11

實例探究

閱讀完本章後，讀者將可以：

1.實驗他在設計和實時方面的知識機電系統的實施。

2.進行從 A 到 Z 的機電系統設計。

3.能夠執行機電一體化設計的不同階段系統。

4.能夠解決控制問題並建立控制律我們必須實時實施。

5.能夠使用 C 語言的中斷概念編寫程序實時實施。

11.1 引言

在前面的章節中，我們開發了一些概念，說明了它們的應用。通過學術實例向讀者展示這些結果如何應用。 更多具體來說，我們已經了解瞭如何設計機

電一體化系統，介紹了成功完成所需設計所必須遵循的不同步驟機電系統。 我們已經介紹了我們必須在設計：

- 機械部分
- 電子線路
- C 語言中用於實時執行的程序

這些工具已用於一些實際系統，並提供了更多詳細信息幫助讀者執行自己的設計。

對於控制算法，我們介紹的大多數示例都是學術性的具有完美的模型。不幸的是，對於實際的系統，我們將擁有只是可以在某些特定條件下描述系統的實現，並且由於某些原因，該模型無法在實際操作中按預期運行，
算法的時間執行。這可能是由於不同的忽視動態可能會改變某些頻率的行為。

本章的目的是向讀者展示我們如何實時實施我們在前幾章中為實際應用開發的理論結果系統。我們將逐步進行並顯示所有步驟，以簡化操作過程。讀者。我們在本章中考慮的案例研究是討論和在前幾章中進行了設計。

11.2 直流電動機套件的速度控制

作為第一個示例，讓我們考慮驅動直流電動機的直流電動機的速度控制，機械部分。該示例的選擇非常重要，因為大多數係統將使用這種直流電動機。我們將考慮的直流電動機由 Maxon 製造公司。該電動機非常重要，因為它帶有齒輪箱（比率 6：1），並且編碼器，每轉給出一百個脈衝，每轉給出 600 個脈衝通過使用正交方法將其發展到 2000 年的革命每轉四百個脈衝。在此示例中，我們使用的系統如果更靈活，我們將在前面介紹控制算法的實時實現並提供更多優勢。

該電動機的數據表給出了所有重要參數，因此容易獲得該執行器的傳遞函數。我們正在考慮的負載在這個例子中是一個帶有刻度的小磁盤，我們想控制速度然後就位。這種設置在圖 1 和 2 中示出。 11.1-（11.2）。磁盤我們他們認為直徑等於 0.06 m，質量等於 0.050 Kg。用這些數據和直流電動機的數據表之一，我們可以獲得轉移磁盤速度和輸入電壓之間的函數。

首先讓我們專注於負載的速度控制。在這種情況下建立該系統的傳遞函數（直流電動機執行器及其負載）可以使用數據表，磁盤上的信息以及 Boukas [1]中的結果或繼續進行識別。 使用第一種方法得出第二章的結果
在[1]中，我們得到：

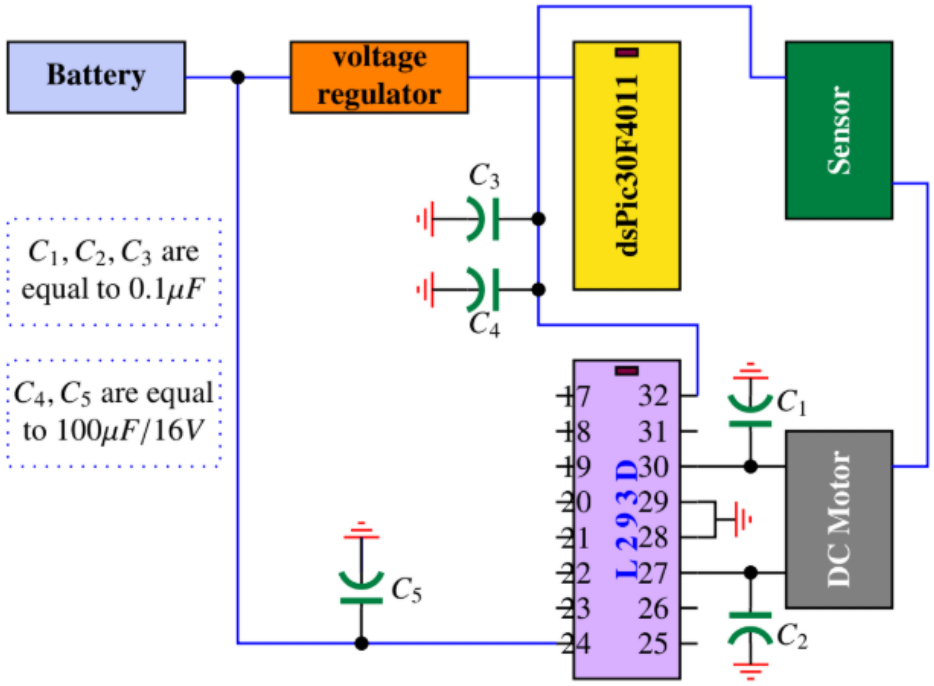

Fig. 11.1 Electronic circuit of dc motor kit

$$G(s) = \frac{K}{\tau s + 1}$$

with

$$K = 48.91$$
$$\tau = 63.921\ ms$$

為了識別我們的系統，我們可以使用時間實施設置和適當的 C 程序。自微控制器擁有有限的內存，識別可以分為兩個步驟。首先，首先實驗確定增益 K，然後使用該增益計算穩態狀態值，可用於計算恆定時間 τ 。

為了設計控制器，我們首先應該指定性能就像我們的系統一樣。作為首要性能，我們要求系統穩定。還需要係統速度在瞬變時具有良好的性能穩態狀態下零誤差的狀態參考，以供階躍參考。為了瞬態，我們希望負載的穩定時間小於或等於 $3\tau/5$ 超調量小於或等於 5%。

