CHAPTER 9

CONTROL FUNCTION

9.1. INTRODUCTION

Models of greenhouses take various forms other than just greenhouses with single or double covering layers. Modelling of greenhouses has been studied extensively (Takakura, 1989). Static models are still predominant, and they are powerful for analyzing particular problems such as comparison of the effectiveness of various thermal screens and estimation of total heat requirements based on actual measurements.

In order to investigate the effectiveness of reflective blinds and the Fresnel prism effect of coverings, light penetration models have been developed.

Several models of total and dynamic systems have been developed. New innovations such as expert systems and practical feed-forward control systems will be included in the models, although there are several problems to be solved such as determination of heat transfer coefficients and sky temperature.

Greenhouse models consist of four sub-systems: 1) light penetration, 2) heat and mass transfer, 3) control function and 4) plant growth. The models are classified as either static or dynamic and as either total- or sub-systems (see Fig. 9.1). Static analyses using sub-systems are predominant for analyzing particular problems such as comparing the effectiveness of various thermal screens. In order to estimate overall heat requirements based on measurements in practical greenhouses, static analyses using total systems have been used. The light environment is one of the most important sub-systems for plant growth, and can be treated separately. Therefore, the light environment has been studied independently, and its models can be either static or dynamic. This sub-system can be included in the total system, of course.

Total and dynamic system models are defined as models that can predict inside environmental factors in the dynamic sense.

The sub-model for plant growth is one of the most difficult to include in the total system, but it is essential. Many primitive models have been reported. Rather sophisticated plant growth models with expert systems are involved in a new area of study although some difficulties have been noted.

Protected cultivation is now being investigated in two main ways. In one, we looked at environmental control using films in static ways; the other involves dynamic mechanical systems such as heaters and coolers. Where the outside climate is mild, most greenhouses are not heated. Even in Japan, more than 65% of the total covered cultivation area is not heated. In Mediterranean countries, there are a large number of unheated greenhouses. On the other hand, in northern Europe, greenhouses with heating systems are essential.

Several control functions are necessary for a heating system, and they must be considered as a unit. For example, in most cases air temperature and humidity

interact with each other. It is possible to change air temperature without changing absolute humidity, but relative humidity changes as a result. A change in inside air temperature by ventilation changes the CO_2 concentration in the greenhouse. Therefore, it is rather difficult to control one factor without affecting others.

However, if we focus on control machines such as heaters or coolers and the main effect of these machines is on one environmental factor such as air temperature, the situation is simplified and can be a good example of actual and complicated control functions. In the present chapter, the relationship between one control machine and one environmental factor is considered for two typical control logics; one is very popular and well-known feedback, and the other is feedforward.

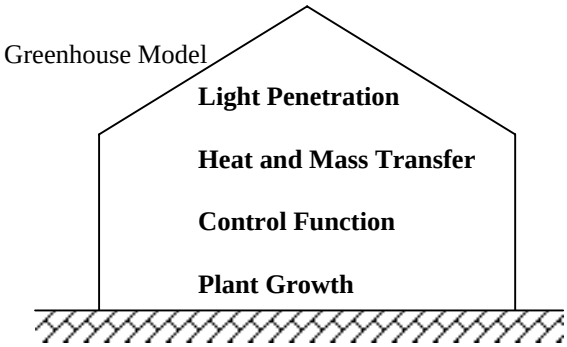

Figure 9.1. Greenhouse sub-models.

9.2. SYSTEM RESPONSE

Let us consider a simple example. Suppose there is a container of water boiling and we put an egg into it; how will the egg's temperature change? It will rise from the room temperature to the boiling temperature. In fact, the temperature distribution varies from one part of the egg to another, and phase change occurs. But in the present example, it is assumed that the temperature distribution is uniform and there is no phase change.

