Saint Martin's
UNIVERSITY

Modeling and Simulation
Project 2

by

Steven Araki

John Scotto Rodriguez

Department of Mechanical Engineering, St. Martin's University
ME 468: Modeling & Simulation
Dr. Shawn Duan
December 3, 2023

Table of Contents

A. Introduction

In this paper, we present a comprehensive analysis of the motion and dynamics of a piston-crankshaft system within an internal combustion engine. Our approach employs advanced techniques such as multibody modeling and virtual prototyping to delve into the fundamental mechanics of this crucial engine component. The piston-crankshaft system serves a pivotal role in converting reciprocating motion to rotary motion, making it essential for understanding and optimizing engine performance.

a. Selection.

We propose to analyze the motion and dynamics of a piston-crankshaft system in an internal combustion engine using multibody modeling and virtual prototyping techniques. This system is a fundamental component of engines and plays a crucial role in converting reciprocating motion into rotary motion.

b. Why we selected this

The piston-crankshaft system was selected for analysis due to its wide-ranging applications in the automotive, aviation, and industrial sectors. Understanding its motion and dynamics is crucial for optimizing engine performance, minimizing vibrations, and enhancing fuel efficiency. Moreover, this analysis will serve as a foundational study for understanding complex mechanical systems. We believe that addressing key challenges of the piston-crankshaft allows for developmental growth for individuals in the field of mechanical engineering. We will be showing this using Autolev in explaining how motion and dynamics of the piston play a crucial role in the functionality of the motion of the multibody part.

c. Significance

The significance of this project lies in its potential to improve engine efficiency, reduce emissions, and enhance overall system reliability. It is also of personal interest because it provides a deep understanding of the interplay between mechanical components and the potential to innovate in a vital engineering area. Understanding the efficiency behind crankshafts allows for individuals to create a more pivoting system that outlines the importance of motion and dynamics within a combustion engine crankshaft. This outline of significance can lead to overall improved translations of the crankshaft.

d. Concepts and Methods

The knowledge acquired from ME 468/MME 568, and GE 205 particularly in kinematics, dynamics, and numerical simulation, will be applied to model the piston-crankshaft system accurately and derive its equations of motion. These concepts and methods will be instrumental in developing a robust simulation.

e. Model of the System

We will be using a model of a two-degree freedom, Cylindrical Joint. This will allow the piston to be in translational motion with the rotation of the crankshaft. It will only be a 2D motion. There will be 2 particles to consider, one in the piston and one on the connection rod. Seen in the Figure below [1].

