulannealbnd(f,x0)
```

例10-16 用全局优化算法取代常规算法

- 不稳定受控对象的控制器设计

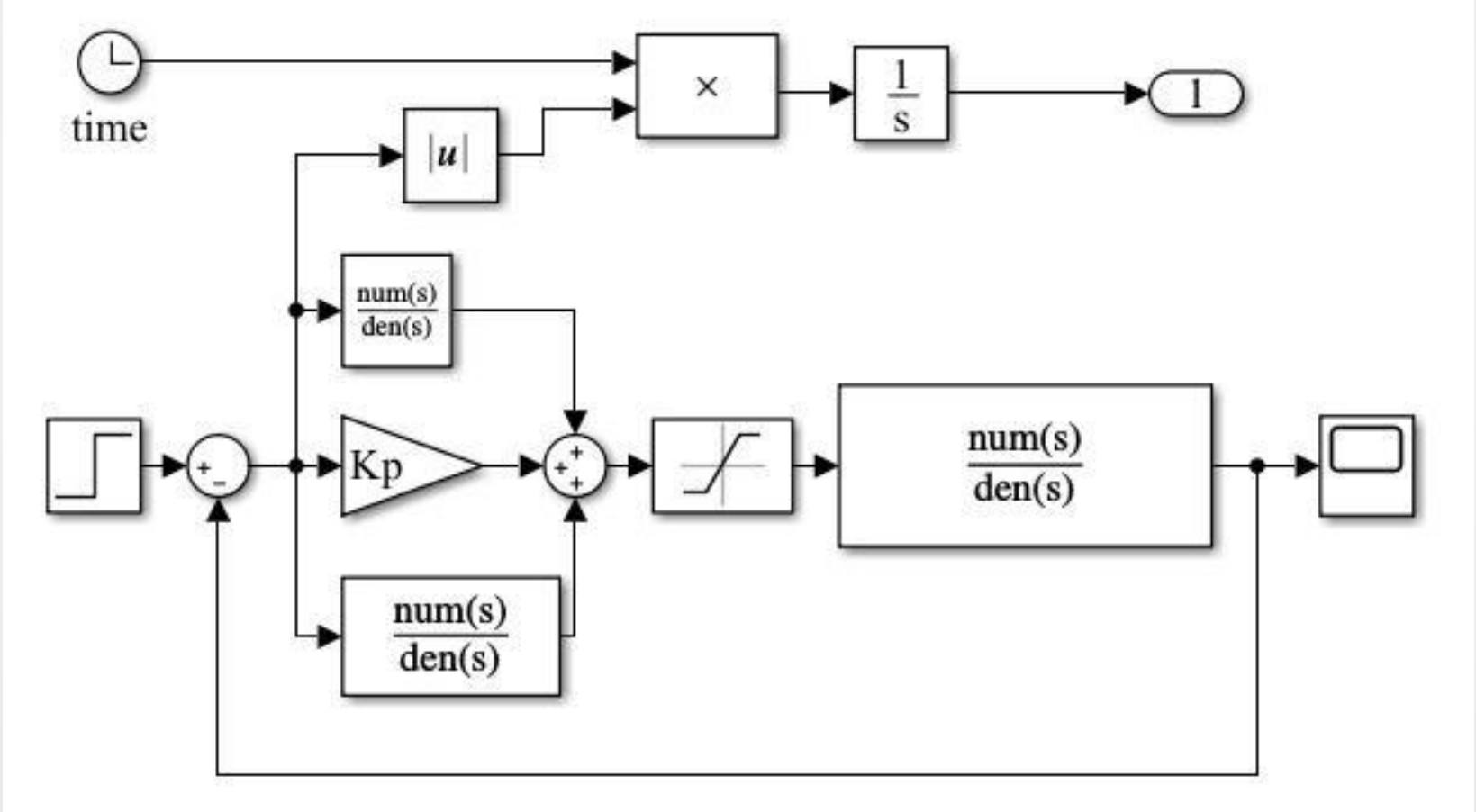
$$G(s)=\frac{s+2}{s^4+8s^3+4s^2-s+0.4}$$

```
>> num=[1 2]; den=[1,8,4,-1,0.4];
```

- 传统寻优方法不易找到一个稳定的初值
- 可以绘制Simulink模型，用OCD直接求解
- 也可以写出目标函数，然后给出全局搜索命令直接寻优

Simulink模型

- 控制系统模型与目标函数计算 c10munsta.slx



全局优化算法求解最优化问题

- 改写目标函数

