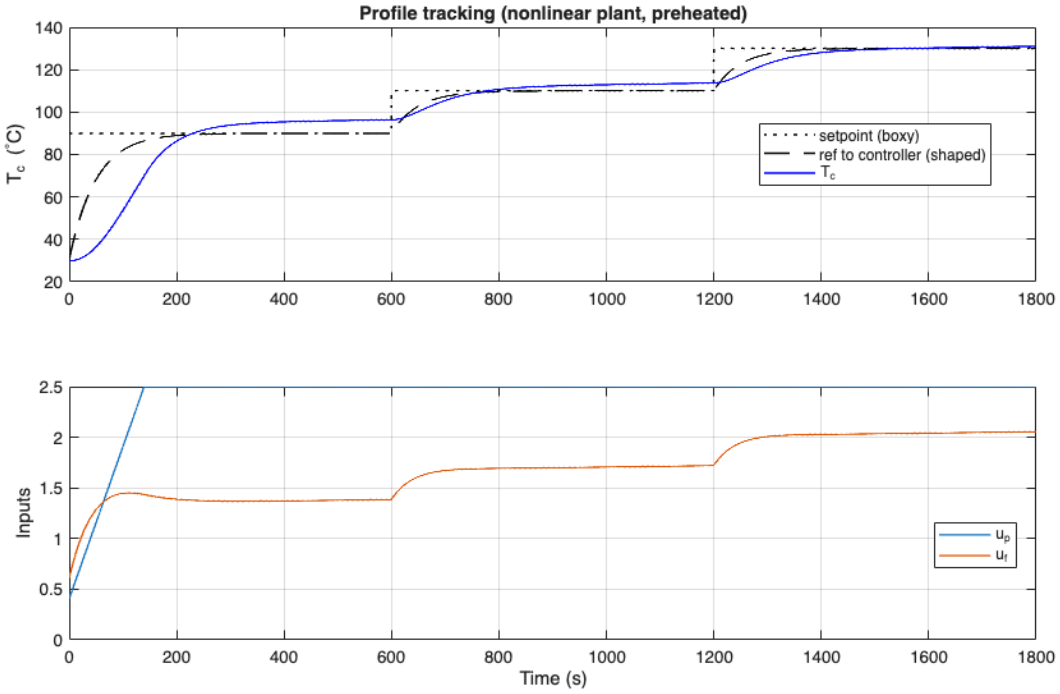
```
function profile_tracking(start_at_ambient)
% 2.7.5 — Profile tracking
%
% If start_at_ambient == true, we start from T_f = T_c = T_amb (cold smoker).
% Otherwise, we start from the equilibrium eq.xe (~110 °C), i.e. a preheated
smoker.

if nargin < 1
    start_at_ambient = false;   % default: preheated
end

[p,op] = smoker_params();                  % params
(Cf,Cc,kf,kfa,kca,beta,Tamb,...)
[A,B,C,~] = smoker_lin(p, op);             % linearization used for design
eq  = smoker_eq(p, op);
u0  = eq.ue;                               % [up0; uf0]   nominal inputs
y0  = eq.y0;                               % nominal Tc

% State-feedback + observer design
opts.method = 'place';
poles_ctrl = [-0.02 -0.03 -0.05 -0.06];   % gentle, well-damped
poles_obs  = [-0.25 -0.50];
[K,Ki_vec,~] = design_gains(p, op, poles_ctrl, poles_obs);
L           = observer_gain(p, op, poles_obs);

Ki_p = Ki_vec(1);                          % scalar integral gain on pellet
channel

% Small-signal DC gain from Tc-ref to inputs (2x1)
Kref = kref_dc(A,B,C,diag([2.5 6.0]));    % same as before

% --- Simulation / shaping parameters ---
dt      = 0.5;
T       = 1800;
TT      = (0:dt:T)';
nT      = numel(TT);

tau_ref = 50;                  % reference prefilter time constant

% Physical actuator bounds (ballpark reasonable)
up_min  = 0.3;                % g/s, avoid extinguishing fire
up_max  = 2.50;                % g/s, hard burn but not crazy
uf_min  = 0.00;                % low but not fully off
uf_max  = 5.00;                % max fan

% Per-step slew limits
du_max_up = 0.0075;            % pellets: smooth
du_max_uf = 0.02;              % fan: can move a bit faster
du_max    = [du_max_up; du_max_uf];

% --- Initial state / logs ---
if start_at_ambient
```

```
    Tamb = p.Tamb;
    xabs = [Tamb; Tamb; eq.xe(3)];    % cold smoker, same hopper mass
    r_f  = Tc_profile(0);            % start ref at first segment (90°C)
else
    xabs = eq.xe;                    % preheated at nominal equilibrium
    r_f  = y0;                       % start ref at nominal Tc
end

zhat = zeros(3,1);                   % observer state (deviation coords)
xI   = 0;                            % scalar integrator on Tc error
uabs = u0;                           % absolute input vector [up; uf]

Tc_tr = zeros(nT,1);                 % Tc trace
UP    = zeros(nT,1);                 % up trace
UF    = zeros(nT,1);                 % uf trace
Rset  = zeros(nT,1);                 % boxy setpoint
Rshp  = zeros(nT,1);                 % shaped reference
MP    = zeros(nT,1);                 % pellet mass
MP(1) = xabs(3);

for k = 1:nT
    t = TT(k);