If we generalize the temperature change of the egg, input it to the control system, and assume the temperature surrounding the egg changes from 0 to 1, then the response curve of the system (egg temperature) is similar to that shown by the solid line in Fig. 9.2(a). This is called the system response with first-order delay. The basic equation of the system response with first-order delay to the unit step input is given as

$$\mathbf{RESP} = 1 - \mathbf{exp}(- \mathbf{t} / \mathbf{DELA}) \tag{9.1}$$

This is a solution to the following differential equation:

$$\mathbf{DELA}*d\mathbf{y}/d\mathbf{t} + \mathbf{y(t)} = \mathbf{EE(t)} \tag{9.2}$$

with a constant, that is **EE(t)** = 1.

If the right-hand side of the equation is in the general form, **X(t)**, then the general solution to eq. 9.2 is

$$\mathbf{y(t)} = \mathbf{exp}(- \mathbf{t} / \mathbf{DELA}) / \mathbf{DELA} * (\int \mathbf{exp}(\mathbf{t} / \mathbf{DELA}) * \mathbf{X(t)}\, d\mathbf{t} + \mathbf{C}) \tag{9.3}$$

If the heat content of the egg is large, its temperature does not rise immediately. A system response of this kind is called a system response with second or higher-order delay. The shape of this type of response is indicated by the chain curve in Fig. 9.2(a). These response curves are simplified and approximated as the response with first-order delay and dead time, as shown by the dotted line in Fig. 9.2(a). This can be interpreted as the response to a unit step change **RESD(t)** which is shown in Fig. 9.2(b).

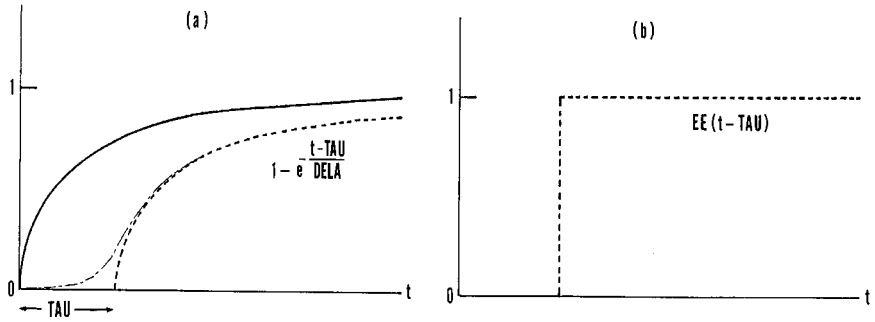

Figure 9.2. Response curve of the system.

This curve is expressed by **RESF(t)** as

$$\mathbf{RESF(t)} = (1.0 - \mathbf{exp}(- (\mathbf{t} - \mathbf{TAU}) / \mathbf{DELA})) \tag{9.4}$$

and

$$\mathbf{RESD(t)} = \mathbf{EE(t} - \mathbf{TAU}) \tag{9.5}$$

where t is time (min), **TAU** is dead time (min or hour) and **DELA** is delay time (min or hour). The function **EE(t)** is a unit step function which changes from 0 to 1 at **TIME** 0 and remains as 1. Therefore, the function **RESD** is the unit step function, which is delayed by **TAU**.

In **CSMP** on mainframes, a **CSMP** function which is called **REALPL** is available. This is the function used to calculate eq. 9.3. Modelling can be simplified with this function, but it is not available in **micro-CSMP** and **MATLAB**.

9.3. PID CONTROL

The most common type of feedback control logic is PID (Proportional, Integral and Differential) control. The error (**ER**) is defined as the difference between a set point (**SP**) and a controlled value (**CV**):

$$\mathbf{ER} = \mathbf{SP} - \mathbf{CV} \tag{9.6}$$

Therefore, the process variable (**PV**) is given as

$$\mathbf{PV} = \mathbf{KK}*(\mathbf{ER} + \frac{1}{\mathbf{TL}}\int_o^t \mathbf{ER} * d\mathbf{t} + \mathbf{TD} * d\mathbf{ER}/d\mathbf{t}) \tag{9.7}$$

where **KK** is a proportionality constant, **TL** is a time constant for integration and **TD** is a time constant for differentiation.

