Closed-loop auto relay tuning - auto tune variation, ATV

by R. K. Herz <rherz@ucsd.edu> October 2011, last revised November 18, 2012

Auto tuning is an alternative to the Ziegler-Nichols closed-loop ultimate-gain method. In the Z-N method, you use proportional-only control (or as close as can get by setting maximum I and minimum D times), then increase gain until a small change in set point or load results in sustained oscillations. See Riggs p. 301-302 and Chau pp. 112-115. A disadvantage to this method is said to be that it can require a lot of trials.

In the closed-loop auto relay tuning or ATV method, the controller is replaced by a relay, temporarily for the purpose of tuning. The relay is driven by the closed-loop error signal. With no relay hysteresis, the relay switches immediately as the error signal changes sign. The relay has a square-wave output of an amplitude 2*d* (+/- *d*) set by the operator. The resulting system output oscillation amplitude 2*a* (+/- *a*) and period P_u are measured and recorded. Some commercial PID control units have a switch to turn on an auto tuning function.

Doyle shows the following equations to set the PID parameters. Ultimate gain, K_u: $K_u = \frac{4d}{\pi a}$

The operator specifies a phase margin φ_m (e.g., 45°) and a "design parameter" α (e.g., 4) , and then computes the PID parameters

$$K_c = K_u \cos(\varphi_m) \qquad \tau_I = \alpha \tau_D \qquad \tau_D = \frac{\tan(\varphi_m)\sqrt{(4/\alpha) + \tan(\varphi_m)}}{2\omega_c} \qquad \omega_c = \frac{2\pi}{P_u}$$

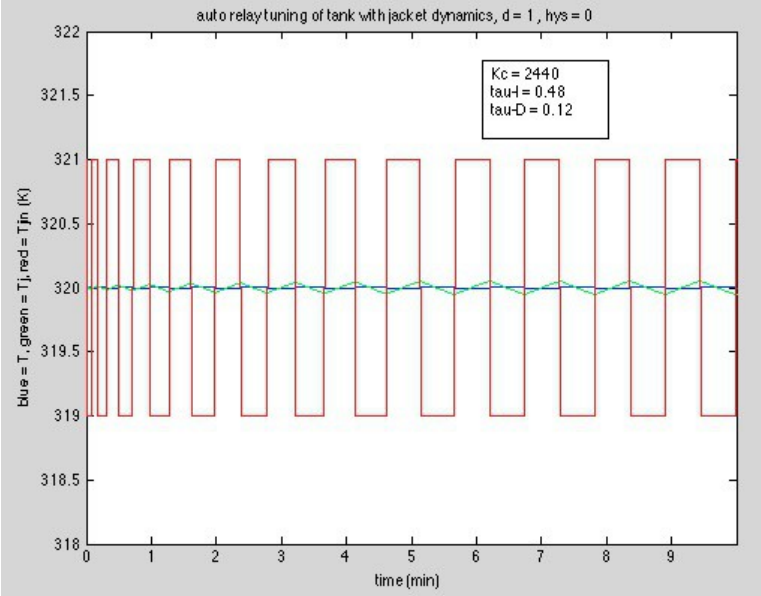
For comparison, here are the tuning parameter relationships for the Ziegler-Nichols ultimate gain method from Antonio Flores T., as also listed by Chau, p. 115 and Riggs and Karim, p. 302.

	K_c	τ_I	τ_D
P	$0.5K_{cu}$		
PI	$0.45K_{cu}$	$P_u/1.2$	
PID	$0.6K_{cu}$	$P_u/2$	$P_u/8$

Consider a mixing tank with a heat transfer jacket where the heat transfer jacket is treated as a well-mixed tank and the manipulated variable is the inlet temperature of the heat transfer fluid. This system is simulated in SimzLab, Control Lab, Division 1, Temperature Control B <www.SimzLab.com>