為了完成適當控制器的設計，我們可以在連續時間內進行設計，然後獲得我們應該在軟件部分，或直接在離散時間內進行所有設計。在其餘的在此示例中，我們將選擇第二種方法。

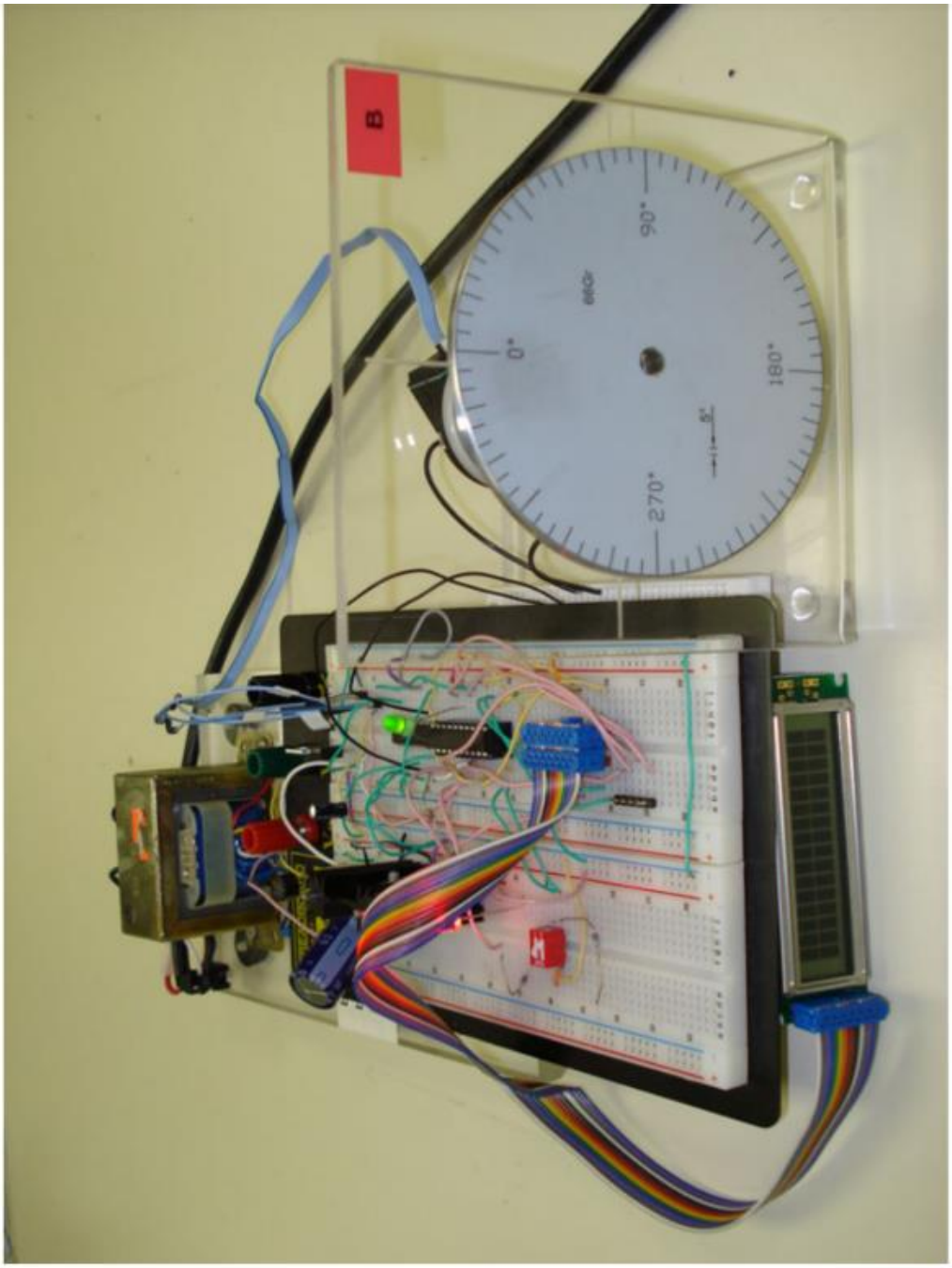

Fig. 11.2 Real-time implementation setup

根據系統傳遞函數的表達式以及所需的性能要處理該結果，我們至少需要一個比例和積分器（PI）控制器。

該控制器的傳遞函數由下式給出：

$$C(s) = K_P + \frac{K_I}{s}$$

其中要確定 KP 和 KI 的增益以迫使負載具有我們強加的表演。

使用零序持有者和 Z-transform 表，我們得到：

$$\begin{aligned} G(z) &= \frac{Kz\left(1-e^{-\frac{T}{\tau}}\right)\left(1-z^{-1}\right)}{(z-1)\left(z-e^{-\frac{T}{\tau}}\right)} \\ &= \frac{K\left(1-e^{-\frac{T}{\tau}}\right)}{z-e^{-\frac{T}{\tau}}} \end{aligned}$$

對於控制器，使用梯形離散化，我們得到：

$$\begin{aligned} C(z) &= \frac{U(z)}{E(z)} = K_P + K_I\frac{T}{2}\frac{z+1}{z-1} \\ &= \frac{\left(K_P+\frac{TK_I}{2}\right)z+\left(-K_P+\frac{TK_I}{2}\right)}{z-1} \end{aligned}$$

將分子和分母除以 z 並回到時間，我們得到：

$$u(k) = u(k-1) + \left(K_P+\frac{TK_I}{2}\right)e(k) + \left(-K_P+\frac{TK_I}{2}\right)e(k-1)$$

