

## **2006-936: SOLVING NONLINEAR GOVERNING EQUATIONS OF MOTION USING MATLAB AND SIMULINK IN FIRST DYNAMICS COURSE**

### **Ali Mohammadzadeh, Grand Valley State University**

ALI R. MOHAMMADZADEH is currently assistant professor of Engineering at School of Engineering at Grand Valley State University. He received his B.S. in Mechanical Engineering from Sharif University of Technology And his M.S. and Ph.D. both in Mechanical Engineering from the University of Michigan at Ann Arbor. His research area of interest is fluid-structure interaction.

### **Salim Haidar, Grand Valley State University**

SALIM M.HAIDAR is currently associate professor of Mathematics at Grand Valley State University. He received his B.S. in Mathematics with a Minor in Physics from St. Vincent College, and his M.S. and Ph.D. in Applied Mathematics from Carnegie-Mellon University. His research studies are in applied nonlinear analysis: partial differential equations, optimization, numerical analysis and continuum mechanics

## **Solving Nonlinear Governing Equations of Motion Using MATLAB and SIMULINK in First Dynamics Course**

### Overview

Students in first dynamics courses deal with some dynamical problems in which the governing equations of motion are simultaneous, second order systems of non-linear ordinary differential equations. There are no known quantitative methods or closed-form solution to these systems of non-linear differential equations. To teach students analytic methods and solution techniques to this category of dynamical problems, the authors devised a model example in which the motion of a particle on a rough cylindrical surface is considered. The authors then formulate qualitative and quantitative solution methods and employ both MATLAB and SIMULINK to these analytic methods and solution techniques to arrive at the stable, numerical solution to the proposed example.

In the model example provided, the governing equations of motion are first obtained in the state form. Four different approaches then were adopted to arrive at the stable, numerical solution of the system of first order, simultaneous, non-linear differential equations of motion.

In the first approach, the governing differential equations were converted to appropriate difference equations, and then a program was written in MATLAB to solve the resulted system of nonlinear algebraic equations. The second approach employed fourth-order Runge-Kutta scheme by writing a program in MATLAB to render a stable solution to the system of differential equations. The third method utilized MATLAB built-in function, “ode45”, to solve the governing non-linear system of differential equations. Finally SIMULINK, which is an extension to MATLAB, was used to provide solutions to the governing differential equations. The results of these different approaches were then compared with each other.

Although MATLAB is recommended as the programming language for some end-of-the-chapter problems in the recent, well-known text by Tongue<sup>1</sup>, none of the popular dynamics texts in the market<sup>1, 2, 3, 4, 5</sup> today (including Tongue’s) use SIMULINK in any form or way. The authors believe that SIMULINK, being a graphical user interface program, has a great potential in promoting better understanding of dynamics subjects, especially when students do not have a differential equations course in their background. SIMULINK is also an excellent tool to reinforce the topics students learn in a typical, undergraduate differential equations course. Last but not least, SIMULINK is a very powerful tool in analysis and design of dynamical systems. The authors used SIMULINK in analysis and design of an automobile suspension system<sup>6</sup> as an exemplary model in vibrations’ class.

This model example, which provided for follow-up homework assignments and a project, helped students learn about efficient numerical methods, and how to employ technology tools, MATLAB and SIMULINK, in solving engineering problems, early in the dynamics class. What students learned here helps them a great deal in the subsequent courses in the curriculum. The state form of the governing differential equations of motion, introduced to students in the follow-up homework problems, is certainly subject of further application in studying the dynamical response of the systems in several subsequent courses, namely dynamic systems, vibrations, and control. Moreover, this treatment by state form of the governing equations is a novel and powerful way to treat differential equations in the classroom as all popular undergraduate differential equations texts in the market today miss out on it.

It should be mentioned that, at our institution, dynamics is taught in the first semester of the junior year and right after the differential equations class, which is offered in the second semester of the sophomore year. In other curriculums, where dynamics is taught before students take differential equations, the method applied here (in the SIMULINK part of this paper) becomes extremely valuable as it can be introduced and employed in the dynamics course before the differential equation class (See Appendix D). Appendix D clearly shows that SIMULINK integration block symbol requires no need for a priori knowledge of differential equations course. Nonetheless, SIMULINK still can be used in the differential equations course, as we did in ours.

The authors of this study received positive feedback from students regarding this experience. Students especially enjoyed using SIMULINK and expressed that this project help them a great deal in understanding not only dynamics topics but also the subjects covered in their previous differential equation course.