9.4. TEMPERATURE CONTROL LOGIC **(CUC120)**

Let's consider the air temperature control of a floor-heating greenhouse, which is shown in Fig. 9.3. This system has a large heat mass in the floor, which creates a large time lag in the response. The system response is given by eq. 9.3 for an arbitrary input and by eq. 9.4 for a unit step input. If we use a PID controller to control the system, the feedback control logic is given by eqs. 9.6 and 9.7, where the unit for **CV** and **KK** is ^{o}C and the unit for **TL** and **TD** is hr. In CSMP, a step function is available; that is,

$$\mathbf{Y} = \mathbf{STEP(TAU)} \tag{9.8}$$

which means that **Y** is 0 for **t** less than **TAU,** and **Y** is 1 for t equal to or greater than **TAU** as shown in the following **MATLAB** script.

```
 If   (t < TAU)
      Y=0;
else
      Y=1;
end
```

In eq. 9.7, the derivative term can be approximated using numerical differentiation, *i.e.*, difference of **ER** over difference of **t**. The integral term can also be approximated using summation as shown in Fig. 9.4. **MATLAB** provides a function for integration that will be introduced in the next program. Normally, process variables (such as **PV** in this case) are used for control facilities such as heaters and coolers. Their units are, for example, expressed by voltage and valve positions. In order to simplify the problem, the unit of **PV** in the present case is assumed to be the same as that for the controlled value -- that is, temperature. The

change in **PV** is in turn the input to the system -- that is, **X(t)** in eq. 9.3. Then, delay is introduced by using the source code listed in the % ---[Delay] section as shown in Fig. 9.4.

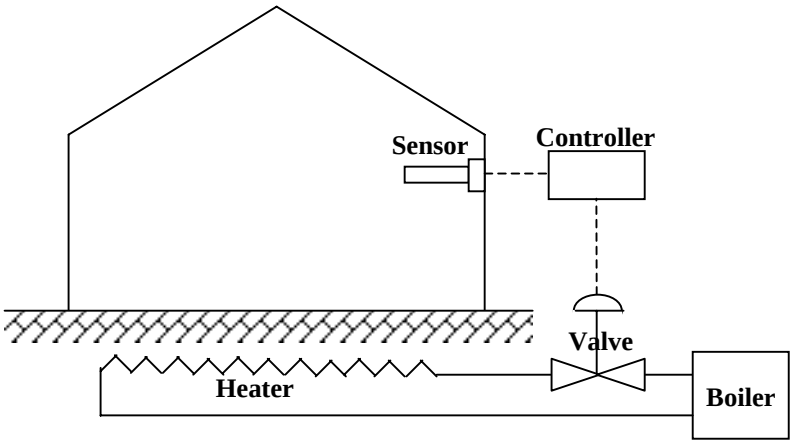

Figure 9.3. A floor-heating greenhouse.

