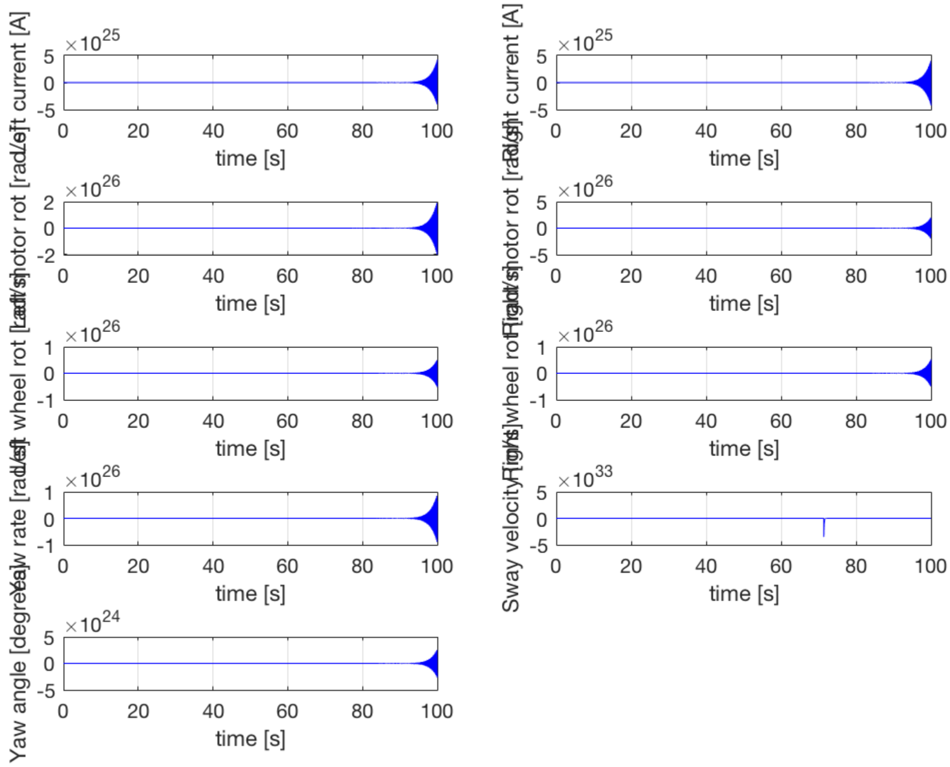
```
% Simulation of a continuous dynamic system described by ordinary
% differential equations.
%
% INITIAL SEGMENT
%
% Define constant model parameters as global variables. Set the
 initial
% conditions of the states and set up the input to the model.
%
% It is good practice to clear the MATLAB workspace at the beginning
 of
% a program

clear all
clc

% Define and initialise the model input and any model parameters
% as global variables so that they can be read in the function
 model.m.

global Vd L R Kt Ke bs Jm Jw Kc KH KN KS KV KY VT Rw Rm Gc KI siref

Vd = 2;                 % Voltage
L = 0.1;                % Inductance
R = 4;                  % Resistance
Kt = 0.35;              %
Ke = 0.35;              %
bs = 0.01;              %
Jm = 0.002;             %
Jw = 0.0011;            %
Kc = 2.2;               %
KH = 2.2;               %
KN = 14.14;             %
KS = 9.81;              %
KV = 0.466;             %
KY = 29.94;             %
VT = 0.75;              %
Rw = 0.064;             %
Rm = 0.124;             %
Gc = 33.3;              %
KI = 0.65;              %
siref = 0.785398;       %

% Define parameters for the simulation

stepsize = 0.001;     % Integration step size
comminterval = 0.01;    % Communications interval
EndTime = 100;      % Duration of the simulation (final time)
i = 0;        % Initialise counter for data storage

% Initial conditions of all states and state derivatives
```

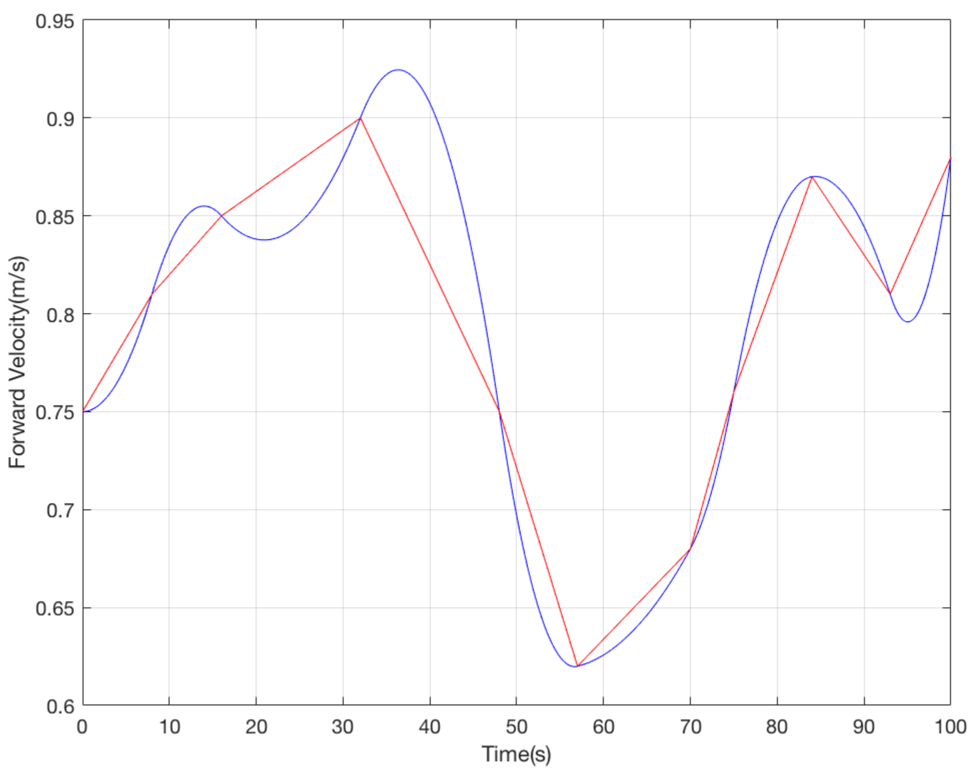
```
x = [0,0,0,0,0,0,0,0,-0.174533]';
xdot = [0,0,0,0,0,0,0,0,0]';
int = 0;
Time = [0,8,16,32,48,57,70,75,84,93,100];
Vel = [0.75,0.81,0.85,0.9,0.75,0.62,0.68,0.76,0.87,0.81,0.88];

% Generate the z values for the spline
count = numel(Time);
z(1) = 0;
for n = 2:1:count
    z(n) = -z(n-1)+2*((Vel(n)-Vel(n-1))/(Time(n)-Time(n-1)));
end

% END OF INITIAL SEGMENT - all parameters initialised
%
% DYNAMIC SEGMENT
%
% The DYNAMIC SECTION is the main section of a simulation program.
 This is
% evaluated for every time interval during the simulation. Therefore
 it is
% an interative process.

for time = 0:stepsize:EndTime,

    % store time state and state derivative data every communication
 interval

    if rem(time,comminterval)==0

        i = i+1;     % increment counter
        tout(i) = time;            % store time
        xout(i,:) = x;       % store states
        xdout(i,:) = xdot;         % store state derivatives
    end            % end of storage

    % Equation for the quadratic spline
if time >= 0 && time < 8
    Vt = ((z(2)-z(1))/(2*((Time(2)-Time(1))))*(time-
Time(1))^2+z(1)*(time-Time(1))+Vel(1));
elseif time >= 8 && time < 16
    Vt = ((z(3)-z(2))/(2*((Time(3)-Time(2))))*(time-