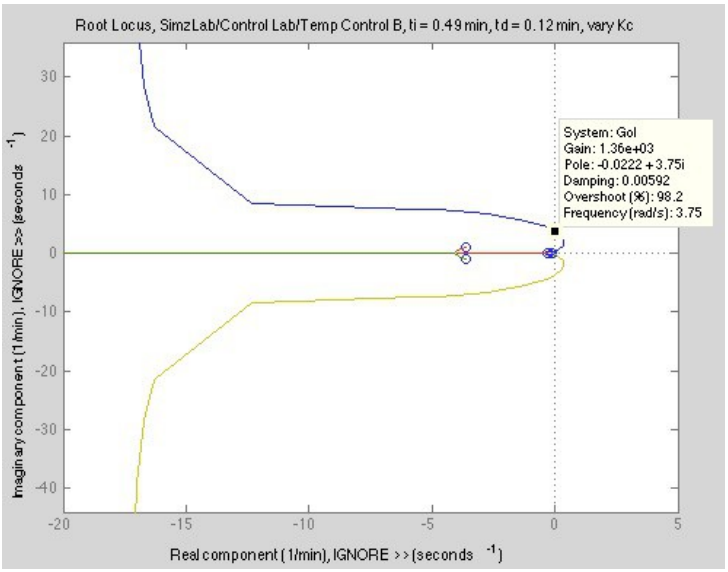
Here, this system is simulated using Matlab. In this Matlab model, we don't have to worry about sensor signal saturation or actuator limits. There is currently no way to conduct the auto relay tuning experiment in SimzLab. The Matlab program is listed at the end of this document.

First let's look at what happens when we run an auto relay tuning experiment with zero hysteresis in the relay, and switching amplitude d = 1 K.



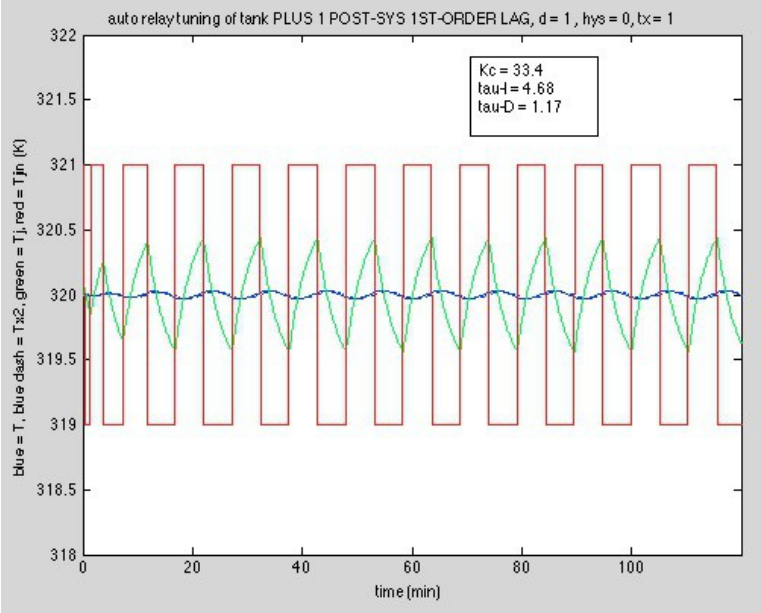
The period of oscillation P_u = 1.07 min and the amplitude a = 0.0004 K. Using Doyle's equations with φ_m = 45° and α = 4, the PID parameters are: K_c = 2440, τ_I = 0.48 min, and τ_D = 0.12 min.

The K_c value is unrealistic. For example, for a step change in set point of 1 K, the proportional mode with this K_c value would demand change in Tj of 2440 K, which is obviously unrealistic. At any Kc value less than about 1360, still an unrealistically high value, the root locus plot shows that the system would be unstable. For these root locus plots, a lead-lag form of the derivative mode was used with coefficient 0.2.