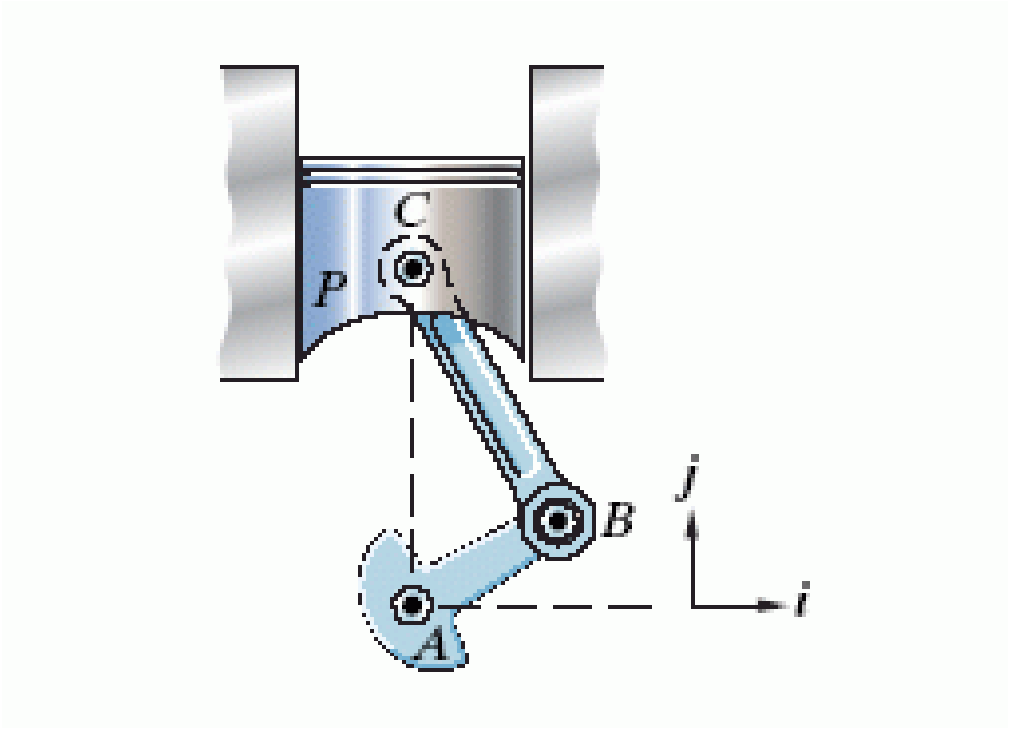


Figure 1. Slider and Crankshaft Mechanism

So, in figure 1. We will define Newtonian reference frame (N) which would be the I and j axes, the B reference frame attached to the rotational part of link (L_{AB}) particle B, frame C is Attached to particle C, which is mounted on link (L_{BC}), point A is the center of the crankshaft, and the constants are the lengths of the links, L_{AB} and L_{BC}.

B. Theory & Procedure

This section details the theoretical underpinnings and procedural steps involved in our analysis. The motion analysis encompasses predicting forces, velocities, accelerations, and torques at

various points in the system. The subsequent presentation of kinematic equations breaks down the complex relationships involved in the piston-crankshaft system's motion, providing a detailed understanding of its behavior.

a. Motion Analysis

The project will involve modeling the piston-crankshaft system's geometry and constraints to analyze its motion. The analysis will include predicting forces, velocities, accelerations, and torques at various points within the system. By analyzing these parameters, we can gain insights into the engine's performance and durability.

With the frames, particles, points and constants explained in the previous section. We can write the kinematic equations as follows.

b. Kinematic Equations:

i. Position Equations:

The piston Instantaneous position can be written as:

$$x = (L_{BC} + L_{AB}) - s \qquad (1)$$

Where:

- s – position from crank center to piston pin center.
- L_{BC} –connecting rod length.
- L_{AB} – crank radius.
- x – instantaneous piston position from piston pin to TDC (Top Dead Center).

With we can say that:

$$s = (L_{BC}\ cos\, q2 + L_{AB}\, cos\, q1) \qquad (2)$$

$$L_{BC}\ sin\, q2 = L_{AB}\, sin\, q1 \qquad (3)$$

From equation 3 we can rearrange in angle q2 form:

$$sin\, q2 = \frac{(L_{AB})}{L_{BC}} sin\, q1 \tag{4}$$

$$cos\, q2 = \sqrt{1 - (\frac{L_{AB}}{L_{BC}})\text{^}2\ sin^2\, q1} \tag{5}$$

Simplifying further we get that:

$$cos\, q2 = 1 - \frac{\left(\frac{L_{AB}}{L_{BC}}\right)^2\ sin^2\, q1}{2} \tag{6}$$

Rearranging this we can say that:

$$1 - cos\, q2 = \frac{\left(\frac{L_{AB}}{L_{BC}}\right)^2\ sin^2\, q1}{2} \tag{7}$$

Expressing the equation in terms of an angle q1 which is the crank angle. Therefore, Substitute equation (2) in equation (1) provides:

$$x = (L_{BC} + L_{AB}) - (L_{BC}\ cos\, q2 + L_{AB}\, cos\, q1) \tag{8}$$

Simplifying we get:

$$x = L_{BC}(1 - cos\, q2) + L_{AB}(1 - cos\, q1) \tag{9}$$

Substituting in equation (7) into equation (9):

$$x = L_{BC}\left(\frac{\left(\frac{L_{AB}}{L_{BC}}\right)^2\ sin^2\, q1}{2}\right) + L_{AB}(1 - cos\, q1) \tag{10}$$

ii. Velocity Equations:

The velocity of point B (particle C) with respect to point A (center of the crankshaft) can be expressed as follows:

$$\overrightarrow{V_B} = \overrightarrow{w_B} \times \overrightarrow{r_{\frac{A}{B}}} \quad (11)$$

where:

- $\boldsymbol{V_B}$ is the velocity of point B with respect to point A.
- $\boldsymbol{\omega_B}$ is the angular velocity of link AB (crankshaft).
- $\boldsymbol{r_{A/B}}$ is the position vector from point A to point B.

The angular velocity $\boldsymbol{\omega_B}$ is related to the generalized speed U1 by $\boldsymbol{\omega_B}$=$U1$.

Similarly, the velocity of point C with respect to point B can be expressed as:

$$\overrightarrow{V_C} = \overrightarrow{V_B} + \overrightarrow{w_{BC}} \times \overrightarrow{r_{\frac{C}{B}}} \quad (12)$$

Where:

- $\boldsymbol{V_C}$ is the velocity of point B with respect to point A.
- $\boldsymbol{\omega_{BC}}$ is the angular velocity of link AB (crankshaft).
- $\boldsymbol{r_{C/B}}$ is the position vector from point B to point C.