結合執行器的傳遞函數及其負載和其中之一控制器，我們得到以下閉環傳遞函數：

$$F(z) = \frac{K\left(1-e^{-\frac{T}{\tau}}\right)\left(K_P+\frac{TK_I}{2}\right)z + K\left(1-e^{-\frac{T}{\tau}}\right)\left(-K_P+\frac{TK_I}{2}\right)}{z^2+\left(K\left(K_P+\frac{TK_I}{2}\right)\left(1-e^{-\frac{T}{\tau}}\right)-1-e^{-\frac{T}{\tau}}\right)z + K\left(1-e^{-\frac{T}{\tau}}\right)\left(-K_P+\frac{TK_I}{2}\right)+e^{-\frac{T}{\tau}}}$$

現在使用所需的表演，很容易得出結論，極點是

$$s_{1,2} = -\zeta\omega_n \pm jw_n\sqrt{1-\zeta^2}$$

其中ζ和ωn 分別代表阻尼比和固有頻率我們系統控制的閉環。

根據控制理論（參見 Boukas [1]），眾所周知，超調 d%和 5%的建立時間 ts 由下式得出：

$$\begin{aligned} d\% &= 100e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \\ t_s &= \frac{3}{\zeta\omega_n} \end{aligned}$$

$$\zeta = 0.707$$

$$\omega_n = \frac{5}{\tau\zeta} = 110.6387 rad/s$$

$$s_{1,2} = -78.2216 \pm 78.2452j$$

$$z_{1,2} = 0.5317 \pm 0.2910j$$

$$\begin{aligned}\Delta_d &= (z - 0.5317 - 0.2910j)(z - 0.5317 + 0.2910j) \\ &= z^2 - 1.0634z + 0.3674\end{aligned}$$

$$1 + C(z)G(z) = \Delta_d$$

$$K_I = \frac{0.3040}{KT\left(1 - e^{-\frac{T}{\tau}}\right)}$$

$$K_P = \frac{-0.4308 + 2e^{-\frac{T}{\tau}}}{2K\left(1 - e^{-\frac{T}{\tau}}\right)}$$

備註 11.2.1 在這種情況下必須謹慎，因為我們不在乎傳遞函數零的位置，因此我們可能會有一些實現此控制器時會感到驚訝。 顯然，我們的表演將獲得（建立時間和超調量）取決於零的位置。有關此問題的更多詳細信息，請向讀者介紹 Boukas [1]。

現在要實施此 PI 控制算法並確保所需的性能我們將使用 Microship1 的微控制器。 這個選擇是由於我們的經驗使用這種類型的微控制器。 讀者可以記住，任何其他微型其他製造商的控制器稍作改動即可完成工作。 在這個例如，我們將使用 Microhip 的 dsPIC30F4011 單片機。我們實現的代碼使用 C 語言編寫。

這種語言是為簡單起見而採用。 該實現具有以下結構：

```
//
// Put here the include
//

#include "p30F4011.h"        // proc specific header

//
```

[1] See www.microchip.com

```
// Define a struct
//
 typedef struct {
      // PI Gains
      float K_P;       // Propotional gain
      float K_I;        // Integral gain
    //
    // PI Constants
    //
    float Const1_pid;  // KP + T KI/2
    float Const2_pid;  // -KP + T KI/2
    float Reference;   // speed reference

    //
    // System variables
    //
    float y_k;    // y_m[k] -> measured output at time k
    float u_k;    // u[k]   -> output at time k
    float e_k;    // e[k]   -> error at time k

    //
    // System past variables
    //
    float u_prec;      // u[k-1] -> output at time k-1
    float e_prec;      // e[k-1] -> error at time k-1

  }PIStruct;

  PIStruct thePI;

  thePI.Const1= thePI.K_P+T*thePI.K_I/2;
  thePI.Const2=-thePI.K_P+T*thePI.K_I/2;
  thePI.Reference=600;


//
// Functions
//
float ReadSpeed(void);

float ComputeControl(void);

float SendControl(void);

//
// Interrupt program here using Timer 1 (overflow of counter Timer 1)
//
void __ISR _T1Interrupt(void)     // interrupt routine code
  {
```

```
        // Interrupt Service Routine code goes here
        float Position_error;

        //
        // Read speed
        //
         thePI.y_m=ReadSpeed();

        thePI.e_k= thePI.Reference-thePI.y_m;

        //
        // Compute the control
        //
        ComputeContrl();

        //
        // Send control
        //
        SendControl();

        IFS0bits.T1IF=0;    // Disable the interrupt
  }


int main ( void )           // start of main application code
{
// Application code goes here
int i;

// Initialize the variables Reference and ThePID.y_m (it can be read
 from inputs) Reference =  0x8000;  // Hexadecimal number
(0b...  Binary number) ThePID = 0x8000;

// Initialize the registers
TRISC=0x9fff; // RC13 and RC14 (pins 15 and 16) are configured as outputs
IEC0bits.T1IE=1;   // Enable the interrupt on Timer 1

// Indefinite loop
while (1)
{
}
return 0
}


      % ReadSpeed function
      int ReadSpeed (void)
           {
           }
```

```
% ComputeControl function
int ComputeControl (void)
{
thePI.u_k=thePI.u_prec+thePI.Const1*thePI.e_k+thePI.Const2*thePI.e_prec;
}

% SendControl function
int Send Control (void)
{
sendControl()

//
// Update past data
//
thePI.u_prec=thePI.u_k;
ThePI.e_prec=thePI.e_k;
}
```

從該結構可以看出，首先我們注意到系統將進入循環，並在每次中斷時調用函數：

- ReadSpeed；
- ComputeControl；
- SendControl；

並採取適當的措施。

ReadSpeed 函數將在每個採樣時間返回加載速度，由 ComputeControl 函數使用。

SendControl 函數發送通過 L293D 芯片向執行器提供適當的電壓。使用編譯器 HighTec C 獲取十六進制代碼，並使用 PicKit-2 上傳該文件在微控制器的存儲器中。