Users can enter 'cuc120', 'cuc120(1)',.., 'cuc120(5)' in the Command Window to run this program. Typical results are shown in Figs. 9.5a, b and c. The response to the unit step change of the set point from 0 to 1^{o}C, which is shown by the line with plus signs, has a 3 hour delay and increases rapidly. In Fig. 9.5a, the response first overshoots the set point and then reaches close to the set point in 48 hours. This means it takes two days to reach steady state after a unit step change. The line with star signs is the error curve, and it is clear that the error decreases rapidly with time. This is not far from reality. It is not difficult to imagine how the situation will be if the input changes periodically. The value **KK** is not only the proportionality constant in PID control logic but also a constant to convert the process variable to the controlled value including the process gain. Various shapes of the response can be obtained by changing the value of **KK** and/or **TAU**. If **KK** is large, overshoot appears (see Fig. 9.5a). Undershoot results from a smaller value of **KK** (see Fig. 9.5b). Smaller oscillations around the set point can be expected after 48 hours, when **TAU** is larger and **KK** is smaller as shown in Fig. 9.5c.

```
% Program for PID control,                                          cuc120.m
function cuc120(pid)
if nargin==0                  % if number of arguments is 0,
  pid=2;                      % case 2 is the default
end
if pid==1                     % case 1: PID control
  Tot_hrs=3600; logic='PID';
  kk = 0.01;tau = 120.0;      % small kk and late response (long delay)
elseif pid==2                 % case 2: PID control
  Tot_hrs=48; logic='PID'; kk=0.1;tau=3.0;% medium kk causing overshoot
elseif pid==3                 % case 3: PID control
  Tot_hrs=48; logic='PID'; kk=0.05;tau=3.0;% small kk causing undershoot
elseif pid==4                 % case 4: PI control
  Tot_hrs=48; logic='PI'; kk = 0.1;tau = 3.0;
elseif pid==5                 % case 5: PD control
  Tot_hrs=48; logic='PD'; kk = 0.1;      tau = 3.0;
end
tl = 0.9;  td = 0.2;  dela = 5.0;
sp = ones(1, Tot_hrs+1);  pv = zeros(1, Tot_hrs+1);
cv=pv;     er=pv;
er(1, 1) = sp(1, 1) - cv(1, 1);   er(1, 2) = sp(1, 2) - cv(1, 2);
total_er = er(1, 1) + er(1, 2);   total_2 = 0;   dt=1;
for t = 2: 1: Tot_hrs;
  switch logic
  case 'PID'                                          %  PID control logic
    pv(1,t+1)=kk*(er(1,t)+td*(er(1,t)-er(1,t-1))/dt+1.0/tl*total_er);
  case 'PI'                                           % PI control
    pv(1, t+1)=kk*(er(1, t)+1.0/tl*total_er);
  case 'PD'                                           % PD control
    pv(1, t+1)=kk*(er(1, t)+td*(er(1, t)-er(1, t-1))/dt);
  case 'P'                                            % P control
    pv(1, t+1)=kk*er(1, t) ;
  case 'I'                                            % I control
    pv(1, t+1)=kk*(1.0 / tl * total_er);
  case 'D'                                            % D control
    pv(1, t+1)=kk*(td*(er(1, t)-er(1, t-1))/dt);
  end
%-------[Delay]-------------------------------------------------------
  if( t < tau )
    cv(1, t+1) = 0;
  else
    tmp = t-tau;      cv(1, t+1) = exp(-tmp/dela) / dela * total_2;
    total_2 = total_2 + exp((tmp)/dela)*pv(1,tmp+1);  % INTGRL term
  end
%---------------------------------------------------------------------
  er(1, t+1) = sp(1, t+1) - cv(1, t+1);
  total_er = total_er + er(1, t+1);      % INTGRL term
end
h1=findobj('tag','cuc120');close(h1);
figure('tag','cuc120','Resize','on','MenuBar','none',...
  'Name','CUC120.m (Figure 1: PID control)',...
  'NumberTitle','off','Position',[160,80,520,420]);
ic = 0:Tot_hrs;
if pid == 1
  plot(ic/24, sp(1, :), ic/24, cv(1, :),'g+-', ic/24, er(1, :),'r*-');
  xlabel('Time elapsed, day');  axis([-inf inf -1 2]);
elseif pid > 1
  plot(ic, sp(1,:), ic, cv(1,:),'g+-', ic, er(1,:),'r*-');
  xlabel('Time elapsed, hr'); axis([-inf inf -0.3 1.5]);
  set(gca,'xtick',[0 6 12 18 24 30 36 42 48],…
```

```
            'ytick',[-0.2 0 0.2 0.4 0.6 0.8 1 1.2]);
end
tit=([logic ' control at kk=' num2str(kk) ', Tau=', num2str(tau)]);
title(tit);ylabel('Relative Temperature, ^oC');
grid on;   legend('Set Point','Response','Error');
```

Figure 9.4. Program to simulate PID control for floor-heating greenhouses **(CUC120.m)**.