### Problem Statement

In our model example, we propose to evaluate the position, velocity and the time at which the 1 pound block leaves the surface of a cylindrical surface on which it slides. The block is assumed to have an initial velocity  $V_0$  at the top of the cylinder and is subject to a constraint friction force of kinetic coefficient of friction,  $\mu_k$  (See Figure 1). To achieve a stable numerical solution, we assume, without loss of generality, a specific initial speed of 10 ft/s for the block and consider the coefficient of kinetic friction between the block and surface to be zero in one case and 0.2 in the other. The radius of cylinder,  $r = 5$  ft.

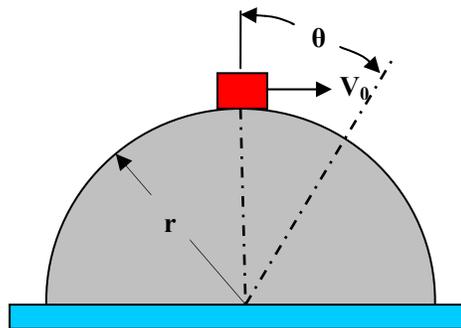


Figure 1

## Formulation

Figure 2 shows the free body and inertia response diagrams of the block,  $\theta$  degrees from the top of the cylinder.

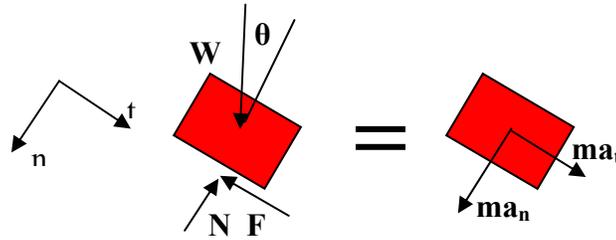


Figure 2

Adopting the path coordinate and applying Newton's second law of motion, one obtains:

$$\sum F_n = ma_n \quad : \quad W \cos \theta - N = m \frac{V^2}{r} \quad (1)$$

$$\sum F_t = ma_t \quad : \quad W \sin \theta - F = ma_t \quad (2)$$

Where  $m$  is the mass,  $W$  is the weight,  $V$  is the speed of the block  $\theta$  degree(s) from the top,  $r$  is the radius of the cylinder,  $N$  is the surface normal force,  $F$  is the surface friction force and  $a_n$  and  $a_t$  are the normal and tangential components of accelerations of the block, respectively. However:

$$F = \mu_k N \quad (3)$$

$$a_t = \frac{d^2 s}{dt^2} \quad (4)$$

Where  $s$  is the path traversed by the block on the cylindrical surface. Solving for  $N$  from equation (1), one obtains:

$$N = m \left( g \cos \theta - \frac{V^2}{r} \right) \quad (5)$$

Substituting  $N$  from (5) into (3), and the result together with (4) into (2), one gets:

$$\frac{d^2 s}{dt^2} = g(\sin \theta - \mu_k \cos \theta) + \mu_k \frac{V^2}{r} \quad (6)$$

The speed of the block can be expressed as:

$$V = \frac{ds}{dt} \quad (7)$$

Or:

$$V = r \frac{d\theta}{dt} \quad (8)$$

Substituting (7) into (6), and rearranging (8), one arrives at the state form of the governing equations of motion as:

$$\frac{ds}{dt} = V \quad (9)$$

$$\frac{dV}{dt} = g(\sin \theta - \mu_k \cos \theta) + \mu_k \frac{V^2}{r} \quad (10)$$

$$\frac{d\theta}{dt} = \frac{V}{r} \quad (11)$$

and subject to the initial conditions:

$$s(0) = 0.$$

$$V(0) = V_0 = 10 \text{ ft / s} \quad (12)$$

$$\theta(0) = 0.$$

The problem at hand is clearly a single degree of freedom autonomous system (since  $s = r\theta$ ) and, therefore, it should be governed by two first order state equations. However, we formulated the problem in the current manner by using equations 9-11 to obtain numerical solutions separately for both angular and curvilinear positions of the block. The block leaves the cylindrical surface when there is no contact with it ( $N = 0$ ) and, at the same time, when the rate of change of the normal force with respect to  $\theta$  is negative. When  $N = 0$ , equation (5) becomes:

$$g \cos \theta - \frac{V^2}{r} = 0 \quad (13)$$

Equations (9), (10), (11) form a set of nonlinear simultaneous differential equations of motion, in state form, with initial conditions (12) and subject to condition (13) on angle  $\theta$ , at which the block leaves the surface of the cylinder.