有關如何獲取十六進制的更多詳細信息代碼，我們邀請讀者閱讀編譯器 HighTec C 或編譯器的手冊 Microchip 的 C30。在這種情況下，國家方法是微不足道的，我們將不發展它。

11.3 直流電機套件的位置控制

讓我們專注於負載位置控制。遵循與負載矢量類似的步驟上一節中開發的位置控制，我們首先需要選擇所需的我們希望系統具有的性能。以下表演是施加：

・系統穩定在閉環狀態；
・穩定時間 ts 為 2%，等於我們可以擁有的最佳時間
・超調等於 5%
・步進功能作為輸入的穩態等於零
使用性能和傳遞函數，很容易得出結論：
比例控制器 KP 足以滿足這些性能。
在此示例中，我們將使用連續時間方法來設計
控制器。 在上一章的基礎上，我們的系統模型如下：

$$G(s) = \frac{K}{s(\tau s + 1)}$$

where K and τ take the same values as for the speed control.

Let the transfer controller be give by:

$$C(s) = K_P$$

Using these expression, the closed-loop transfer function is given by:

$$\begin{aligned} F(s) &= \frac{C(s)G(s)}{1 + C(s)G(s)} \\ &= \frac{\frac{KK_P}{\tau}}{s^2 + \frac{1}{\tau}s + \frac{KK_P}{\tau}} \end{aligned}$$

Since the system is of type 1, it results that the error to a step function as input is equal to zero with a proportional controller.

Based on the specifications, the following complex poles:

$$s_{1,2} = -\zeta w_n \pm j w_n \sqrt{1 - \zeta^2}$$

will do the job and the corresponding characteristic equation is given by:

$$s^2 + 2\zeta w_n s + w_n^2 = 0.$$

Equating this with the one of the closed-loop system we get:

$$\begin{aligned} 2\zeta w_n &= \frac{1}{\tau} \\ w_n^2 &= \frac{KK_P}{\tau}. \end{aligned}$$

To determine the best settling time t_s at 2 %, notice that we have:

$$t_s = \frac{4}{\zeta w_n}.$$

Using now the fact:

$$\zeta w_n = \frac{1}{2\tau}$$

因此，此控制器在%2 處的最佳建立時間為 8 倍系統的恆定時間。 小於可獲得的任何值。 其實這是如果我們在改變 KP 時關注閉環系統的根源，那麼這是微不足道的。 這個由圖 11.3 給出。 為了固定控制器的增益，所需極點 s1,2 = -7.5±j，我們使用該圖並選擇 ζ = 0.707。 得出 KP = 0.1471。

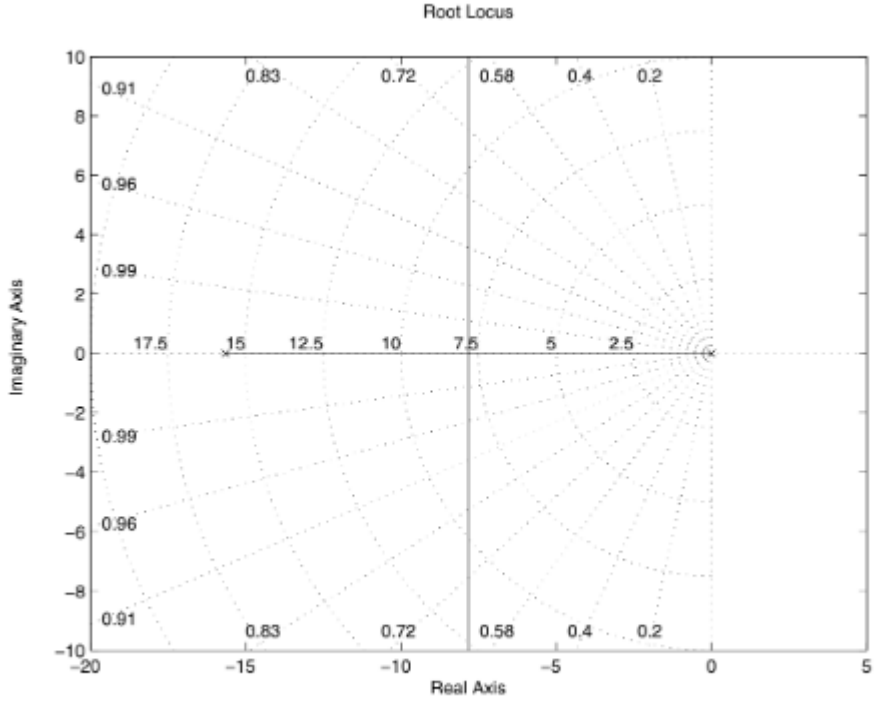

Fig. 11.3 Root locus of the dc motor with a proportional controller

使用該控制器，幅度為的階躍函數的時間響應等於 30 度由圖 11.4 表示，由此我們可以得出結論：控制器以%2 的穩定時間滿足所有期望的性能至 0.5115 s。 但是，如果我們實現此控制器，實際情況將與由於變速箱的齒隙不包括在使用的模型中因此，實時結果將有所不同，並且誤差永遠不會為零。 至為了克服這個問題，我們可以使用比例和微分控制器在%2 處提供更好的建立時間。給出該控制器的傳遞函數通過：

$$C(s) = K_P + K_D s$$

其中 KP 和 KD 是要確定的增益。

備註 11.3.1 重要的是要注意，使用比例和衍生控制器將在閉環傳遞中引入零，這可能會改善放置時間是否合適。 根據其位置，超調和穩定時間將受到影響。 有關此問題的更多詳細信息，請參閱讀者至[1]。