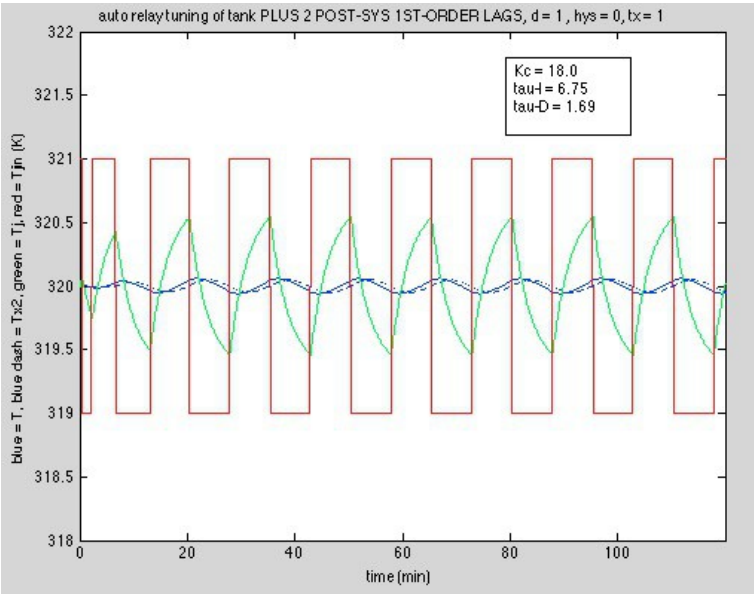
Note that in a real system there would be noise in the signal that would have to be filtered. The filtering time constant added to the system will affect the auto tuning results.

Next, repeat the auto tuning experiment with one first-order lag added before the tank temperature measurement. The tank has a time constant of 10 min and the jacket has a time constant of 3.33 min. The added lag has a time constant of 1 min.



The added lag has a very large effect. The solid blue line is the tank temperature and the dashed blue line is the tank temperature after the added first-order lag (y-axis label should read Tx1 not Tx2). There is not a big difference between the two lines but there is a big change in the oscillation amplitude and frequency (note that time scale changed above from 10 to 120 min). Using the same values of the coefficients in Doyle's equations, the PID parameters are: $K_c = 33.4$, $\tau_I = 4.68$ min, and $\tau_D = 1.17$ min - much more reasonable values.

Next, add a second first-order lag in series with the first. The PID parameters are: $K_c = 18.0$, $\tau_I = 6.75$ min, and $\tau_D = 1.69$ min

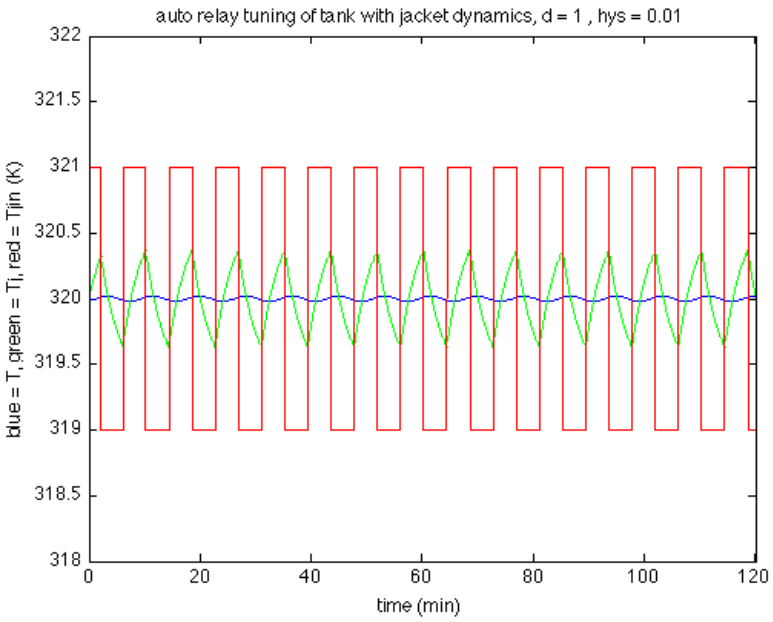


Without the added lags, the tuning parameters that were obtained for this system were unrealistic. Riggs recommends the auto tuning method for "slow-responding" systems.