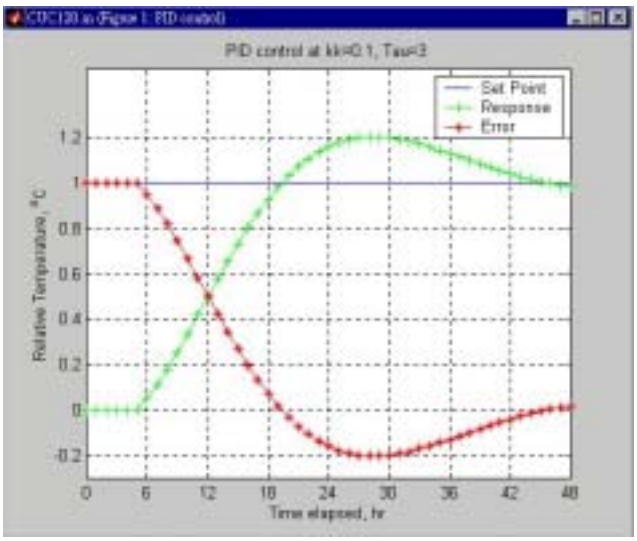

Figure 9.5a. Response and error curves for unit step change (showing overshoot).

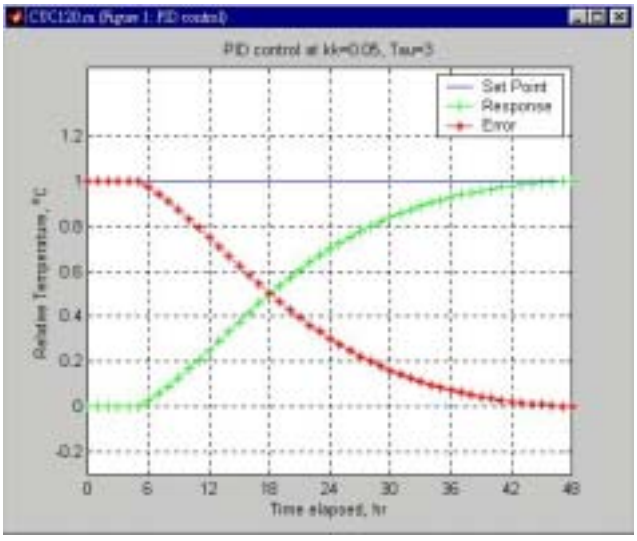

Figure 9.5b. Response and error curves for unit step change (showing undershoot).

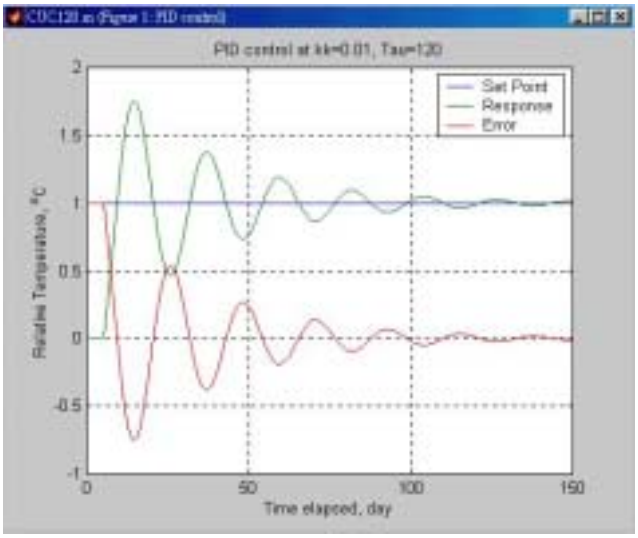

Figure 9.5c. Response and error curves for unit step change with long delay.