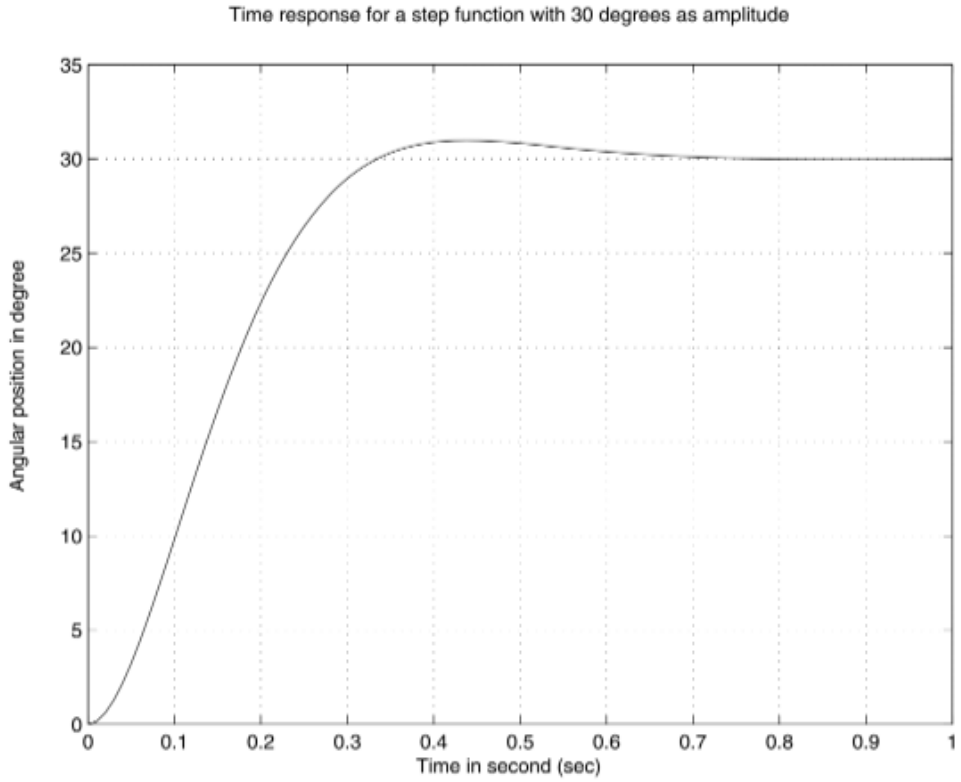

Fig. 11.4 Time response for a step function with 30 degrees as amplitude

With this controller the closed-loop transfer function is given by:

$$\begin{aligned} G(s) &= \frac{C(s)G(s)}{1 + C(s)G(s)} \\ &= \frac{\frac{KK_Ds + KK_P}{\tau}}{s^2 + \frac{1+KK_D}{\tau}s + \frac{KK_P}{\tau}} \end{aligned}$$

和以前一樣，控制器的設計使用了兩個複雜的極點。 要是我們將得到的兩個特徵方程式等價：

$$\begin{aligned} 2\zeta w_n &= \frac{1 + KK_D}{\tau} \\ w_n^2 &= \frac{KK_P}{\tau}. \end{aligned}$$

在這種情況下，我們有兩個未知變量 KP 和 KD 以及兩個代數唯一確定增益的方程。它們的表達式如下：

$$K_P = \frac{\tau w_n^2}{K}$$
$$K_D = \frac{2\tau\zeta w_n}{K}$$

現在使用期望的性能，我們得出與之前類似的結論。輸入的穩態誤差等於幅度等於 30 度的階躍函數實例等於零，並且阻尼比 ζ 對應於過衝等於%5 等於 0.707。穩定時間，ts 為%2，我們可以將其固定為系統時間常數的比例，得出：

$$w_n = \frac{4}{\zeta t_s}.$$

現在，如果將穩定時間固定為 3τ，我們將得到：

$$w_n = 29.4985.$$

使用這些值，我們可以得到以下控制器增益值：

$$K_P = 1.1374$$
$$K_D = 0.0545.$$

它給出了以下複雜的兩極：

零為：

$$z = -20.8618.$$

使用該控制器，輸入幅度等於的時間響應在圖 11.5 中表示了 30 度。從該圖可以看出超調和建立時間少於使用比例獲得的控制器。實施比例或比例和微分控制我們需要獲得控制律的遞推方程。以此目的，我們需要使用不同的方法離散化控制器的傳遞函數 ods 較早提出。讓我們使用梯形方法替換 s 乘 2

$$C(z) = K_p \text{ for the proportional controller}$$
$$C(z) = K_P + K_D \frac{2}{T}\frac{z-1}{z+1} \text{ for the proportional and derivative controller}$$

