```
function [para_est, f_factor, alpha_rt, SW_NF98] = PAA2(xk, SW, F_init,
 theta_init, ...
    lambda_init, lambda_end, SW_lambda, alpha_init, alpha_end,
 SW_2Stage,lambda_gain)
%#eml
persistent N N_vec x phi F theta lambda x_hat e iter phi_F_phi alpha ...
    delta_theta sum_dtheta;

persistent alpha2 x_hat_post v e_post ek_post fxd_comp lambda2...
    switch_iter_track flag_series_parallel flag_judge_alg count1stage;% 2nd-
stage PAA
persistent flag_jump flag_jump_pre flag_jump_prePre lambda_temp iter_jump
 theta_pre theta_pre_n;
% persistent    switch_iter_track               RT_adapOn_iter_track; %
 2012-09-11 enhanced local convergence

if isempty(N)
    N       = uint16(2);
    N_vec   = uint16(4);
    x       = zeros(4,1);
    phi     = [0; 0];

    F       = F_init;
    theta   = theta_init; %
    lambda  = lambda_init;

    x_hat   = 0;
    e       = 0;
    iter    = 0;
    phi_F_phi = 0;
    delta_theta = [0;0];
    sum_dtheta =0;
    alpha = alpha_init;

    % 2nd-stage PAA
    e_post = zeros(4,1);
    alpha2 = 0.96;
    fxd_comp = theta_init;
    flag_series_parallel = 1;
    flag_judge_alg = 1;
    count1stage = 0;
    v = 0;
    ek_post = 0;
    x_hat_post = 0;
    switch_iter_track = 0;
    lambda2 = 1;
    flag_jump = 0;
    lambda_temp = 1;
    iter_jump = 0;
    flag_jump_pre = 0;
    flag_jump_prePre = 0;
    theta_pre = theta;
    theta_pre_n = 2;
end
if SW == 0
    %========= Adaptation turned off. ==========
    % update observation vector
    x(2:4)   = x(1:4-1);
    x(1)     = xk;

    para_est = theta;
    f_factor = lambda;
    alpha_rt = alpha;
    SW_NF98 = 1;
else
    if flag_series_parallel
```

read data

```
        phi     = [-x(1)-x(3);-x(2)];%TBD
        % a priori estimate
        x_hat =phi'*theta-x(4);%TBD
        % a priori estimation error
        e = xk - x_hat;
        phi_F_phi = phi'*F*phi;
        % adaptation gain update
        F = 1/lambda * ( F - F*(phi*phi')*F / (lambda+phi_F_phi) );
        delta_theta = F*phi*e/(lambda+phi_F_phi);%TBD
        % parameter estimation update
        theta = theta + delta_theta;
        % a posteriori prediction
        x_hat_post = phi'*theta-x(4);%TBD
        if SW_lambda == 0 || SW_lambda == 2
            if SW_2Stage == 0
                %         alpha = alpha_init;
                if SW_lambda == 0
                    lambda = lambda_end -...
                        (lambda_end-lambda) * 0.996;
                    alpha = alpha_end - (alpha_end-alpha) * 0.99;
                elseif SW_lambda == 2
                    if iter < 100
                        lambda = lambda_end -...
                            (lambda_end-lambda) * 0.996;
                    else
                        lambda = 1-( 1-phi'*F*phi/(1+phi_F_phi) )*...
                            (xk-x_hat_post)^2*5e6;%/0.00000018;%0125 000030
                        if lambda < 0.5
                            lambda = 0.5;
                            alpha = alpha_init; % alpha adap
                        end
                    end
                    alpha = alpha_end - (alpha_end-alpha) * 0.99;
                end
            else
                lambda = lambda_end -...