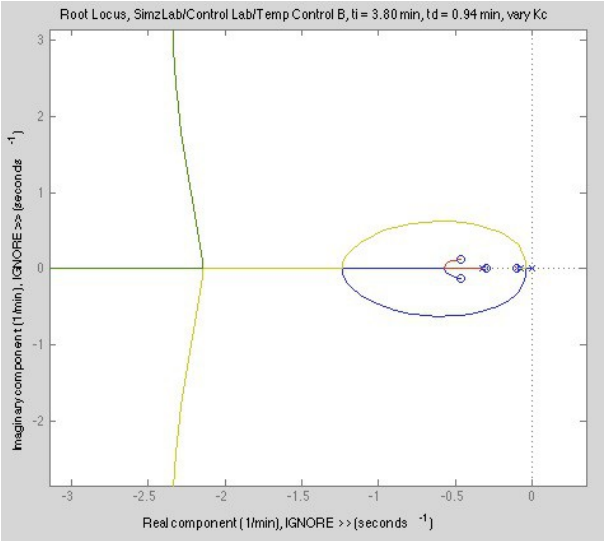
Note that the time units - seconds, minutes, hours - are arbitrary here, which shows that they are not important in determining whether a system is "fast" or "slow" responding. It is the order of the system, the presence of measurement and actuator lags, and the presence of delay times that determines "fast" or "slow" in the context of auto relay tuning.

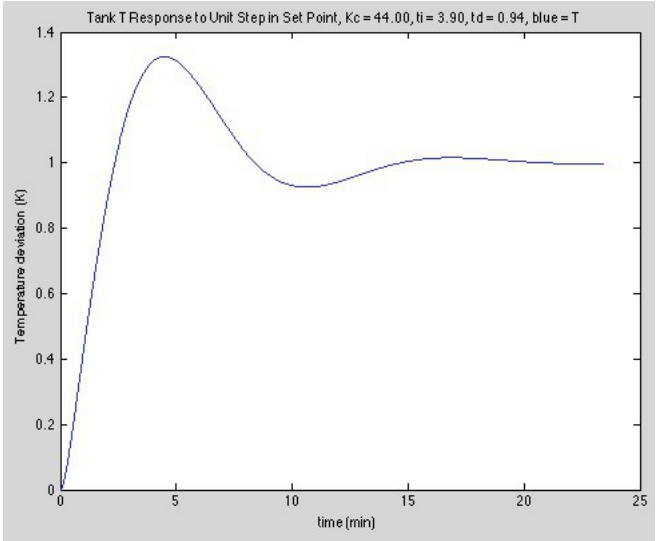
The original paper on auto tuning by Astrom and Hagglund discusses relay hysteresis but many textbooks do not mention it. Let's go back to the original system without the added lags and see the effect of adding hysteresis to the relay. The relay now will switch when the error is greater than the hysteresis value, or less than the negative of the hysteresis value. A real system will have some hysteresis even without hysteresis added to the relay. An example is using a temperature measurement device that reads in increments of 0.1 K.

Here are the results for the original system and a hysteresis value of 0.01 K and $d = 1$ K. The period of oscillation P_u = 8.34 min and the amplitude $a = 0.0204$ K. The PID parameters are: $K_c = 44$, τ_I = 3.8 min, and τ_D = 0.94 min.