ω_{BC} is related to the generalized speed U2.

iii. Acceleration Equations:

The acceleration of particle B with respect to point A is given by:

$$\overrightarrow{a_B} = -\omega_B^2 \overrightarrow{r_{\frac{A}{B}}} \quad (13)$$

where $\boldsymbol{a_B}$ is the acceleration of point B with respect to the Newtonian frame, and $\boldsymbol{\alpha_{AB}}$ is the angular acceleration of link AB.

The angular acceleration α_{AB} is related to the generalized acceleration $\dot{U}1$ by $\alpha_{AB}=\dot{U}1$.

Similarly, the acceleration of point C with respect to point B is given by:

$$\overrightarrow{a_c} = \overrightarrow{a_b} - \omega_{BC}^2 \overrightarrow{r_{\frac{C}{B}}} + \overrightarrow{\propto_{BC}} \times \overrightarrow{r_{\frac{C}{B}}} \quad (14)$$

where α_{BC} is related to the generalized acceleration $\dot{U}2$.

iv. Transformation Matrices:

With the coordinate transformation between $\widehat{b_1}, \widehat{b_2}, \widehat{b_3}$ and $\widehat{a_1}, \widehat{a_2}, \widehat{a_3}$ we can show it with this table 1.

Table 1. Coordinate Transformation from frame B to frame A

	$\widehat{a_1}$	$\widehat{a_2}$	$\widehat{a_3}$
$\widehat{b_1}$	cos q_1	-sin q_1	0
$\widehat{b_2}$	sin q_1	cos q_1	0
$\widehat{b_3}$	0	0	1

The instant velocity of the particle B:

$$\overrightarrow{V_B} = \omega \frac{L_{AB}}{L_{BC}} cos\, q1 \quad (15)$$

And the acceleration of particle B:

$$\overrightarrow{a_B} = -\omega_B^2 \frac{L_{AB}}{L_{BC}} sin\, q1 \quad (16)$$

Piston instantaneous velocity of particle C using equation is the first derivative of equation (11)

$$\overrightarrow{V_{insC}} = L_{AB}\omega\left(\frac{\frac{L_{AB}}{L_{BC}}\ sin\ 2\,q1}{2} + sin\,q1\right) \quad (17)$$

Piston Instantaneous Acceleration. The instantaneous acceleration of the piston is done the same as in instantaneous velocity did.

$$\overrightarrow{a_{insC}} = L_{AB}\omega^2\left(\frac{L_{AB}}{L_{BC}}\ sin\ 2\,q1 + cos\,q1\right) \quad (18)$$

Piston Pin Position. The displacement of the piston with respect to crank angle can be derived from simple trigonometry. This can then be differentiated to yield velocity and acceleration of the piston. The piston pin position is the position from crank center to the piston pin center and can be formulated from cosine rule of the trigonometry.

$$L_{BC} = L_{AB}^2 + s^2 - 2L_{AB}s\,cos\,q1 \quad (19)$$

Piston Pin Velocity. Piston pin velocity is the upward velocity from crank center along cylinder bore center and can be calculated as the first derivative of equation (19)

$$\overrightarrow{V_C} = -L_{AB}\,sin\,q1 - \frac{L_{AB}^2\,sin\,q1\,cos\,q1}{\sqrt{L_{BS}^2 - L_{AB}^2\,sin^2\,q1}} \quad (20)$$

Piston Pin Acceleration. Piston pin acceleration is the upward acceleration from crank center along cylinder bore center and can be calculated as the second derivative of equation (19) with respect to angle q1.

$$\overrightarrow{a_C} = -L_{AB}\,cos\,q1 - \frac{L_{AB}^2\,(cos^2\,q1 - sin^2\,q1)}{\sqrt{L_{BS}^2 - L_{AB}^2\,sin^2\,q1}} - \frac{(L_{AB}^2)^2\ (cos^2\,q1\,sin^2\,q1)}{\sqrt{L_{BS}^2 - L_{AB}^2\,sin^2\,q1}} \quad (21)$$

v. Forces acting on the system equations.

Gravity Force: The force due to gravity acts vertically downward on the piston. Its magnitude is given by $F_g = m * g$, where m is the mass of the piston.

Inertial Force: This force arises due to the acceleration of the mass m on the piston. It can be calculated where $F_{inertial} = m * a_c$ where a_c is the linear acceleration of the mass.