z + 1 用於比例和微分控制器如果我們通過控件分別表示 u（k）和 e（k）以及引用之間的誤差以及瞬時 kT 的輸出，我們得到以下表達式：

1. 比例

$$u(k) = K_P e(k)$$

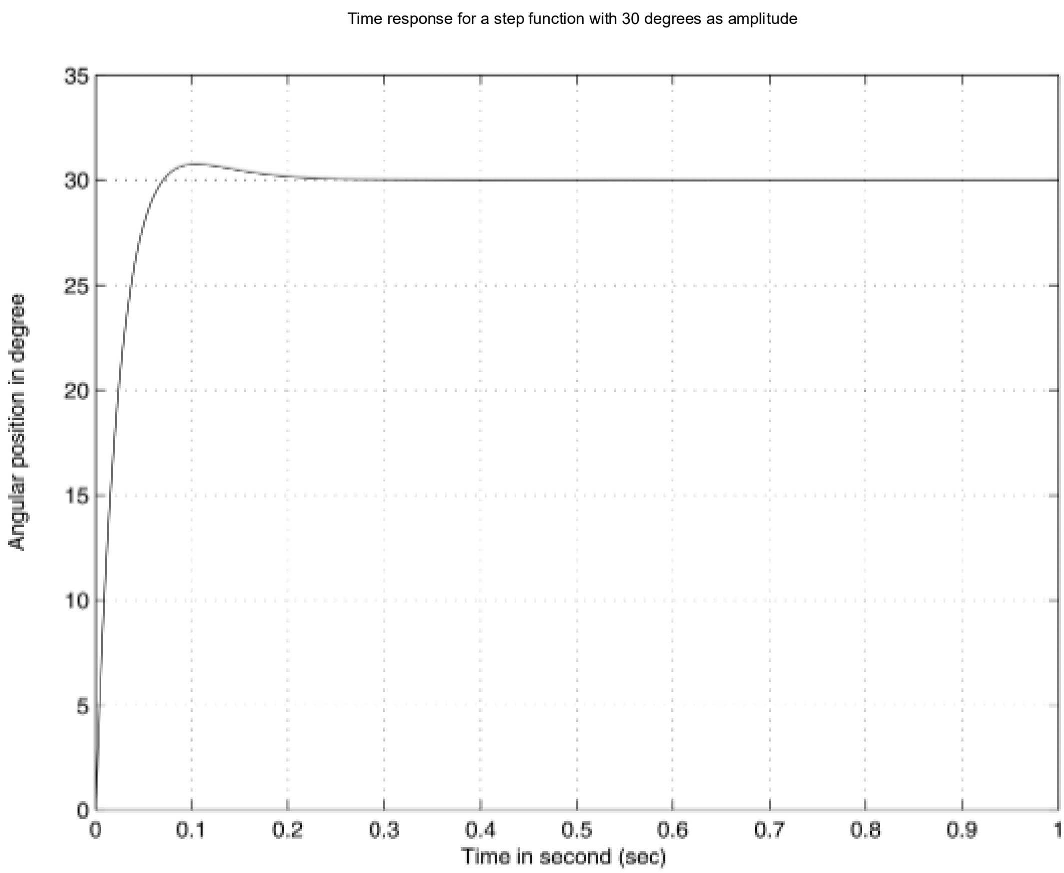

Fig. 11.5 Time response for a step function with 30 degrees as amplitude

2. for the proportional and derivative controller

$$u(k) = -u(k-1) + K_P + \frac{2K_D}{T} e(k) + K_P - \frac{2K_D}{T} e(k-1)$$

The implementation is this controller uses the same function with some minors changes. The listing the corresponding functions is:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Main program                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

main
% Data

% Variables

% While loop while (1) do
        ReadSpeed;
```

```
        ComputeControl;
        SendControl; end;
% ReadSpeed function
% ComputeControl function
% SendControl function
```

現在讓我們使用此示例的狀態空間表示形式並設計一個狀態反饋控制器，將保證所需的性能。 對於這種情況，我們首先將假定完全進入各州，其次，我們將其放寬為-假設我們只能訪問該職位。 就像我們之前所做的我們可以在連續時間或離散時間進行。

先前我們建立了該系統的狀態空間描述，並給出了通過：

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{\tau} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{K}{\tau} \end{bmatrix} u(t)$$

其中 x（t）∈R2（x1（t）=θ（t）和 x2（t）=θ˙（t）），u（t）∈R（施加電壓）。從穩定時間為%2 等於 3τ 的期望性能中，我們得到與以前相同的主導極點，因此特徵方程相同，Δd（s）= s2 +2ζwns + w2n = 0（ζ= 0.707 和 wn = 43ζ τ）。使用控制器表示閉環特徵方程式：

$$\det(sI - A + BK) = 0$$

通過均衡這兩個方程式，我們可以得到以下收益：

$$K_1 = 1.1146$$

$$K_2 = 0.0326$$

使用該控制器，輸入幅度等於 30 度表示在圖 11.6 中。從該圖可以看出超調和建立時間就是我們想要的。這很重要注意在穩態狀態下錯誤的存在。這個錯誤可以是如果在循環中添加一個積分動作，則消除。有關此的更多詳細信息，請參閱讀者閱讀[1]。

對於第二種情況，由於無法訪問加載速度，我們可以從該位置計算它或使用觀察者估計系統狀態。因為它前面說過，我們用於觀察者設計的極點應該更快比控制器設計中使用的那些

選擇以下極點（s1,2，為設計中實際部分的四倍）控制器）：

$$s_{1,2} = -4\zeta w_n \pm j w_n \sqrt{1 - \zeta^2}$$

$$= -83.4218 \pm 20.8618j$$

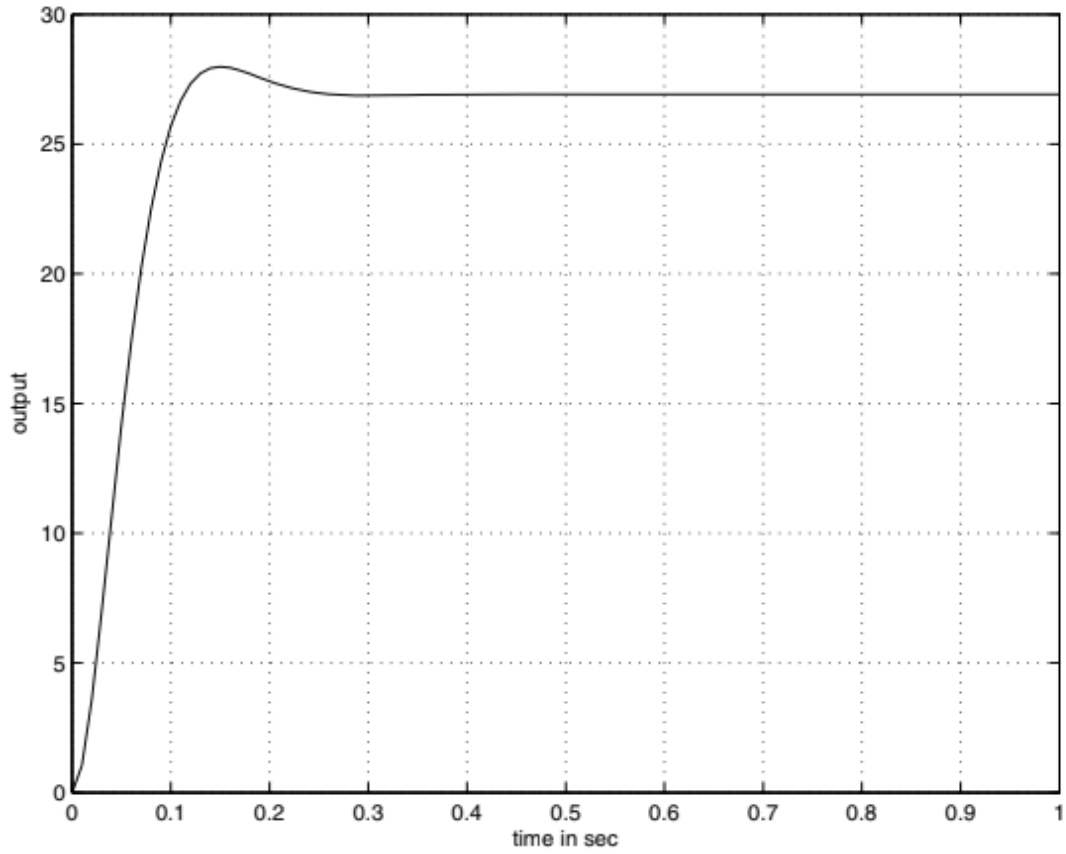

Fig. 11.6 Time response for a step function with 30 degrees as amplitude

我們為觀察者帶來以下收益：

$$L_1 = 151.2$$

$$L_2 = 5029.4$$

控制器的增益與完全訪問控制器的情況相同。狀態向量。

在下面的 Matlab 中，我們提供控制器和觀察器的設計同時給出模擬，以顯示系統和觀察者。

