```matlab
function [state_matrix,F,stage] = state_matrix_func(const,~,state)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Function that takes in the current state of the rocket, and has
% equations for calculating the derivatives based on current state

% This function will output the derivatives of each part of the state
% matrix

% state_matrix = d/dt[x;vx;z;vz;mr;V;m_air]

% This function outputs the values of each of these derivatives for ode45,
% and F, the thrust, and stage, an integer value of the stage the rocket is
% in.

% This function requires input of the constant structure, time, and the
% current state since the derivatives are dependant on the current state.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %Constants
   %These come from the const struct that is inputted

    g = const.g; %m/s^2
    c_dis = const.c_dis;
    rho_air = const.rho_air; %kg/m^3
    V_b = const.V_b; %m^3
    p_atm = const.p_atm; %pa
    gamma = const.gamma;
    rho_w = const.rho_w; %kg/m^3
    R_air = const.R_air; %J/kgK
    c_D = const.c_D;
    p_0 = const.p_0; %pa
    theta_i = const.theta_i; %degree
    z0 = const.z_0; %m
    l_s = const.l_s; %m
    At = const.At; %m^2
    Ab = const.Ab; %m^2
    V_0a = const.V_0a; %m^3
    m_0a = const.m_0a; %kg

    % Lowercase v is velocity, uppercase V is  volume

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Devectorizing the State matrix
    x1 = state(1);
    x2 = state(2);

    z1 = state(3);
    z2 = state(4);
```

```
m_r = state(5);
V_air = state(6);
m_air = state(7);

 %Calculating distance and theta, to see if the bottle is of the launch
 %stand, and locking theta if it is.
distance = sqrt(x1^2+(z1-z0)^2);

if distance < l_s
    theta = theta_i;
else
    theta = atan(z2/x2);
end

%Calculating dynamic pressure and drag at any point in the integration
v_bottle = sqrt(x2^2+z2^2);
q = 0.5 * rho_air * (v_bottle^2);
drag = q*Ab*c_D;
```

%%

```
Not enough input arguments.

Error in state_matrix_func (line 24)
    g = const.g; %m/s^2
```

Stages of flight

```
if V_air < V_b  % Stage 1: Water expulsion

    %Calculating pressure of air
    p = p_0*(V_0a/V_air)^gamma;


    %Calculating state of exhaust
    v_e = sqrt(2*(p-p_atm)/rho_w); %Velocity of the exhausted water

    %calulating mass flow rate of water
    mdot_w = c_dis * rho_w * At * v_e;

    %Calculating thrust
    F = mdot_w * v_e;  %Thrust created by exhausted water

    %Calculating the time rate of change of water
    dVdt = c_dis * At * v_e;

    %Finding net forces in x and z directions
    FnetZ = F*sin(theta)-drag*sin(theta)-m_r*g;
    FnetX = F*cos(theta)-drag*cos(theta);
```

```
        ddt1 = x2; %x Velocity
        ddt2 = FnetX/m_r; %Thrust - drag
        ddt3 = z2; %Z velocity
        ddt4 = FnetZ/m_r;  %Thrust-drag-gravity
        ddt5 = -mdot_w; % Current total mass change in kg/s
        ddt6 = dVdt; %Change in volume of air
        ddt7 = 0; %Change in mass of air
        %disp('Water expulsion');
        stage = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    elseif V_air >= V_b   % Stages 2-4


        %Calculating P end and T end for calculating pressure later
        p_end = p_0*(V_0a/V_b)^gamma;


        %Calculating density, Temperature, and pressure of air
        rho = m_air/V_b;
        p = p_end*(m_air/m_0a)^gamma;
        T = p/(rho*R_air);

        %these are the actual current temperature and density

        %Determining if the flow is choked to find v_e
        p_star = p*(2/(gamma+1))^(gamma/(gamma-1));

        if p_star > p_atm
            T_e = (2/(gamma+1))*T;
            rho_e = p_star/(R_air*T_e);
            v_e = sqrt(gamma*R_air*T_e);
            p_e = p_star;

        else
            M = sqrt(((p/p_atm)^((gamma-1)/gamma)-1)*(2/(gamma-1)));
            T_e = T*(1+(gamma-1)/2*M^2)^-1;
            rho_e = p_atm/(R_air*T_e);
            p_e = p_atm;

            v_e = M*sqrt(gamma*R_air*T_e);


        end


        %Calculating thrust
        m_dota = c_dis*rho_e*At*v_e;
        F = m_dota*v_e + (p_e-p_atm)*At;

        %Calculating change in total mass
        mdot_r = -c_dis*rho_e*At*v_e;
```

```
        FnetZ = F*sin(theta)-drag*sin(theta)-m_r*g;
        FnetX = F*cos(theta)-drag*cos(theta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if p>p_atm %Stage 2: Air Expulsion

            ddt1 = x2; %x Velocity
            ddt2 = FnetX/m_r; %Thrust - drag
            ddt3 = z2; %Z velocity
            ddt4 = FnetZ/m_r;  %Thrust-drag-gravity
            ddt5 = mdot_r;
            ddt6 = 0;
            ddt7 = mdot_r;
            %disp('Air Expulsion');
            stage = 2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        elseif p <= p_atm && z1 > 0  %Stage 3: Ballistic Stage

            ddt1 = x2;
            ddt2 = -cos(theta)*drag/m_r;
            ddt3 = z2;
            ddt4 = -sin(theta)*drag/m_r-g;
            ddt5 = 0;
            ddt6 = 0;
            ddt7 = 0;
            %disp('Ballistic');
            stage = 3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        elseif z1<=0   %Stage 4: Landed (no change in any variable)

             ddt1 = 0;
             ddt2 = 0;
             ddt3 = 0;
             ddt4 = 0;
             ddt5 = 0;
             ddt6 = 0;
             ddt7 = 0;
             %disp('Landing');
             stage = 4;

        end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    else  %Error phase just in case
            ddt1 = 0;
            ddt2 = 0;
            ddt3 = 0;
            ddt4 = 0;
            ddt5 = 0;
```

```
        ddt6 = 0;
        ddt7 = 0;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Creating the state matrix with the parts to integrate

```
%This will be outputted by the function and will be put into ode45

    state_matrix = [ddt1;ddt2;ddt3;ddt4;ddt5;ddt6;ddt7];

end
```

Published with MATLAB® R2023a