9.5. FEEDBACK VS. FEEDFORWARD CONTROL

9.5.1. Feedback and feedforward

Feedback control logic is well-established in both strategy and hardware configuration for many process controls. PID logic is one of the typical feedback control methods and is widely used. On the other hand, feedforward control is usually mentioned briefly in control textbooks, but is rarely found in practical applications. The main reason is that feedforward control depends on some kind of predictive model, even for physical systems, and works well alone in the practical sense. Therefore, the best solution is the combination of feedback and feedforward control if the latter is necessary.

When the system has a large time lag, as with floor heating, feedback techniques cannot effectively minimize the errors caused by the time lag in the system. This situation requires a form of predicted action. In feedforward control, the controller acquires information about potential upsets which have not yet affected the behavior of the process, anticipates the effect of these upsets on the process, and counteracts them before they are manifested on the process. The feedforward technique offers a potentially better solution for a system with large heat mass.

In the present section, it is easily shown that conventional feedback control applied to floor-heated greenhouses results in a large delay from the set point when floor heating starts and a large overshoot when heaters are turned off. Feedforward control introduces a prediction scheme, which will regulate the rate of heat input prior to the expected time and give complete control in this case because the system is perfectly predictable (see Takakura, *et al.*, 1993).

9.5.2. System response and its application to feedforward control

One well-known technique for predicting the dynamic response of a linear system to an arbitrary change of an input function is the weighing function method. In this method, the responses to all inputs are obtained by the principle of superposition. The theory of superimposition can be applied to any linear system.

Using Duhamel's integral, the response **R(t)** to an arbitrary function **F(t)** using the weighing function **W(t)** is as follows:

$$\mathbf{R}(\mathbf{t}) = \int_0^t \mathbf{F}(\mathbf{t} - \tau) * \mathbf{W}(\tau)\, d\tau \tag{9.9}$$

where **W(t)** is the derivative of a unit response.

If the unit response to the floor water temperature is defined as **Rf(t)**, the inside air temperature change **Ti(t)** due to floor water is expressed as

$$\mathbf{Ti}(\mathbf{t}) = \int_0^t \mathbf{Q}(\mathbf{t} - \tau) * \mathbf{Rf}'(\tau)\, d\tau \tag{9.10}$$

where the superscript (') means the first derivative with respect to time. From the Laplace transform of the convolution integral, and by taking the Laplace transform of eq. 9.10, the following equation is given;

$$\mathbf{Q}(\mathbf{s}) = \mathbf{Ti}(\mathbf{s}) / \mathbf{Rf}'(\mathbf{s}) \tag{9.11}$$

9.5.3. A floor-heating greenhouse with ideal conditions **(CUC122)**

The present model is very simplified and hypothetical, but adequate for showing the difference between feedback and feedforward logics. The unit step response of the inside air temperature due to the step change of floor water temperature is expressed by eq. 9.4 with a dead time and delay (**RF(s)** is equal to the Laplace transform of **RESF(t)** in eq. 9.4). In the present model, heat transfer through the cover is assumed instantaneous and no response equation is needed for outside air temperature change. The delay of the control system is also introduced by the %---[DELAY] section as listed in Fig. 9.4. The system is well-defined and all responses are defined by analytical equations.