```
clear all

%data tau=0.064 k=48.9
A = [0 1;0 -1/tau];
B = [0 ; k/tau];
C = [1 0];
D = 0;

% controller design
K = acker(A,B,[-3+3*j -3-3*j]);
L = acker(A',C',[-12+3*j -12-3*j])';
```

```
% Simualation data Ts = 0.01; x0 = [1 ; 1]; z0 =
[1.1 ; 0.9]; Tf = 2;     %final time

%augmented system
Ah = [A                    -B*K;
          L*C          A-B*K-L*C];
Bh = zeros(size(Ah,1),1); Ch = [C D*K];
Dh = zeros(size(Ch,1),1); xh0 = [x0 ; z0];
t=0:Ts:Tf; u = zeros(size(t)); m =
ss(Ah,Bh,Ch,Dh);

%simulation
[y,t,x] = lsim(m,u,t,xh0);

%plotting figure; plot(t,y); title('Output');
xlabel('Time in sec') ylabel('Output') grid

figure; plot(t,x(:,1:size(A,1))); title('States of the system');
xlabel('Time in sec') ylabel('System states') grid

figure; plot(t,x(:,size(A,1)+1:end)); title('states of the
observer'); xlabel('Time in sec') ylabel('Observer states') grid
```

Figs （11.7）-（11.9）給出了輸出，系統狀態和觀察者狀態的說明。

