Contents

- Input checking
- Demo
- Dynamic Model


```
function [rhs] = UAVRHS(x,u,g,tau)
```

```
% Given state vector "x", command vector "u", and constants "g" and "tau",
% compute the Right Hand Side state derivative
%
% Usage:
% UAV_RHS(x, u, g, tau)
%
% INPUTS:
% x - (7,1) element state vector
%     V       true airspeed (m/s)
%     gamma   air relative flight path angle (radians)
%     psi     air relative flight heading angle (radians)
%     x       East position (m)
%     y       North position (m)
%     h       altitude (m)
%     Tbar    normalized excess thrust (N)
%
% u - (3,1) element command vector
%     v       velocity command (true airspeed, m/s)
%     psi     heading command (rad)
%     h       altitude command (m)
%
% g - gravity (m/s^2)
% tau - engine thrust response time (s)
%
% OUTPUTS:
% rhs - (7,1) right hand side output of the changes in the state vector

% initalize rhs
rhs = [0;0;0;0;0;0;0];
```

Input checking

```
if g <= 0
    error("G must not be negative")
end
if tau < 0
    error("Engine thrust response time must be greater than zero")
end
if length(x) ~= 7
    error("State vector must contain 7 elements")
end
if length(u) ~= 3
    error("Control vector must contain 3 elements")
end
```

Demo

```
if nargin == 0
    x = [5;0;0;6;10;15;2];
    u = [2;5;3];
    g = 9.81;
    tau = 2;
end
```

Dynamic Model

```
% current state
V = x(1);                   % true airspeed
gamma = x(2);               % air-relative flight path angle
psi = x(3);                 % air-relative heading
x_e = x(4);                 % East position
y_n = x(5);                 % North position
h = x(6);                   % altitude
T = x(7);                   % normalized excess thrust

% current controls
L = u(1);                   % velocity command
phi = u(2);                 % heading command
T_c = u(3);                 % altitude command

% State limits
vMax = 30; % max speed (m/s)
hMax = 39; % near top of course (m)
hMin = 1;  % near bottom of course of course (m)

% rhs calculations

vDot = T*g - g*sin(gamma);                    % vDot
if (abs(V) < vMax)
    rhs(1) = vDot;
elseif (sign(V) == sign(vDot))
    % Case when V is beyond maximum: only allow vDot that reduces abs(V)
    % When V and vDot have same sign, this will INCREASE abs(V)

    % Prevent accelerating beyond vMax
    rhs(1) = 0;
else
    rhs(1) = vDot; % vDot is decreasing abs(V)
end

rhs(2) = 1/V * (g*L*cos(phi) - g *cos(gamma)); % gammaDot
rhs(3) = 1/(V*cos(gamma))*(g*L*sin(phi));      % psiDot
rhs(4) = V*cos(gamma)*sin(psi);                % xDot
rhs(5) = V*cos(gamma)*cos(psi);                % yDot

hDot = V*sin(gamma);                           % hDot
if (h > hMin && h < hMax)
    rhs(6) = hDot;
elseif (h < hMin && hDot > 0)
    % Allow hDot only if it INCREASES h
    rhs(6) = hDot;
elseif (h > hMax && hDot < 0)
    % Allow hDot only if it DECREASES h
    rhs(6) = hDot;
else
    % Prevent leaving bounds of course
```

```
    rhs(6) = 0;
end
rhs(7) = 1/tau*(T_c-T);                         % TbarDot
```

```
end
```

Published with MATLAB® R2020a