Now, the set point of inside air temperature **Ti(t)** is set to

$$\mathbf{Ti}(\mathbf{t}) = \mathbf{K33} + \mathbf{A33} * \mathbf{sin}(\mathbf{OMEGA} * \mathbf{t}) \tag{9.12}$$

After taking the Laplace transform of eqs. 9.4 and 9.12, the results are substituted into eq. 9.11. The inverse Laplace transform of eq. 9.11 gives finally the set point of floor water temperature in order to realize the inside air temperature change defined by eq. 9.12.

$$\mathbf{Q(t)} = \mathbf{K33} + \mathbf{A33} * \mathbf{OMEGA} * \mathbf{DELA} * \mathbf{cos}(\mathbf{OMEGA} * (\mathbf{t} + \mathbf{TAU})) + \mathbf{A33} * \mathbf{sin}(\mathbf{OMEGA} * (\mathbf{t} + \mathbf{TAU})) \quad (9.13)$$

The program to compare the two control logics is shown in Fig. 9.6 and a typical result is indicated in Fig. 9.7. Feedback logic only follows the error which is detected at the present time. If the system delay is very large, as in the present case, the feedback system cannot catch up with the error in principle. A one or two hours delay is common in practical large-scale greenhouses. A larger gain in a feedback system can give a quicker response, but it brings larger amplitude difference in the response curve. In the present case, the system response curve is defined without any error, and gives a perfect fit with the set point for feedforward control after a certain time lapse. The %---[Delay] section of the source code listed in Fig. 9.6a provides comparison on feedback (PID) and feedforward control. The approach in deriving **CV** is the same as in **CUC120**, and the approach in deriving **CVF** is more accurate. The **quad8** or **quadl** function is used to do the integration from 0 to **t-TAU**. The function **quad8** was replaced by **quadl** in **MATLAB** from version 6 (Release 12).

```
% Control function for floor-heating greenhouse                CUC122.m
% Comparison of feedback (PID) and feedforward control
function cuc122
global TAU; global  DELA;
KK = 0.1;
TL = 0.9;  TD = 0.5;  TAU = 3.0; DELA = 5.0; % unit of time is hr
totnum=48; totnum2=totnum+1;
SP=zeros(1, totnum2); ER=zeros(1, totnum2); PV=zeros(1, totnum2);
CV=zeros(1, totnum2); CVF=zeros(1, totnum2);
TOTAL_RESP=0;
SP(1,1)=sin(2*pi/24*0);           SP(1,2)=sin(2*pi/24*1);
ER(1, 1) = SP(1, 1) - CV(1, 1);   ER(1, 2) = SP(1, 2) - CV(1, 2);
TOTAL_ER = ER(1,1) + ER(1, 2);
PV(1,2)=KK*(ER(1,1)+TD*(ER(1,1)-0)+1/TL*TOTAL_ER);
for T = 2: 1: totnum;
   CLOCK=mod(T,24);   SP(1,T+1)=sin(2*pi/24*CLOCK);
   PV(1,T+1)=KK*(ER(1,T)+TD*(ER(1,T)-ER(1,T-1))+1/TL*TOTAL_ER);
   % TOTAL_ER is the Integral term of ER
   % PV   = KK*(ER + TD*DERIV(ICD,ER) + 1.0/TL*INTGRL(ICI,ER))
   % Feedback (PID) control logic
%----[Delay]-------------------------------------------------------------
   if T<TAU
      CV(1, T+1) = 0;             CVF(1, T+1) = 0;
   else
      TMP = T-TAU;
      CV(1, T+1) = exp(-TMP/DELA) / DELA * TOTAL_RESP;
      TOTAL_RESP = TOTAL_RESP + exp((TMP)/DELA)*PV(1, TMP+1);
   %  CVF(1,T+1)=exp(-TMP/DELA) / DELA * quad8('feedf',0,TMP);
   %  quad8 is obsolete at MATLAB version 6 (R12)
      CVF(1,T+1)=exp(-TMP/DELA) / DELA * quadl('feedf',0,TMP);
```

```
   % quadl is not available prior to MATLAB version 6 (R12)
   End
%-----------------------------------------------------------------------
   ER(1, T+1) = SP(1, T+1) - CV(1, T+1);
% Error (ER) is the diff. between set point (SP) and controlled value (CV)
   TOTAL_ER = TOTAL_ER + ER(1, T+1);
end
h1=findobj('tag','cuc122');  close(h1);
figure('tag','cuc122','Resize','on','MenuBar','none', 'Name',...
   'CUC122.m (Figure 1: Feedback control vs. Feedforward control)',...
   'NumberTitle','off','Position',[160,80,520,420]);
ic = 0:totnum;
plot(ic, SP(1,:),ic, CV(1,:),'k+-',ic, CVF(1,:),'r*-');
set(gca,'xtick',[0 6 12 18 24 30 36 42 48],'ytick',[-1 -0.5 0 0.5 1]);
axis([-1 totnum2 -inf inf]);
xlabel('Time elapsed, hr');       ylabel('Temperature, ^oC');
legend('Set Point','Feedback','Feedforward');  grid on;
```