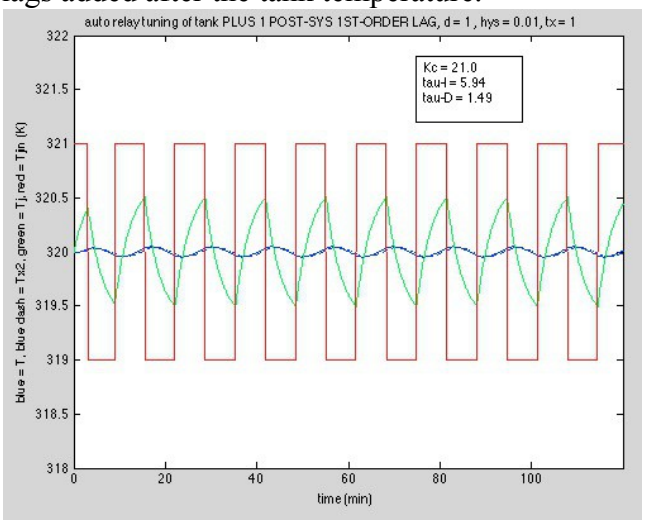


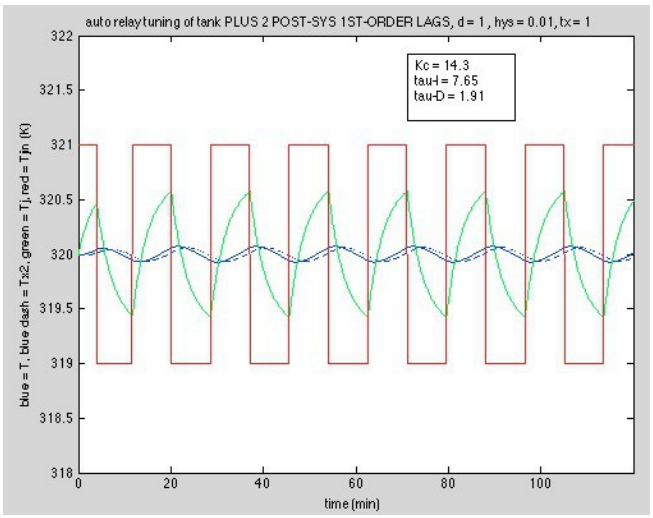
This is a dramatic change in the results. Note the difference in the time scale from the original plot with no hysteresis and no added lags (0-10 min) and the time scale of the plot above. The oscillation period has increased from 1 min to 8 min. The tuning parameter values are getting more realistic, although the initial command value for T_j on a step change in set point is still too high: the initial command for T_j is over 200 K for a 1 K change in set point. The root locus plot below shows that the closed-loop system is stable.



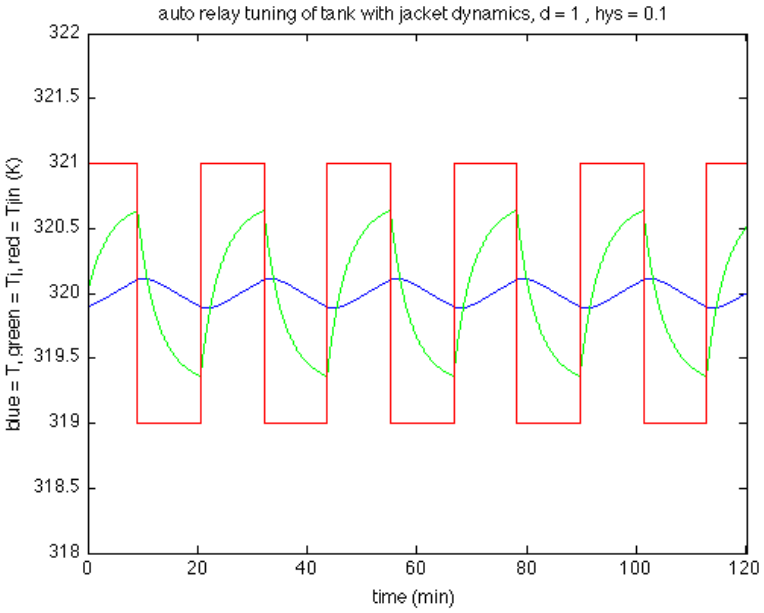


Here are the auto tuning results for a hysteresis value of 0.01 K and one (left) and two (right) first-order lags added after the tank temperature.