```
function y=c10funun_1(x)
assignin('base','Kp',x(1)); assignin('base','Kd',x(2));
assignin('base','Ki',x(3));
[t_time,a,y_out]=sim('c10munst_a',[0,10]);  y=y_out(end,1);
```

- 用粒子群优化算法求解

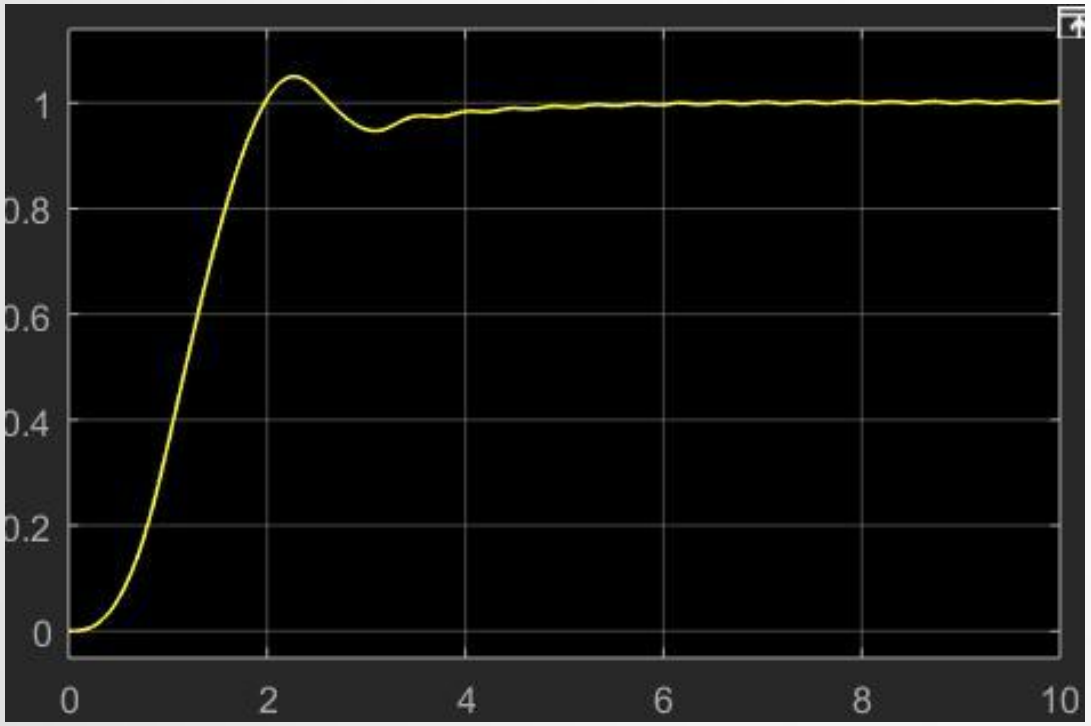
```
>> x1=particleswarm('c10funun_1',3), c10funun_1(x1(1:3))
```

- 直接搜索方法寻优

```
>> x2=patternsearch(@c10funun_1,rand(3,1))
```

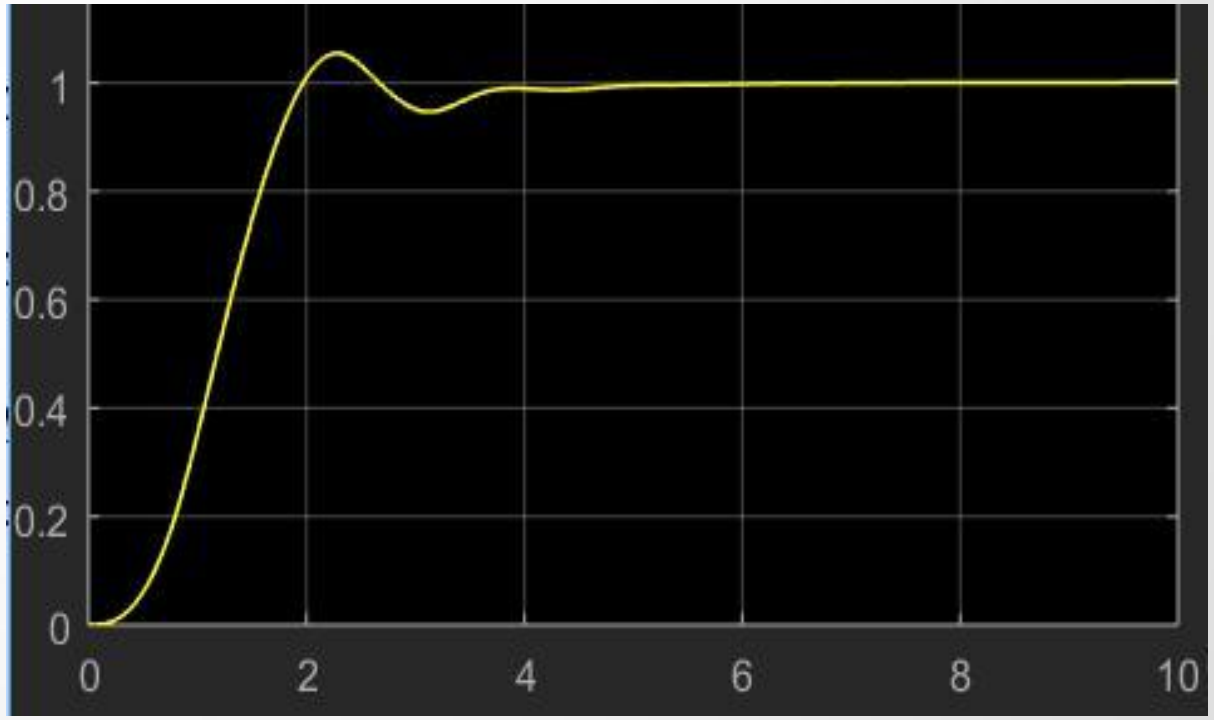
两种方法比较

- 粒子群算法 1.0837
  - Kp=10982, Ki=12698, Kd=30.7379;



模式搜索算法 1.0428

Kp=36.0111,Ki=41.6803, Kd=0.1911



神经网络及控制小结

- 神经网络的基本概念
  - 神经元 —— 权值、阈值、激活函数
  - 神经网络 —— 网络结构、节点
- 不同的神经网络控制器
  - 单个神经元的控制器
  - 基于BP神经网络的控制器仿真模型
  - 基于径向基网络的控制器仿真模型

智能优化算法小结

- MATLAB全局优化工具箱
  - 遗传算法、粒子群算法、模式搜索、模拟退火算法
- 测试问题的求解 —— 更可能得到全局最优解
- 在控制器最优设计中的应用
  - 不稳定受控对象的最优PID设计
- 可以将算法嵌入OCD与OptimPID

Q & A

感谢您的聆听和反馈