```

```
                    (lambda_end-lambda) * 0.996;
                alpha = alpha_end - (alpha_end-alpha) * 0.99;
            end
        end
        % +++++++++++++++++++++++++++++++++++++++++++
        % filter state update
        x(2:4)      = x(1:4-1);
        x(1)        = xk;

        switch_iter_track = iter;
        count1stage       = count1stage + 1;
        if SW_2Stage
            if (SW_lambda == 2) || (SW_lambda == 0)
                if 1
                    if count1stage > 200
                        flag_series_parallel    = 0;
                    end
                end
            end
        end

    else        %          SW_2Stage == 1 and algorithm converged
        if iter == switch_iter_track + 1 % initialize for the local algorithm
            lambda2 = 0.99;
            alpha2 = 0.97;
            e_post = [0;0;0;0];
            theta_pre = theta;
            theta_pre_n = ceil(switch_iter_track/100)+5;
            lambda_temp = 1;
        end

        fxd_comp    = theta;
        % regressor update
        phi = [-x(1)-x(3)+alpha2*e_post(1)+(alpha2^3)*e_post(3);x(2)-
alpha2*e_post(2)];%TBD
        % a priori prediction error
        e   = xk+x(4)-alpha2^4*e_post(4)-phi'*theta;%TBD
        % a priori adaptation error
        psi=[alpha2*e_post(1)+alpha2^3*e_post(3);alpha2^2*e_post(2)];%TBD
        v   = e+psi'*fxd_comp+alpha2^4*e_post(4);%TBD
        phi_F_phi = phi'*F*phi;
        F = 1/lambda2 * ( F - F*(phi*phi')*F / (lambda2+phi_F_phi) );
        % parameter estimation update
        theta = theta + (F*phi*v/(1+phi_F_phi));%TBD
        % a posteriori prediction
        ek_post = xk+x(4)-alpha2^4*e_post(4)-phi'*theta;%TBD
        x_hat_post = xk - ek_post;
        if flag_judge_alg
            lambda2 = lambda_temp - (lambda_temp-lambda2) * 0.996;%0.98;
        end
        % ////////
        if iter==(theta_pre_n*100)
            theta_pre(2) = theta_pre(1);
            theta_pre(1) = ek_post;%theta(1);
            theta_pre_n = theta_pre_n + 1;
        end
        if iter > 500
                if abs(xk-x_hat_post)>0.03
                    flag_jump = 1;
                end
                if flag_jump_pre == 0 && flag_jump == 1
                    iter_jump = iter;
                        lambda2 = 0.95;
                        lambda_temp = 0.999;
                        flag_judge_alg = 1;
                    if flag_jump_prePre == 1 % if i had just jumped
                        lambda2 = 0.993;
                        lambda_temp = 0.999;
                        flag_judge_alg = 0;
                    end
                    flag_jump_pre = 1;
                    flag_jump_prePre = 1;
                end
                if iter-iter_jump > 120
                    flag_jump_prePre = 0;
                elseif iter-iter_jump > 60
                    flag_jump_pre = 0;
                    flag_jump = 0;
                end
        else
%             lambda_temp = 0.99;
        end
        % ////////
        alpha2 = 0.996 - (0.996-alpha2) * 0.97;
        if SW_lambda == 2
            if (iter == 3*800+0) || ...
                    (iter == 6*800+0) || ...
                    (iter == 9*800+0) || ...
                    (iter == 12*800+0)
                flag_series_parallel = 1;
                lambda = lambda_init-0.02;
%                 F = [1 0;0 1]*500;
                count1stage = 0;
            end
        elseif SW_lambda == 3
            if (iter == 5*800-1) || ...
                    (iter == 14*800-1)
                flag_series_parallel = 1;
                lambda = 0.991;
            end
        end
        % TBD
        % Update states vectors x and e_post
        x(2:4)=x(1:4-1);
        x(1)=xk;
        e_post(2:4)=e_post(1:4-1);
        e_post(1)=ek_post;
    end
    iter = iter + 1; % iteration update
    para_est = theta;
    f_factor = lambda2;
    end
    alpha_rt = alpha;
    if 1 %abs(theta)<1.5609 %2*cos(2*pi/800*86)%1.5208% 2*cos(2*pi/800*90)
        SW_NF98 = 1;
    else
        SW_NF98 = 0;
    end
end
```

```
Not enough input arguments.

Error in sim (line 18)
    F       = F_init;
```

Published with MATLAB® R2022a