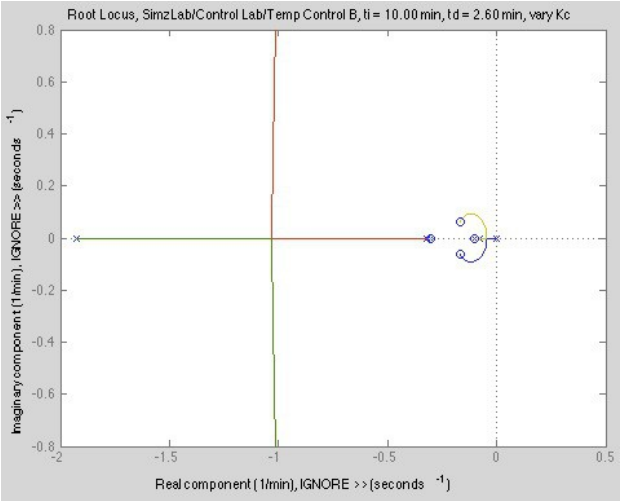


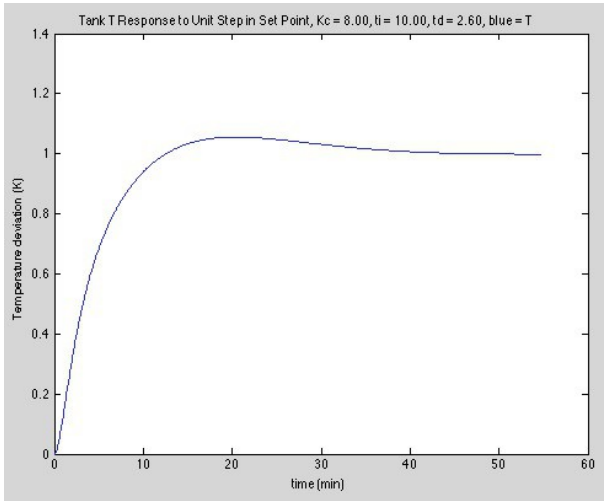


Below are the results for a hysteresis value of 0.1 K and d = 1 K and no added lags. The period of oscillation P_u = 23 min and the amplitude a = 0.11 K. The PID parameters are: K_c = 8.0, τ_I = 10 min, and τ_D = 2.6 min.