Now, the tangential component of the crank force is given by $F_{tangential} = m * L_{Ab} * \propto_{BC}$

$$F_{net} = F_{inertial} - F_{tangential} \quad (22)$$

This can be simplified to

$$0 = -m * L_{AB} * \propto_{BC} \quad (23)$$

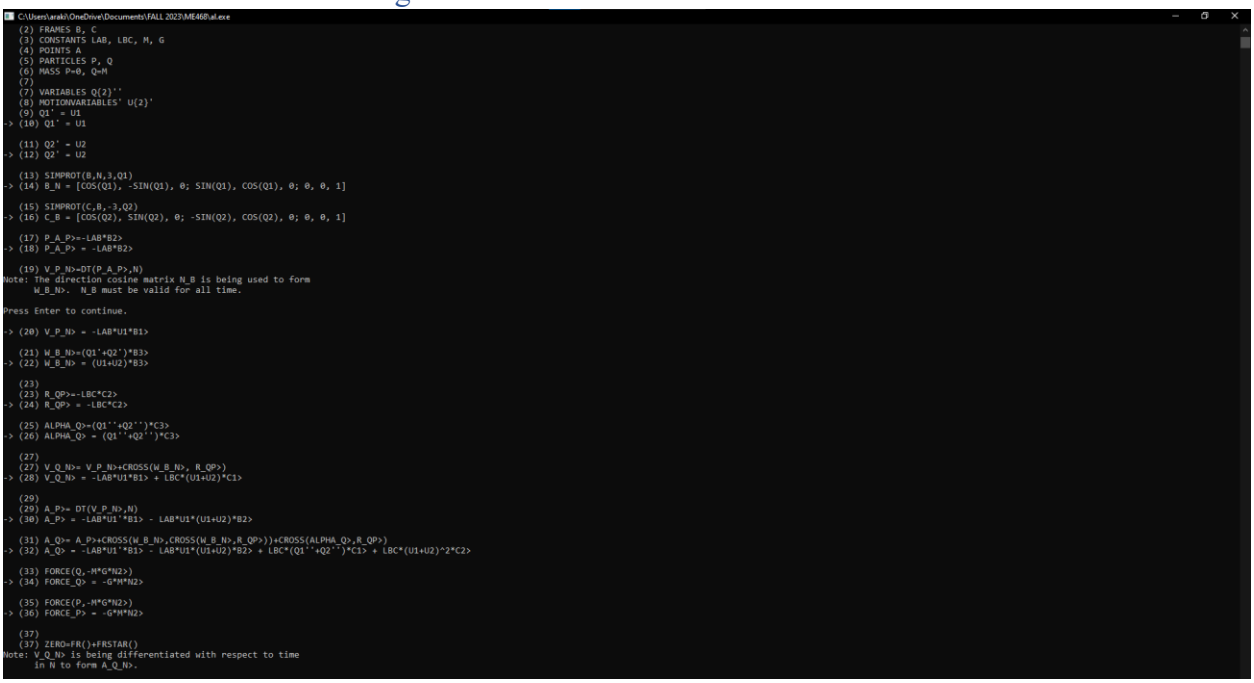
This equation indicates that the net force in the vertical direction is equal to the negative of the tangential force in the crank. This implies that the vertical motion is solely influenced by the tangential component of the crank force.

It's important to note that this analysis assumes ideal conditions and neglects friction and other dissipative forces. Real-world systems may require more complex models to account for these factors.

C. Simulation Results and Analysis

The subsequent section of the paper will present and analyze the results obtained from the simulation, offering insights into the behavior of the piston-crankshaft system.

a. AutoLev Coding



```
Press Enter to continue.

-> (38) ZERO[1] = M*(G*LBC*SIN(Q1-Q2)-2*G*LAB*SIN(Q1)-2*LAB*LBC*SIN(Q2)*U2*(U1+U2)-LBC*(LBC-LAB*COS(Q2))*U2'-(LAB^2+LBC^2-2*LAB*LBC*COS(Q2))*U1')

-> (39) ZERO[2] = LBC*M*(G*SIN(Q1-Q2)+LAB*SIN(Q2)*U1*(U1+U2)-LBC*U2'-(LBC-LAB*COS(Q2))*U1')

   (40) KANE()
-> (41) ZERO[1] = M*(G*LBC*SIN(Q1-Q2)-2*G*LAB*SIN(Q1)-2*LAB*LBC*SIN(Q2)*U2*(U1+U2)-LBC*(LBC-LAB*COS(Q2))*U2'-(LAB^2+LBC^2-2*LAB*LBC*COS(Q2))*U1')

-> (42) ZERO[2] = LBC*M*(G*SIN(Q1-Q2)+LAB*SIN(Q2)*U1*(U1+U2)-LBC*U2'-(LBC-LAB*COS(Q2))*U1')

   (43) UNITS G=M/SEC^2, M=KG, T=SEC,Q1=DEG,Q2=DEG%,Q1=RAD/SEC
   (44) INPUT LAB=0.5, LBC=0.5, G=9.81, M=2
   (45) INPUT Q1=10, Q2=10, U1=0, U2=0
   (46) INPUT TFINAL=16, INTEGSTP=0.1, ABSERR=1.0E-07, RELERR=1.0E-07
   (47) OUTPUT T, Q1, Q2, U1, U2
   (48)
```