Time(2))^2+z(2)*(time-Time(2))+Vel(2));
elseif time >= 16 && time < 32
    Vt = ((z(4)-z(3))/(2*((Time(4)-Time(3))))*(time-
Time(3))^2+z(3)*(time-Time(3))+Vel(3));
elseif time >= 32 && time < 48
    Vt = ((z(5)-z(4))/(2*((Time(5)-Time(4))))*(time-
Time(4))^2+z(4)*(time-Time(4))+Vel(4));
elseif time >= 48 && time < 57
    Vt = ((z(6)-z(5))/(2*((Time(6)-Time(5))))*(time-
Time(5))^2+z(5)*(time-Time(5))+Vel(5));
elseif time >= 57 && time < 70
```

```
    Vt = ((z(7)-z(6))/(2*((Time(7)-Time(6)))))*(time-
Time(6))^2+z(6)*(time-Time(6))+Vel(6));
elseif time >= 70 && time < 75
    Vt = ((z(8)-z(7))/(2*((Time(8)-Time(7)))))*(time-
Time(7))^2+z(7)*(time-Time(7))+Vel(7));
elseif time >= 75 && time < 84
    Vt = ((z(9)-z(8))/(2*((Time(9)-Time(8)))))*(time-
Time(8))^2+z(8)*(time-Time(8))+Vel(8));
elseif time >= 84 && time < 93
    Vt = ((z(10)-z(9))/(2*((Time(10)-Time(9)))))*(time-
Time(9))^2+z(9)*(time-Time(9))+Vel(9));
elseif time >= 93 && time < 100
    Vt = ((z(11)-z(10))/(2*((Time(11)-Time(10)))))*(time-
Time(10))^2+z(10)*(time-Time(10))+Vel(10));
end

Vt_array(i) = Vt;       % Store velocities in an array

    % DERIVATIVE SECTION
 %
 % The DERIVATIVE SECTION contains the statements needed to evaluate
 the
 % state derivatives - these statements define the dynamic model
 (model.m)

    int = int + (stepsize*((Kc*siref)-(KH*x(9))));
    deltav = (Gc*((siref*Kc)-(KH*x(9)))) + (int*Gc*KI);

    Vin(1) = Vd - deltav;
    Vin(2) = Vd + deltav;

    xdot = state_space_model(x,Vin);

 % END OF DERIVATIVE SECTION
 %
 % INTEG SECTION
 %
    % Numerical integration of the state derivatives for this time
 interval

    x = rk4int('state_space_model', stepsize, x, Vin);

    % END OF INTEG SECTION

end

% END OF DYNAMIC SEGMENT

%
% TERMINAL SEGMENT
%
% The TERMINAL SEGMENT contains statements that are executed after the
 simulation
% is complete e.g. plotting results
```

```
figure(1)          % define figure window number
subplot(5,2,1)                                   % select the first
 subplot
plot(tout,xout(:,1),'b-')                        % plot state 1 against
 time
xlabel('time [s]')                               % label x-axis
ylabel('Left current [A]')                       % label y-axis
grid on                                          % turn grid on

subplot(5,2,2)                                   % select the second
 subplot
plot(tout,xout(:,4),'b-')                        % plot state 4 against
 time
xlabel('time [s]')                               % label x-axis
ylabel('Right current [A]')                      % label y-axis
grid on                                          % turn grid on

subplot(5,2,3)                                   % select the third