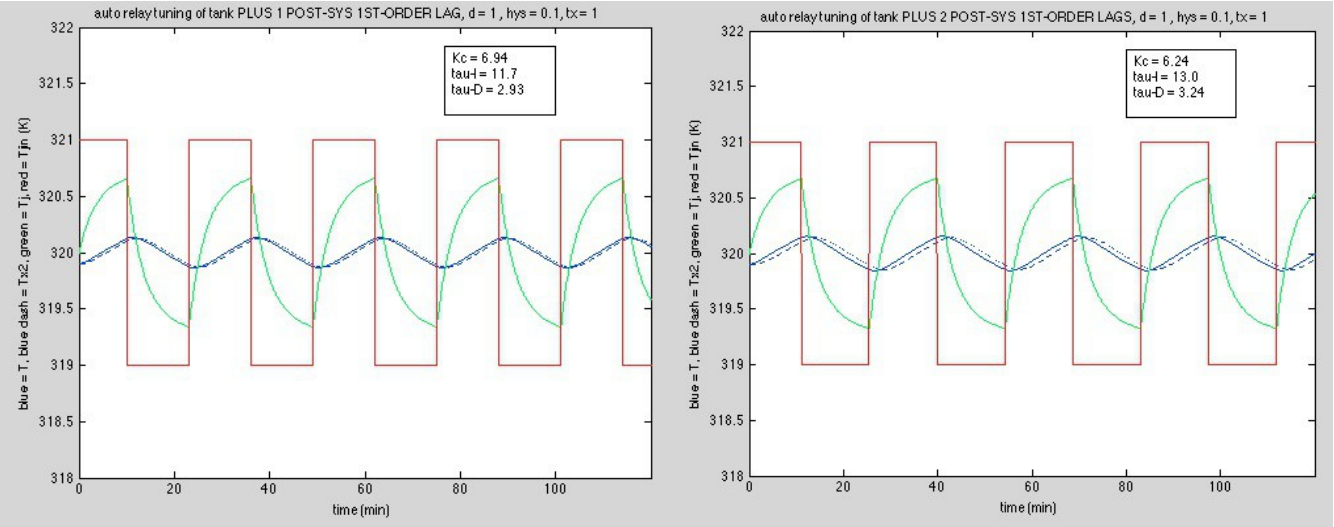


This is now getting to be a reasonable value of K_c and the root locus plot continues to show a stable system. Notice that only a few cycles are required to establish a steady periodic pattern. This can be an advantage of using auto tuning with a slow-responding system vs. trial and error tuning methods.





Here are the auto tuning results for a hysteresis value of 0.1 K and one (left) and two (right) first-order lags added after the tank temperature.



For this inherently linear system, there was no effect of changing the forcing amplitude *d*.

We have found that relay hysteresis and sensor and actuator lag can affect the auto tuning results and, thus, the values of the estimated PID parameters.

References

Astrom, K.J. and Hagglund, T., Automatica, vol. 20, p. 645 (1984).

Antonio Flores T., "Relay Feedback Auto Tuning of PID Controllers," http://www.scribd.com/doc/48001931/Relay-PID-Auto-Tuning

Chau, P. C., *Process Control Modules: A First Course with MATLAB*. Cambridge University Press, 2002.

Doyle, F. J. , *Process Control Modules: A Software Laboratory for Control Design*. Upper Saddle River, N.J: Prentice Hall PTR, 2000.

Riggs, J. B. and Karim, M. N., *Chemical and Bio-Process Control*, 3rd. ed., Lubbock, TX: Ferret Publishing, 2006.

