《机器人导论实验》报告

|  |  |  |
| --- | --- | --- |
| **第四次实验** | **日期：2022-11-12** | **得分：** |
| **学号：** | **姓名：*Steven*** | **专业：智能科学与技术** |

# 实验准备

安装Matlab，熟悉Simulink基本流程。

理解PID控制原理。

# 实验原理

PID基本要素由比例、积分、微分构成，其控制公式为：

每一项要素完成不同任务，对系统功能产生不同的影响，其原理框图如图5.2所示。



图1 PID原理框图

其中比例、积分、微分调节的作用分别为：

1. 比例调节：即时反馈控制系统的偏差信号 e(t)，偏差一旦产生，调节器立即产生控制作用以减小偏差。
2. 积分调节：主要用于消除静态误差，提高系统的无差度。积分作用的强弱取决于积分时间常数T\_i，T\_i越大，积分作用越弱，反之越强。
3. 微分调节：能反应偏差信号的变化趋势（变化速率），并能在偏差信号的值变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减小调节时间。

# 实验任务

通过Matlab或Octave或其他仿真软件，仿真出一个PID动图。

# 实验过程与实验结果

本次实验首先要根据PID控制原理，在Simulink中搭建模型。于是按照其流程，构建了如图所示的简单模型。

图示

描述已自动生成

图２ PID仿真控制模型

同时经过调试，设定的P、I、D值如下，使得输出最终能够收敛。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面

描述已自动生成

图３ P、I、D的取值设置

随后是绘制动态曲线图，使用Matlab完成。由于不清楚如何用Matlab代码调用Simulink程序，故使用Matlab根据已有结果重新绘制。绘图代码如下

|  |
| --- |
| clear all  close all  clc  G = tf(1, [2 1]); % 传递函数1：1/(2\*s + 1)  G\_2 = tf(3, [2 1]); % 传递函数2：3/(2\*s + 1)  % Kp的作用  for Kp = 0:0.1:3      G1 = Kp;      sys = feedback(G1 \* G \* G\_2, 1); % 单位负反馈系统      step(sys);      title(" ");      xlabel("t", "Interpreter", "latex");      ylabel("u(s)", "Interpreter", "latex");      text(0.5, 0.9, "Kp="+num2str(Kp), "Interpreter", "latex", "Units", "normalized");      text(0.5, 0.8, "$K\_i=0$", "Interpreter", "latex", "Units", "normalized");      text(0.5, 0.7, "$K\_d=0$", "Interpreter", "latex", "Units", "normalized");      axis([0 10 0 1.6]);      pause(0.01);  end  % Ki的作用  Kp = 3;  for Ki = 0:0.05:1      G1 = tf([Kp Ki], [1 0]); % 比例-积分控制 (Kp\*s + Ki) / s      sys = feedback(G1 \* G \* G\_2, 1); % 单位负反馈系统      step(sys);      title(" ");      xlabel("t", "Interpreter", "latex");      ylabel("u(s)", "Interpreter", "latex");      text(0.5, 0.9, "$K\_p=3$", "Interpreter", "latex", "Units", "normalized");      text(0.5, 0.8, "Ki="+num2str(Ki), "Interpreter", "latex", "Units", "normalized");      text(0.5, 0.7, "$K\_d=0$", "Interpreter", "latex", "Units", "normalized");      axis([0 10 0 1.6]);      pause(0.01);  end  % Kd的作用  Kp = 3;  Ki = 1;  for Kd = 0:0.1:2      G1 = tf([Kd, Kp, Ki], [1 0]); % PID控制 (kd\*s^2 + Kp\*s + Ki) / s      sys = feedback(G1 \* G \* G\_2, 1); % 单位负反馈系统      step(sys);      title(" ");      xlabel("t", "Interpreter", "latex");      ylabel("u(s)", "Interpreter", "latex");      text(0.5, 0.9, "$K\_p=3$", "Interpreter", "latex", "Units", "normalized");      text(0.5, 0.8, "$K\_i=1$", "Interpreter", "latex", "Units", "normalized");      text(0.5, 0.7, "Kd="+num2str(Kd), "Interpreter", "latex", "Units", "normalized");      axis([0 10 0 1.6]);      pause(0.01);  end |

代码与执行结果如下

图形用户界面, 应用程序

描述已自动生成

图４ 动态ＰＩＤ曲线代码及结果

详细过程视频将会放在压缩包的附件中。

# 实验心得

虽然第一次使用Simulink，在仿真环境中配置时一开始有些不知所措，但由于已经比较习惯Matlab了，很快就完成了仿真模型。

相比之下，PID原理则有些难以理解，之前在上课时，虽然内心明白什么是积分、微分，但是并不清楚为什么要这么做，也就是“知其然不知其所以然”。之后看了一个[视频](https://www.bilibili.com/video/BV1xQ4y1T7yv/)，这其中使用了“调节水温”的过程打比方，才明白了为什么要这么做，也因此受益匪浅。