 subplot
plot(tout,xout(:,2),'b-')                        % plot state 2 against
 time
xlabel('time [s]')                               % label x-axis
ylabel('Left motor rot [rad/s]')                 % label y-axis
grid on                                          % turn grid on

subplot(5,2,4)                                   % select the forth
 subplot
plot(tout,xout(:,5),'b-')                        % plot state 5 against
 time
xlabel('time [s]')                               % label x-axis
ylabel('Right motor rot [rad/s]')                % label y-axis
grid on                                          % turn grid on

subplot(5,2,5)                                   % select the fifth
 subplot
plot(tout,xout(:,3),'b-')                        % plot state 3 against
 time
xlabel('time [s]')                               % label x-axis
ylabel('Left wheel rot [rad/s]')                 % label y-axis
grid on                                          % turn grid on

subplot(5,2,6)                                   % select the sixth
 subplot
plot(tout,xout(:,6),'b-')                        % plot state 6 against
 time
xlabel('time [s]')                               % label x-axis
ylabel('Right wheel rot [rad/s]')                % label y-axis
grid on                                          % turn grid on

subplot(5,2,8)                                   % select the eighth
 subplot
plot(tout,xout(:,8),'b-')                        % plot state 8 against
 time
```

```
xlabel('time [s]')                              % label x-axis
ylabel('Sway velocity [m/s]')                   % label y-axis
grid on                                         % turn grid on

subplot(5,2,7)                                  % select the seventh
 subplot
plot(tout,xout(:,7),'b-')                       % plot state 7 against
 time
xlabel('time [s]')                              % label x-axis
ylabel('Yaw rate [rad/s]')                      % label y-axis
grid on                                         % turn grid on

subplot(5,2,9)                                  % select the nineth
 subplot
plot(tout,xout(:,9),'b-')                       % plot state 9 against
 time
xlabel('time [s]')                              % label x-axis
ylabel('Yaw angle [degrees]')                   % label y-axis
grid on                                         % turn grid on

figure(2)
plot(tout,Vt_array,'b-')
hold on
plot(Time,Vel,'r-')
xlabel('Time(s)')
ylabel('Forward Velocity(m/s)')
grid on

% END OF TERMINAL SECTION

% END OF SIMULATION PROGRAM
```


time [s]
Yaw angle [degrees]
Sway velocity [m/s]


Forward Velocity(m/s)
Time(s)

Published with MATLAB® R2016a