The MATLAB Code (in Appendix) was output with values:

INPUT LAB=0.5 m, LBC=0.5 m, G=9.81 m/sec^2, M=2 kg

INPUT Q1=10, Q2=10, U1=0, U2=0, (Q1 and Q2 being rad/s)

With a time of 16 seconds.

This creates the final analysis with graphs, Q1 vs time, Q2 vs time, U1 vs time, and U2 vs time. This shows the angle of rotation in the Crankshaft arm and the piston. This is shown in Figures 2-5 below.

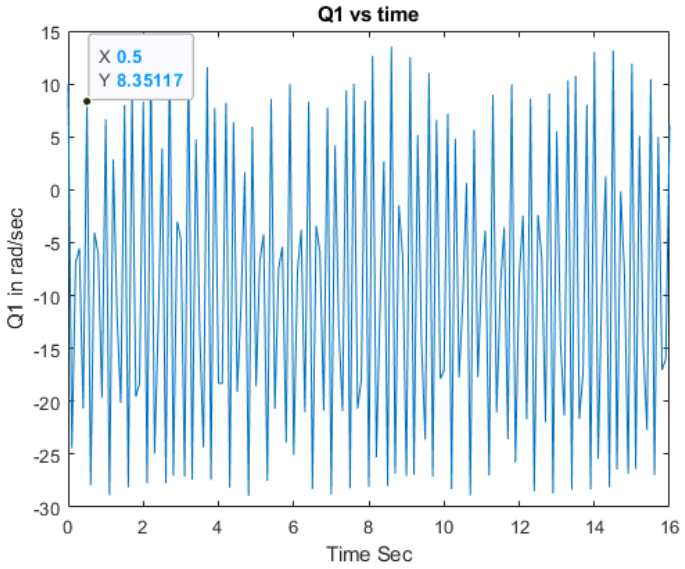


Figure 2. Q1 vs Time

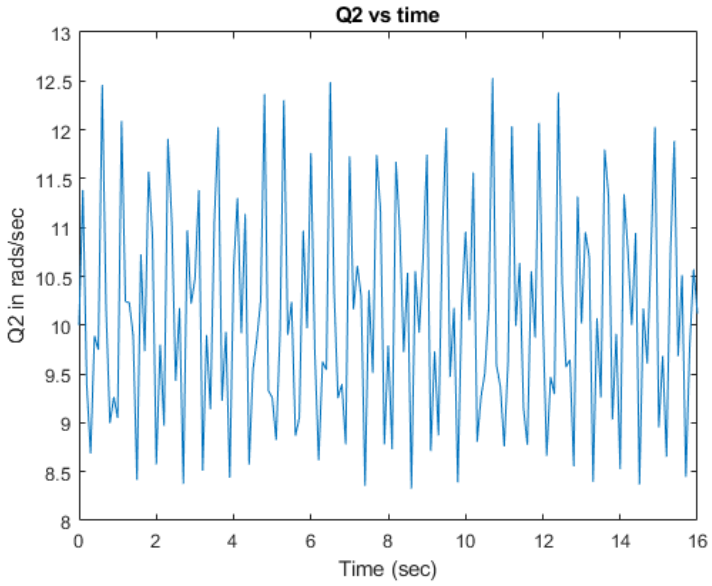


Figure 3. Q2 vs Time

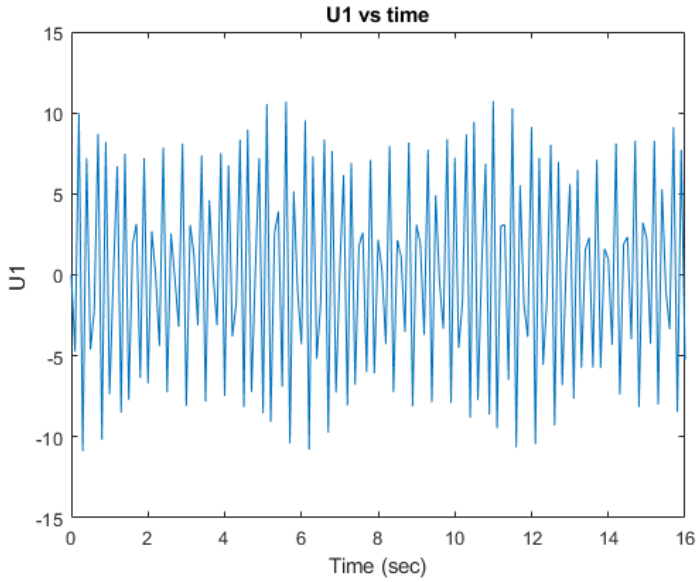


Figure 4. U1 vs Time

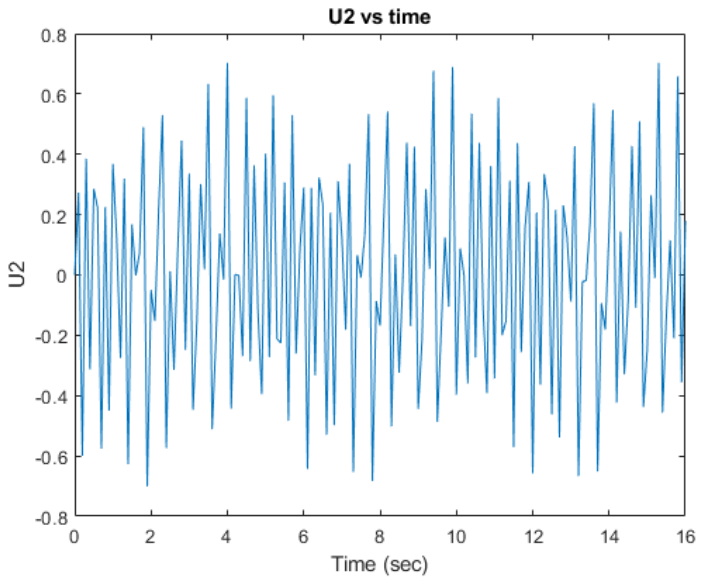


Figure 5. U2 vs Time

D. Conclusion

The integration of Autolev coding in conjunction with MATLAB has proven to be a highly successful approach for our comprehensive analysis of the piston-crankshaft system. Through meticulous modeling and simulation, we have successfully translated theoretical concepts into practical insights, culminating in informative MATLAB graphs that vividly depict the system's behavior over time.

The graphical representations of displacement over time provide a clear and dynamic visualization of the piston's motion within the crankshaft system. These graphs not only validate the accuracy of our Autolev coding but also offer a tangible representation of the complex interplay between the piston and crankshaft components.

Furthermore, the MATLAB-generated graphs illustrating velocities and accelerations at various points within the system are instrumental in demonstrating the robustness of our analytical framework. The nuanced details of these dynamic parameters showcase the intricate dynamics of the piston-crankshaft system and affirm the success of our simulation.

By successfully incorporating Autolev coding into the MATLAB environment, we have not only achieved a comprehensive understanding of the piston-crankshaft system's behavior but have

also demonstrated the efficacy of our approach in generating practical and insightful results. This success underscores the potential of our methodology in contributing valuable insights to the optimization of internal combustion engines and the broader application of multibody modeling and virtual prototyping in the realm of mechanical engineering.

References

[1] Kuang-Hua Chang, Design Theory and Methods Using CAD/CAE, Academic Press, 2015, Page xvii, ISBN 9780123985125, https://doi.org/10.1016/B978-0-12-398512-5.11001-1.