Matlab Program Listing

```
% well-mixed tank with heat transfer jacket
% with jacket dynamics described
%
% here integrate the ODEs in time domain
% and investigate AUTO RELAY TUNING or ATV
%
% R. Herz <herz@ucsd.edu>

fprintf('---------------------- \n') % run separator
clear all

% ------------------------------------------------------------------
%   THESE ARE PARAMS THAT ARE CHANGED TO INVESTIGATE RELAY HYSTERESIS
```

```
% params for auto relay tuning
hys = 0.1; % K, +/- hysteresis in relay
d = 1; % K, +/- in relay change of command TjinC to actuator
% WARNING: d value must be large enough to cause the tank T to change more
%          than hys value or won't get relay switching

% WARNING: make sure you establish steady cycling and
%          have more than one complete cycle in last fraction of data
%          that you look at (lastfrac)
lastfrac = 0.5; % last fraction of results that we will look at

dt = 0.01; % min, time step for integration
tfinal = 120; % min, final time

% ---------------------------------------------------------------------

% parameters that can be changed by user in SimzLab
q = 5e-3; % m3/min, flow rate of process liquid into tank
UA = 10000; % J/min/K, product of heat transfer coefficient U and area A

% parameters that are fixed in SimzLab
V = 0.1; % m3, volume of process liquid in tank
rho = 1000; % kg/m3, density of process liquid in tank
Cp = 2000; % J/kg/K, heat capacity of process liquid in tank

% characteristic times of the open-loop system - units are minutes
% first for the main tank with process fluid
tauf = V/q;
taux = rho*Cp*V/UA;
taup = 1/(1/tauf + 1/taux);

% next for the jacket volume
Vj = 0.5*V; % m3, volume of heat transfer fluid in jacket set in SimzLab
taufj = 5; % min, fluid space time in jacket
tauxj = rho*Cp*Vj/UA;
taupj = 1/(1/taufj + 1/tauxj);

R = 320; % K, set point T
Tin = 320; % K, main tank inlet Tin
Tjin = 320; % K, NOMINAL jacket inlet Tjin, relay switches about this by +/- h
TjinOLD = Tjin; % K, to save current Tjin for relay hysteresis

% dt is set above
% tfinal is set above
T0 = 320 - 1.01*hys; % need some error to get relay to start switching
Tj0 = 320;

% Use Euler's method of numerical integration because
% I had problems using ode45 for integration: spurious bumps in T profiles.
% The problems were with setting TjinC and TjinOLD inside the deriv file
% as ode45 was adjusting its integration parameters and re-taking time
% steps, which fouled up saving earlier TjinC values in TjinOLD.

% initialize variables
t = 0;
T = T0;
Tj = Tj0;
TjinC = 320;

nsteps = tfinal/dt;
for i = 1:nsteps

    % set relay based on error
    % relay sets the jacket inlet TjinC value
    % which is the TjinC(OMMAND), since Tjin here used as fixed reference value
```

```
    e = R - T(i);
    if e >= hys
        TjinC(i) = Tjin + d;
    elseif e < -hys
        TjinC(i) = Tjin - d;
    else
        TjinC(i) = TjinOLD;
    end
    TjinOLD = TjinC(i);

    % compute derivatives
    % main tank T
    dTdt= -1/taup * T(i) + 1/tauf * Tin + 1/taux * Tj(i);
    % jacket Tj
    dTjdt = -1/taupj * Tj(i) + 1/taufj * TjinC(i) + 1/tauxj * T(i);

    % update time and temperatures
    t(i+1) = t(i) + dt;
    T(i+1) = T(i) + dTdt * dt;
    Tj(i+1) = Tj(i) + dTjdt * dt;
    TjinC(i+1) = TjinC(i);

end

plot(t,T,'b',t,Tj,'g',t,TjinC,'r')
axis([0 max(t) min(TjinC)-1 max(TjinC)+1])
title(['auto relay tuning of tank with jacket dynamics, d = ', ...
    num2str(d),' , hys = ',num2str(hys)])
ylabel('blue = T, green = Tj, red = Tjin (K)')
xlabel('time (min)')

% now see if I can process to get out T amplitude and oscillation period

[r c] = size(t);
% t is a row vector so want c value for number of elements
maxT = max(T)
minT = min(T)
amp = (maxT-minT)/2

% WARNING: make sure you establish steady cycling and
%          have more than one complete cycle in last fraction of data
%          that you look at (lastfrac)
% lastfrac IS SET ABOVE
tlast = t(c-lastfrac*nsteps:c);
Tlast = T(c-lastfrac*nsteps:c);
Tjlast = Tj(c-lastfrac*nsteps:c);
TjinClast = TjinC(c-lastfrac*nsteps:c);
[r c] = size(TjinClast);
flag01 = 0;
flag02 = 0;
y = TjinClast(1)
for i = 1:c
    x = TjinClast(i);
    if (x ~= y) && (flag01 == 0)
        tstart = i
        flag01 = 1
        continue
    end
    if (x == y) && (flag01 == 1)
        tmid = i
        flag01 = 2;
        flag02 = 1
        continue
    end
    if (x ~= y) && (flag02 == 1)
```

```
        tend = i
        flag02 = 2
        continue
    end
end
ttstart = tstart
ttmid = tmid
ttend = tend
Tx = Tlast(tstart:tend);
Tjx = Tjlast(tstart:tend);
tx = tlast(tstart:tend);
Tjinx = TjinClast(tstart:tend);

figure
plot(tx,Tx,'b',tx,Tjx,'g',tx,Tjinx,'r')
title('the one cycle picked to get amplitude and period')
axis([min(tx) max(tx) min(Tjinx)-1 max(Tjinx)+1])
ylabel('blue = T, green = Tj, red = Tjin (K)')
xlabel('time (min)')

Pu = tlast(tend) - tlast(tstart) % min, period of auto relay cycling

% now compute PID params per Doyle, "Process Control Modules..." (2000)
Ku = 4*d/pi/amp
wc = 2*pi/Pu
alpha = 4
phim = pi/4
Kc = cos(phim)*Ku
taud = tan(phim)*sqrt(4/alpha + tan(phim))/2/wc
taui = alpha*taud
```