### Solution Methods

A closed-form solution to the system of non-linear, time-dependent, ordinary differential equations (9), (10), (11) subject to initial conditions (12) and constraint equation (13) is not possible. Three different numerical approaches are employed in this study to calculate the angle, velocity, position and time at which the block leaves the surface of the cylinder; namely, an appropriate difference equations scheme, Runge-Kutta method, and a MATLAB built-in function ode45. This is the first time that the students try to use numerical methods to solve system of differential equations in the curriculum. Pedagogically, it is very useful if students try different

solution approaches and compare the results of these different ways. We also introduced students to SIMULINK to teach them a fourth approach to obtain a stable solution to the problem at hand, in particular, and to this category of simultaneous non-linear differential equations in general.

#### Approach I- Difference Equations Scheme

In this approach we approximate the variables and their derivatives in equations (9), (10), and (11) as follows:

$$\begin{aligned}\frac{ds}{dt} &\approx \frac{s(t) - s(t - \Delta t)}{\Delta t} \\ \frac{dV}{dt} &\approx \frac{V(t) - V(t - \Delta t)}{\Delta t} \\ \frac{d\theta}{dt} &\approx \frac{\theta(t) - \theta(t - \Delta t)}{\Delta t}\end{aligned}\quad (14)$$

One also approximates the block speed as the average speed value at times  $t$  and  $t + \Delta t$  as:

$$V \approx \frac{V(t - \Delta t) + V(t)}{2} \quad (15)$$

Substituting (14) and (15) into (9), (10), and (11), one obtains:

$$\begin{aligned}\frac{s(t) - s(t - \Delta t)}{\Delta t} &= \frac{V(t) + V(t - \Delta t)}{2} \\ \frac{V(t) - V(t - \Delta t)}{\Delta t} &= g(\sin \theta(t - \Delta t) - \mu_k \cos \theta(t - \Delta t)) + \mu_k \frac{(V(t - \Delta t))^2}{r} \\ \frac{\theta(t) - \theta(t - \Delta t)}{\Delta t} &= \frac{V(t) + V(t - \Delta t)}{2r}\end{aligned}\quad (16)$$

Solving for the variables, velocity, position and angle  $\theta$  at time  $t$  from equations (16), one arrives at the governing difference equations:

$$\begin{aligned}V(t) &= V(t - \Delta t) + \Delta t \left[ g(\sin \theta(t - \Delta t) - \mu_k \cos \theta(t - \Delta t)) + \mu_k \frac{(V(t - \Delta t))^2}{r} \right] \\ s(t) &= s(t - \Delta t) + \Delta t \left( \frac{V(t) + V(t - \Delta t)}{2} \right) \\ \theta(t) &= \theta(t - \Delta t) + \Delta t \left( \frac{V(t) + V(t - \Delta t)}{2r} \right)\end{aligned}\quad (17)$$

The above system of algebraic equations, subject to initial conditions (12), is then solved by marching through time from initial  $t = 0$ , when the block was launched at the top of the cylinder, until the time  $t$  when the block leaves the cylinder surface. The angle  $\theta$  and the velocity  $V$  are substituted into equation (13) at the end of each time step to check whether the block has lost contact with the surface or not. Without going further into the details, we mentioned to our students the importance of choosing appropriate time steps to obtain a stable solution. A time step of  $\Delta t = 0.1$  ms was adopted in this simulation. Appendix A represent the MATLAB<sup>®</sup> program used to implement this approach. Although the error of these first order difference operators is  $O(\Delta t)$ , their numerical stability is much better than the higher order ones.

#### Approach II- Fourth-Order Runge-Kutta Method

A fourth-Order Runge-Kutta<sup>7</sup> solution technique is employed to solve the system of non-linear time-dependent first-order differential equations (9, 10, 11), subject to the initial conditions (12) and condition (13) for evaluating the kinematics state of the block at the time of loss of contact with the cylindrical surface. Appendix B shows the detailed MATLAB<sup>®</sup> program to perform this integration.

#### Approach III- MATLAB<sup>®</sup> Built-in Function ode45

The MATLAB<sup>®</sup> built-in function ode45<sup>8</sup>, which is an implicit implementation of fourth-order Runge-Kutta method, is used to solve the system of non-linear differential equations, (9), (10), and (11), subject to the initial conditions (12) and condition (13). The detail of calling this function in MATLAB<sup>®</sup> is also shown in Appendix C.

#### Approach IV- SIMULINK Solution

SIMULINK<sup>®</sup>, which is an extension to MATLAB<sup>®</sup>, provides its users with a graphical user interface that is fun to try and much easier to use than MATLAB<sup>®</sup> itself (or Maple<sup>®</sup>) or any other traditional command-line programs, such as C, FORTRAN or BASIC. It is used to model complex nonlinear systems, with a relative ease in comparison with these command-line software programs, and as a result boosts productivity in a significant way.