Appendix

```
function Team3Project2
SolveOrdinaryDifferentialEquations
% File  Team3Project2.m  created by Autolev 4.1 on Wed Nov 22 19:05:41 2023


%=========================================================================
function VAR = ReadUserInput
global   G LAB LBC Q1pp Q2pp;
global   Q1 Q2 U1 U2;
global   Q1p Q2p U1p U2p;
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

%------------------------------+--------------------------+-------------------+-----
------------
% Quantity                     | Value                    | Units             |
Description
%------------------------------|--------------------------|-------------------|-----
------------
G                              =  9.81;                   % M/SEC^2
Constant
LAB                            =  0.5;                    % UNITS
Constant
LBC                            =  0.5;                    % UNITS
Constant
Q1pp                           =  0.0;                    % UNITS
Constant
Q2pp                           =  0.0;                    % UNITS
Constant

Q1                             =  10;                     % DEG
Initial Value
Q2                             =  10;                     % DEG
Initial Value
U1                             =  0;                      % UNITS
Initial Value
U2                             =  0;                      % UNITS
Initial Value

TINITIAL                       =  0.0;                    % UNITS
Initial Time
TFINAL                         =  16;                     % UNITS
Final Time
```

```
INTEGSTP                        =  0.1;                  % UNITS
Integration Step
PRINTINT                        =  1;                    % Positive Integer
Print-Integer
ABSERR                          =  1.0E-07;              %
Absolute Error
RELERR                          =  1.0E-07;              %
Relative Error
%-------------------------------+-------------------------+-------------------+-----
------------

% Unit conversions
Pi       = 3.141592653589793;
DEGtoRAD = Pi/180.0;
RADtoDEG = 180.0/Pi;
Q1 = Q1*DEGtoRAD;
Q2 = Q2*DEGtoRAD;

% Reserve space and initialize matrices
COEF = zeros(2,2);
RHS = zeros(1,2);

% Evaluate constants
% Set the initial values of the states
VAR(1) = Q1;
VAR(2) = Q2;
VAR(3) = U1;
VAR(4) = U2;



%=========================================================================
function OpenOutputFilesAndWriteHeadings
FileIdentifier = fopen('Team3Project2.1', 'wt');   if( FileIdentifier == -1 )
error('Error: unable to open file Team3Project2.1'); end
fprintf( 1,             '%%       T             Q1             Q2             U1
U2\n' );
fprintf( 1,             '%%     (SEC)          (DEG)          (DEG)         (UNITS)
(UNITS)\n\n' );
fprintf(FileIdentifier, '%% FILE: Team3Project2.1\n%%\n' );
fprintf(FileIdentifier, '%%       T             Q1             Q2             U1
U2\n' );
fprintf(FileIdentifier, '%%     (SEC)          (DEG)          (DEG)         (UNITS)
(UNITS)\n\n' );



%=========================================================================
% Main driver loop for numerical integration of differential equations
%=========================================================================
function SolveOrdinaryDifferentialEquations
global   G LAB LBC Q1pp Q2pp;
global   Q1 Q2 U1 U2;
global   Q1p Q2p U1p U2p;
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
```

```
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

OpenOutputFilesAndWriteHeadings
VAR = ReadUserInput;
OdeMatlabOptions = odeset( 'RelTol',RELERR, 'AbsTol',ABSERR, 'MaxStep',INTEGSTP );
T = TINITIAL;
PrintCounter = 0;
mdlDerivatives(T,VAR,0);
while 1,
  if( TFINAL>=TINITIAL & T+0.01*INTEGSTP>=TFINAL ) PrintCounter = -1; end
  if( TFINAL<=TINITIAL & T+0.01*INTEGSTP<=TFINAL ) PrintCounter = -1; end
  if( PrintCounter <= 0.01 ),
     mdlOutputs(T,VAR,0);
     if( PrintCounter == -1 ) break; end
     PrintCounter = PRINTINT;
  end
  [TimeOdeArray,VarOdeArray] = ode45( @mdlDerivatives, [T T+INTEGSTP], VAR,
OdeMatlabOptions, 0 );
  TimeAtEndOfArray = TimeOdeArray( length(TimeOdeArray) );
  if( abs(TimeAtEndOfArray - (T+INTEGSTP) ) >= abs(0.001*INTEGSTP) )
warning('numerical integration failed'); break;  end
  T = TimeAtEndOfArray;
  VAR = VarOdeArray( length(TimeOdeArray), : );
  PrintCounter = PrintCounter - 1;
end
mdlTerminate(T,VAR,0);



%==========================================================================
% mdlDerivatives: Calculates and returns the derivatives of the continuous states
%==========================================================================
function sys = mdlDerivatives(T,VAR,u)
global   G LAB LBC Q1pp Q2pp;