Figure 9.6a. A model to demonstrate feedback and feedforward control logics for a floor-heating greenhouse **(CUC122.m).**

```
% Intgrl term of feedforward control                                feedf.m
function y=FEEDF(t)
% This part is a copy of the eqs. 8, 9, 10, and 11 in the paper by
% T. Takakura, et al., Trans. ASAE (Vol. 37, 939-945)
global TAU; global DELA;
  OMEGA=2*pi/24;   A33=1;     K33=0;
  TT=K33+A33*OMEGA*DELA*cos(OMEGA*(t+TAU))+A33*sin(OMEGA*(t+TAU));
  y = exp(t/DELA).*TT;
```

Figure 9.6b. Subprogram of CUC122 **(feedf.m)**.

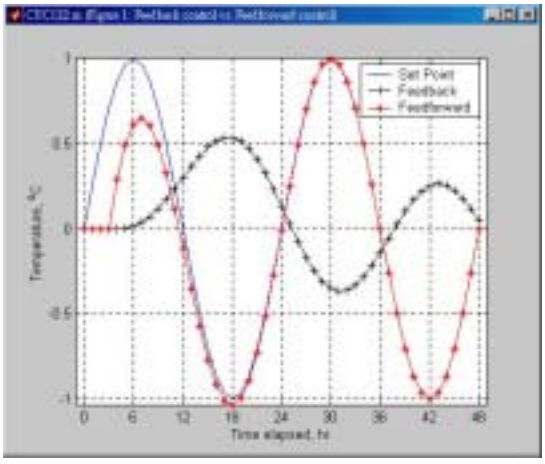

Figure 9.7. Air temperature changes controlled by feedback and feedforward logics with the set point temperature.

MATLAB FUNCTIONS USED

quad8	Numerically evaluate integral, higher order method. Q = QUAD8('F',A,B) approximates the integral of F(X) from A to B to within a relative error of 1e-3 using an adaptive recursive Newton Cotes 8 panel rule. 'F' is a string containing the name of the function. The function must return a vector of output values if given a vector of input values. Q = Inf is returned if an excessive recursion level is reached, indicating a possibly singular integral. Note: Function only available before and in version 5.3 (Release 11), obsolete after version 6.0 (Release 12).
quadl	Numerically evaluate integral, adaptive Lobatto quadrature. Q = QUADL(FUN,A,B) tries to approximate the integral of function FUN from A to B to within an error of 1.e-6 using high order recursive adaptive quadrature. The function Y = FUN(X) should accept a vector argument X and return a vector result Y, the integrand evaluated at each element of X. Note: Function available after version 6.0 (Release 12)

PROBLEMS

1. Verify that eq. 9.3 is the general solution of eq. 9.2.

2. Examine how the system changes if there is only PI and P control logic, modifying the program **CUC120**.

3. Evaluate the effect of **KK** on the system response by re-running the program **CUC120** for several **KK** values.

4. Change the input (**SP**) in the model **CUC120** from the step change to a periodic change and plot the result.

5. Change the gain (**KK**) as well as time constants (**TD**, **TL**) in order to get better fit with the set point and explain the effect of each factor on the accuracy.