    % --- profile: 0–600:90, 600–1200:110, 1200–1800:130 ---
    r_set   = Tc_profile(t);
    Rset(k) = r_set;

    % reference shaping
    r_f     = r_f + dt*(r_set - r_f)/tau_ref;
    Rshp(k) = r_f;

    % outputs & errors
    y     = xabs(2);          % actual Tc
    ydev  = y - y0;           % deviation from nominal
    e     = r_f - y;          % tracking error

    %control in deviation coordinates

    % feedforward (2x1) and state-feedback (2x1 on zhat)
    du_ff = Kref*(r_f - y0);           % small-signal DC feedforward
    % integral state affects pellet channel only: u_int = -Ki_p * xI
    u_int = -Ki_p * xI;
    du_fb = -K*zhat + [u_int; 0];      % 2x1 feedback+integral

    % pre-saturation absolute command
    u_unsat = u0 + du_ff + du_fb;      % [up_unsat; uf_unsat]

    % apply static saturation bounds
    up_sat = min(max(u_unsat(1), up_min), up_max);
    uf_sat = min(max(u_unsat(2), uf_min), uf_max);
    u_cmd  = [up_sat; uf_sat];

    % apply per-step slew limits to get actual commanded uabs
    uabs = slew_limit(uabs, u_cmd, du_max);   % uses helper
```

```
    % anti-windup on pellet integrator xI
    % tell anti_windup about the pellet channel only
    xI = anti_windup(xI, e, dt, ...
                     u_unsat(1), ...   % pre-sat pellet command
                     uabs(1),    ...   % post-slew/post-sat pellet command
                     Ki_p,       ...   % integral gain in u_int = -Ki_p*xI
                     up_min, up_max);  % actuator limits for up

    % --- plant + observer update ---
    xabs = xabs + dt*smoker_nl(t, xabs, @(~)uabs, p);

    % use pellet deviation part for observer input (consistent with design)
    du_dev = [uabs(1) - u0(1); 0];     % only pellet deviation into B
    zhat   = zhat + dt*(A*zhat + B*du_dev + L*(ydev - C*zhat));

    % logs
    Tc_tr(k) = y;
    UP(k)    = uabs(1);
    UF(k)    = uabs(2);
    MP(k)    = xabs(3);
end

pellets_used = MP(1) - MP(end);
fprintf('Pellets used over profile: %.1f g\n', pellets_used);

% --- plots ---
fig = figure(5); clf(fig); tiledlayout(fig,2,1)

nexttile;
plot(TT,Rset,'k:',TT,Rshp,'k--',TT,Tc_tr,'b-','LineWidth',1.1); grid on
ylabel('T_c (^{\circ}C)');
if start_at_ambient
    title('Profile tracking (nonlinear plant, cold start)')
else
    title('Profile tracking (nonlinear plant, preheated)')
end
legend('setpoint (boxy)','ref to controller
(shaped)','T_c','Location','best')

nexttile;
plot(TT,UP,'-','LineWidth',1.0); hold on
plot(TT,UF,'-','LineWidth',1.0); grid on
ylabel('Inputs'); xlabel('Time (s)');
legend('u_p','u_f','Location','best')
end
```

```
Pellets used over profile: 4354.6 g
```



Published with MATLAB® R2025a