Contents


```
function u = UAVControl(x0,stateCmd,stateCmdDot, data)
```

```
% Compute Lift (Lbar), bank angle (phi) and Thrust (Tbar) required for
% a commanded state.
% INPUTS:
% x0            (6,1)          %
% State:     x = [V;gamma;psi;x;y;h;Tbar]
%   --------------------------------------
%     V     true airspeed (m/s)
%     gamma  air relative flight path angle (rad)
%     psi   air relative flight heading angle (rad)
%     x     East position (m)
%     y     North position (m)
%     h     altitude (m)
%     Tbar  normalized excess thrust
% stateCmd      (5,1)   Commanded velocity, heading, altitude, and horizontal
%                       position.  [v;psi;h;x;y]
%
% stateCmdDot   (3,1)   Commanded rate of change of velocity, heading, altitude.
%                       [vDot;psiDot;hDot]
%
%
% data              Data structure with fields:
%                       g     Gravitational acceleration (1,1)
%                       Kh    Altitude control gains     (1,2)
%                       KL    Lateral control gains      (1,2)
%                       Ks    Longitudinal control gains (1,2)
% OUTPUTS:
% u             (1,3)   Command vector [Lbar, phi, Tbar]
%                       Lbar     Normalized Lift required, (1,1)
%                       phi      Bank angle (x = East, y = North), (rad), (1,1)
%                       Tbar     Normalized Thrust required, (1,1)
```

Demo

```
if nargin == 0
    disp('Demo Mode')
    x0 = [ 1;  1; 0; 0; 0; 0;1];
    stateCmd = [ 2;  2; 4; 3; 1];
    stateCmdDot = [ 1;  1; pi/4];
    K = [ 1; 1; 1; 1; 1; 1];
    data.g = 9.81; % (m/s)
    data.Kh = [1,1];
    data.KL = [1,1];
    data.Ks = [1,1];
end
```

Input Checking

```
% check state vector is complete
if length(x0) ~= 7 || ~isreal(x0)
    error('Error inital state not complete or is not real')
end
% check stateCmd vector is complete and real
if length(stateCmd) < 5 || ~isreal(stateCmd)
    error('stateCmd must have 5 real elements')
end
% check stateCmdDot is complete and real
if length(stateCmdDot) < 3 || ~isreal(stateCmdDot)
    length(stateCmdDot)
    ~isreal(stateCmdDot)
    error('stateCmdDot must have 3 real elements')
end
% check data is a structure
if ~isstruct(data)
    error('data must be a struct')
end
```

Constants

```
% gravitational acceleration on Earth
g = data.g; % (m/s/s)

% uncertainites
nH = 0; % altitude uncertainty, (m)
nHDot = 0; % altitude derivative uncertainty, (m/s)
nV = 0; % speed uncertainty, (m/s)
nPsi = 0; % air-relative heading uncertainty, (rad)
nZeta = 0; % along-track position uncertainty (m)
nEta = 0; % cross track position uncertainty (m)

% define gains
Kh1 = data.Kh(1);
Kh2 = data.Kh(2);
KL1 = data.KL(1);
KL2 = data.KL(2);
KN1 = data.Ks(1);
KN2 = data.Ks(2);
```

Compute function

```
% pull apart x0 vector
v = x0(1,1);
gamma = x0(2,1);
psi = x0(3,1);
xe = x0(4,1);
yn = x0(5,1);
h = x0(6,1);

% define derivatives necessary for computing Lbar, phi, Tcbar
hDot = v*sin(gamma);
xeDot = v*cos(gamma)*sin(psi);
ynDot = v*cos(gamma)*cos(psi);
```

```
% pull apart stateCmd
vCmd = stateCmd(1,1);
psiCmd = stateCmd(2,1);
hCmd = stateCmd(3,1);
xeCmd = stateCmd(4,1);
ynCmd = stateCmd(5,1);

% pull apart stateCmdDot
vCmdDot = stateCmdDot(1,1);
psiCmdDot = stateCmdDot(2,1);
hCmdDot = stateCmdDot(3,1);

% compute ground speed
vGround = sqrt(xeDot^2 + ynDot^2);

% compute zeta and eta
values = [sin(psiCmd) cos(psiCmd); cos(psiCmd) -1*sin(psiCmd)] * [ xe - xeCmd; yn - ynCmd ];
zeta = values(1); eta = values(2);
```

Compute phi, Lbar and Tbar

```
% phi = bank angle, +- pi/2
% Lbar = normalized excess thrust
% Tbar = normalized excess thrust

% make sure phi is real
X = (vCmd/g)*psiCmdDot - ((KL1*vCmd)/g)*(psi - psiCmd + nPsi) - (KL2/g)*(eta + nEta);

if X > sin(pi/2)
    X = sin(pi/2);
end
if X < -sin(pi/2)
    X = -sin(pi/2);
end

phi = asin(X);
Lbar = 1/cos(phi)*(1-Kh1/g*(hDot - hCmdDot + nHDot) - Kh2/g*(h - hCmd + nH));
Tbar = sin(gamma) + vCmdDot/g - KN1/g*(vGround - vCmd + nV) - KN2/g*(zeta + nZeta);

phiMax = pi/2; LbarMax = 10; TbarMax = 1;

if phi > phiMax
    phi = phiMax;
elseif phi < -phiMax
    phi = -phiMax;
end

if Lbar > LbarMax
    Lbar = LbarMax;
elseif Lbar < -LbarMax
    Lbar = -LbarMax;
end

if Tbar > TbarMax
    Tbar = TbarMax;
elseif Tbar < -TbarMax
    Tbar = -TbarMax;
end
```

```
u = [Lbar, phi, Tbar];

if ~isreal(u)
    u
    error('Imaginary control vector')
end
```

Published with MATLAB® R2020a