global   Q1 Q2 U1 U2;
global   Q1p Q2p U1p U2p;
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

% Update variables after integration step
Q1 = VAR(1);
Q2 = VAR(2);
U1 = VAR(3);
U2 = VAR(4);
Q1p = U1;
Q2p = U2;

COEF(1,1) = 2*LAB*LBC*cos(Q2) - LAB^2 - LBC^2;
COEF(1,2) = -LBC*(LBC-LAB*cos(Q2));
COEF(2,1) = LAB*cos(Q2) - LBC;
COEF(2,2) = -LBC;
RHS(1) = 2*G*LAB*sin(Q1) + 2*LAB*LBC*sin(Q2)*U2*(U1+U2) - G*LBC*sin(Q1-Q2);
RHS(2) = -G*sin(Q1-Q2) - LAB*sin(Q2)*U1*(U1+U2);
VARp(1) = COEF(1,1)*COEF(2,2) - COEF(1,2)*COEF(2,1);
SolutionToAxEqualsB(2) =(COEF(1,1)*RHS(2)-COEF(2,1)*RHS(1))/VARp(1);
```

```
SolutionToAxEqualsB(1) =(COEF(2,2)*RHS(1)-COEF(1,2)*RHS(2))/VARp(1);

% Update variables after uncoupling equations
U1p = SolutionToAxEqualsB(1);
U2p = SolutionToAxEqualsB(2);

% Update derivative array prior to integration step
VARp(1) = Q1p;
VARp(2) = Q2p;
VARp(3) = U1p;
VARp(4) = U2p;

sys = VARp';



%=============================================================================
% mdlOutputs: Calculates and return the outputs
%=============================================================================
function Output = mdlOutputs(T,VAR,u)
global   G LAB LBC Q1pp Q2pp;
global   Q1 Q2 U1 U2;
global   Q1p Q2p U1p U2p;
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

% Evaluate output quantities
Output(1)=T;  Output(2)=(Q1*RADtoDEG);  Output(3)=(Q2*RADtoDEG);  Output(4)=U1;
Output(5)=U2;
FileIdentifier = fopen('all');
WriteOutput( 1,                Output(1:5) );
WriteOutput( FileIdentifier(1), Output(1:5) );



%=============================================================================
function WriteOutput( fileIdentifier, Output )
numberOfOutputQuantities = length( Output );
if numberOfOutputQuantities > 0,
  for i=1:numberOfOutputQuantities,
    fprintf( fileIdentifier, ' %- 14.6E', Output(i) );
  end
  fprintf( fileIdentifier, '\n' );
end



%=============================================================================
% mdlTerminate: Perform end of simulation tasks and set sys=[]
%=============================================================================
function sys = mdlTerminate(T,VAR,u)
FileIdentifier = fopen('all');
fclose( FileIdentifier(1) );
fprintf( 1, '\n Output is in the file Team3Project2.1\n' );
```

```
fprintf( 1, '\n To load and plot columns 1 and 2 with a solid line and columns 1 and
3 with a dashed line, enter:\n' );
fprintf( 1, '    someName = load( ''Team3Project2.1'' );\n' );
fprintf( 1, '    plot( someName(:,1), someName(:,2), ''-'', someName(:,1),
someName(:,3), ''--'' )\n\n' );
sys = [];



%=============================================================================
% Sfunction: System/Simulink function from standard template
%=============================================================================
function [sys,x0,str,ts] = Sfunction(t,x,u,flag)
switch flag,
  case 0,  [sys,x0,str,ts] = mdlInitializeSizes;    % Initialization of sys, initial
state x0, state ordering string str, and sample times ts
  case 1,  sys = mdlDerivatives(t,x,u);             % Calculate the derivatives of
continuous states and store them in sys
  case 2,  sys = mdlUpdate(t,x,u);                  % Update discrete states x(n+1)
in sys
  case 3,  sys = mdlOutputs(t,x,u);                 % Calculate outputs in sys
  case 4,  sys = mdlGetTimeOfNextVarHit(t,x,u);     % Return next sample time for
variable-step in sys
  case 9,  sys = mdlTerminate(t,x,u);               % Perform end of simulation tasks
and set sys=[]
  otherwise error(['Unhandled flag = ',num2str(flag)]);
end



%=============================================================================
% mdlInitializeSizes: Return the sizes, initial state VAR, and sample times ts
%=============================================================================
function [sys,VAR,stateOrderingStrings,timeSampling] = mdlInitializeSizes
sizes = simsizes;             % Call simsizes to create a sizes structure
sizes.NumContStates  = 4;     % sys(1) is the number of continuous states
sizes.NumDiscStates  = 0;     % sys(2) is the number of discrete states
sizes.NumOutputs     = 5;     % sys(3) is the number of outputs
sizes.NumInputs      = 0;     % sys(4) is the number of inputs
sizes.DirFeedthrough = 1;     % sys(6) is 1, and allows for the output to be a
function of the input
sizes.NumSampleTimes = 1;     % sys(7) is the number of samples times (the number of
rows in ts)
sys = simsizes(sizes);        % Convert it to a sizes array
stateOrderingStrings = [];
timeSampling         = [0 0]; % m-by-2 matrix containing the sample times
OpenOutputFilesAndWriteHeadings
VAR = ReadUserInput
```