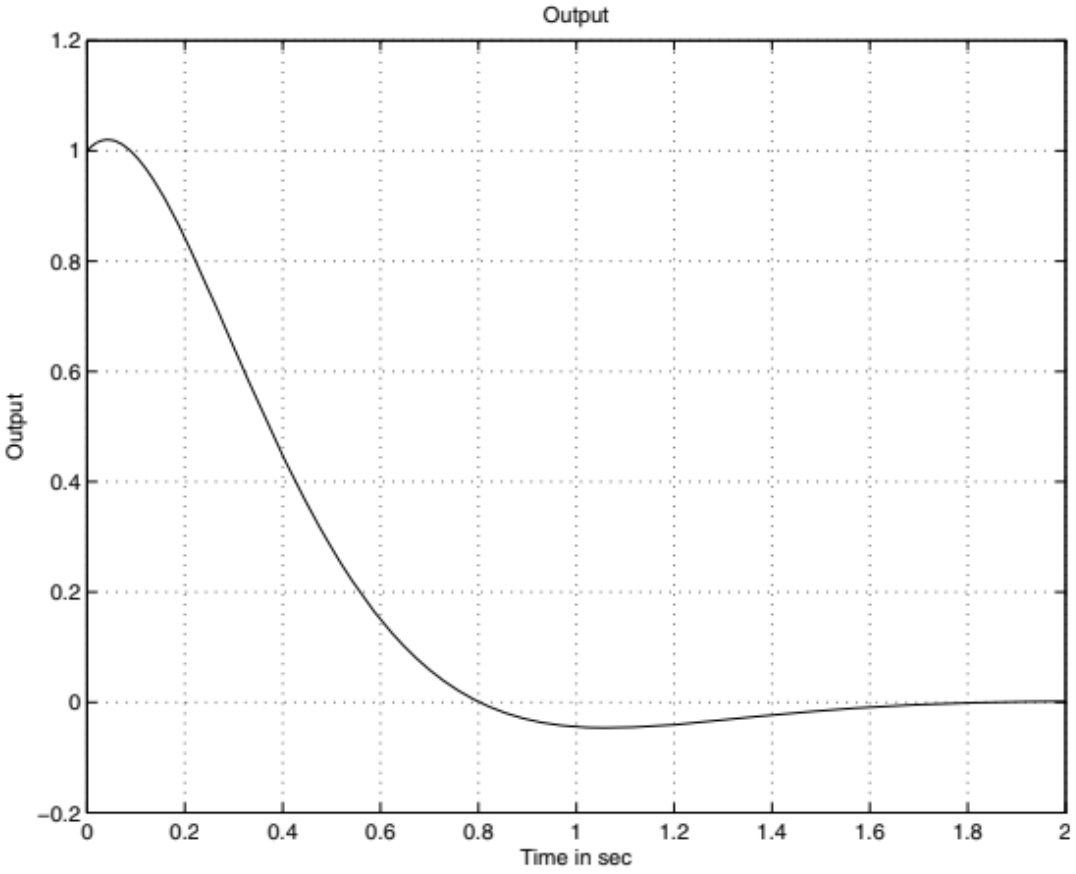

Fig. 11.7 Output versus time

我們還可以使用線性二次方程式設計狀態反饋控制器調節器。 實際上，如果我們為成本函數選擇以下矩陣：

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$$
$$R = 10$$

備註 11.3.2 通常，選擇矩陣沒有魔術規則成本函數。 但總的來說，我們使用較高的值來控制實例將強制控件採用較小的值，並可能防止飽和。
使用這些矩陣和 Matlab 函數 lqr，我們得到：

$$K = 0.3162\ 0.6875.$$

我們還可以使用魯棒控制的結果設計狀態反饋控制器部分。 由於系統沒有不確定性，也沒有外部干擾，我們可以設計一個用於名義動態的狀態反饋控制器。
使用系統數據和 Maltlab，我們得到：

$$X = \begin{bmatrix} 1.1358 & -0.3758 \\ -0.3758 & 1.1465 \end{bmatrix}$$
$$Y = \begin{bmatrix} -0.0092\ 0.0228 \end{bmatrix}$$

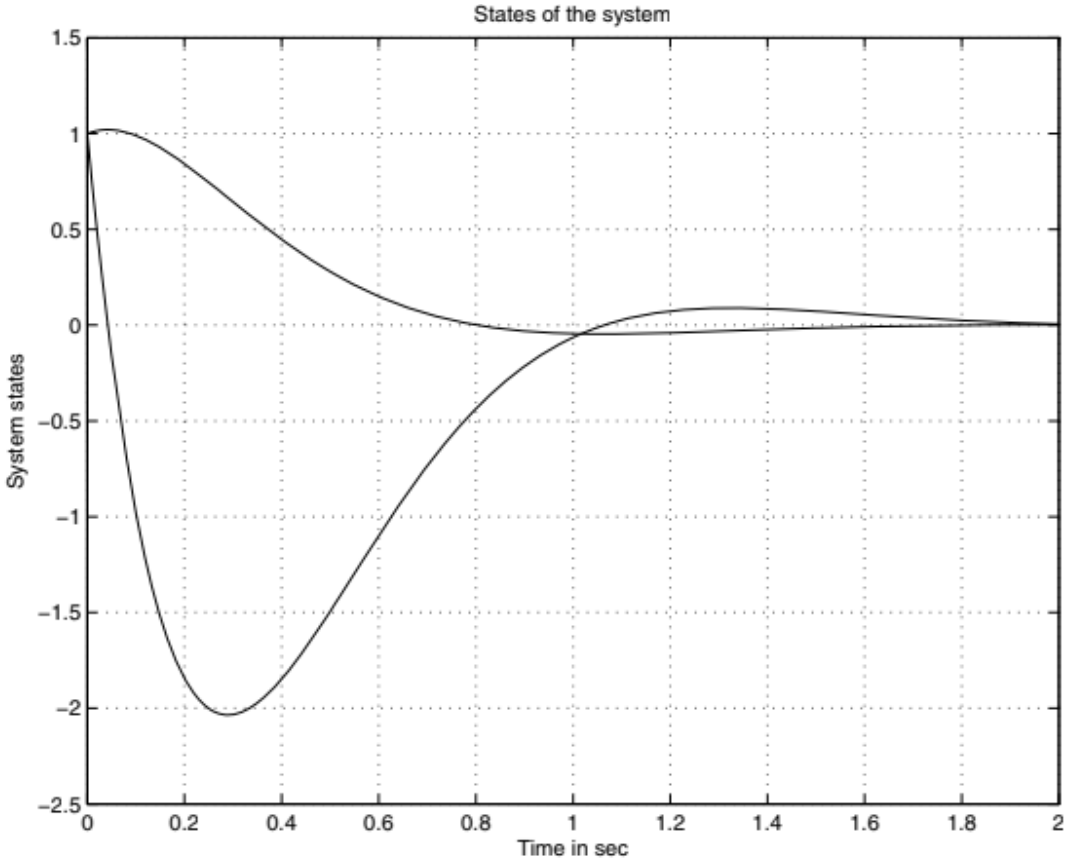

Fig. 11.8 System's states versus time

給出了相應的控制器增益：

K = −0.0017 0.0193

備註 11.3.3 由於我們有直流電動機套件的連續時間模型，因此我們用它來設計控制器增益。 在這種情況下，我們解決了以下問題

LMI：

$$AX + XA + BY + YB < 0$$

增益 K 由下式得出：K = YX-1。

有關連續時間情況的更多詳細信息，請向讀者介紹 Boukas [2]以及其中的參考文獻。

心得:

由於我們設計系的學生大多是純機械科或製圖系進來的，大多沒有機電相關的知識，但是如果要設計機台，就必須要有機電相關的認知，這篇文章能夠教我們如果使用機電系統。