To employ SIMULINK in solving system of nonlinear simultaneous ordinary differential equations it is not necessary to convert the system into the state form. Contrary to the above discussed approaches, we did not use the state form of the equations of motion (Equations 9, 10, and 11), rather we remodeled our problem using equations (6) (a second order ODE) and (11).

Appendix D shows the detailed SIMULINK<sup>®</sup> model of the problem at hand. It is seen in the SIMULINK<sup>®</sup> model that the left hand side of equation (6), namely, the term  $(\frac{d^2 s}{dt^2})$ , after going through two integrator blocks (1/s), renders the position of the block. Further elaboration on this model is presented in the result and discussion part of this document.

## Results and Discussions

To check the result of simulation, we study the exact solution to the governing differential equations of motion for the case of no friction ( $\mu_k = 0$ ). In that case, governing differential equations of motion reduce to:

$$\frac{dV}{dt} = g \sin \theta \quad (18)$$

$$\frac{d\theta}{dt} = \frac{V}{r} \quad (19)$$

Subject to initial conditions (12) and condition (13). These can be solved by elementary techniques. Using equation (19) and the chain rule, equation (18) is written as:

$$\frac{dV}{dt} = \frac{dV}{d\theta} \frac{d\theta}{dt} = \frac{V}{r} \frac{dV}{d\theta} = g \sin \theta \quad (20)$$

Which upon separation of variables and integration we obtain:

$$V^2 = V_0^2 + 2gr(1 - \cos \theta) \quad (21)$$

Upon substituting V from condition (13) into (21) we arrive at the angle at which the block leaves the surface of the cylinder:

$$\theta = \cos^{-1} \left( \frac{V_0^2 + 2gr}{3gr} \right) = \cos^{-1} \left( \frac{100 + 2(32.2)(5)}{3(32.2)(5)} \right) = 29.1078^\circ \quad (22)$$

Which upon substitution of (22) into (13), we obtain:

$$V = \sqrt{gr \cos \theta} = \sqrt{32.2 \times 5 \times \cos(29.1078^\circ)} = 11.8603 \text{ ft/s}$$

We allotted three 50-minutes class periods to acquaint our students with numerical methods such as Runge-Kutta, finite difference, and state form of differential equations. Moreover, one of these class periods was devoted to introducing students to SIMULINK by showing them how to construct simulation model for the problem at hand. Later on in the semester, students used MATLAB and SIMULINK to do homework assignments and complete a project that was given to them to reinforce the ideas and methods presented in this paper. We would also like to point out that students who go on to take vibrations, system and control, or differential equations courses will certainly reap great deal of benefit from this first experience in the dynamics class. Our students who took these subsequent courses further acknowledge this point.

Table 1 next compares all the above cases with the exact solution. As it is evident from the table all approaches clearly predict the kinematics values, for the exact solution, with excellent accuracy,  $O(10^{-4})$ .

TABLE 1

Kinematics Properties of the Block When it leaves the Cylindrical Surface  
No Friction Case ( $\mu_k = 0.$ )

| Approach                 | Time (s) | Angle $\theta$ (deg) | Position (ft) | Velocity (ft/s) |
|--------------------------|----------|----------------------|---------------|-----------------|
| Finite Difference        | 0.2393   | 29.1095              | 2.5403        | 11.8598         |
| Fourth order Runge-Kutta | 0.2393   | 29.1106              | 2.5404        | 11.8606         |
| MATLAB ode45 Function    | 0.2393   | 29.1107              | 2.5404        | 11.8606         |
| SIMULINK Model           | 0.2392   | 29.1069              | 2.5400        | 11.8603         |
| Exact Solution           | —        | 29.1078              | 2.5401        | 11.8603         |

Table 2 shows the results for the friction case, where the kinetic friction coefficient  $\mu_k = 0.4$ . Again all the approaches renders basically same values. The differences are mainly due to numerical truncation error.

Appendix D shows the simulation of the exercise for  $\mu_k = 0.4$ . The kinematical values that are observed in Table 2 for the SIMULINK model are values taken from the Display Blocks in Appendix D. The Stop Block used in the model is to terminate the simulation when condition (13) is met. The Logic Block seen in the model is used to implement condition (13). The model uses three integrator blocks to arrive at position, velocity and the angle  $\theta$  for the block. The Clock Block on top of the model keeps track of the simulation time. These visual blocks, taken from block diagram concept<sup>9</sup>, aid students a great deal to simulate a dynamical system in a relatively short period of time in comparison with a typical command-line program. As we mentioned before, there is also no need to convert the second order differential equations of motion into first order (state form), to arrive at a stable numerical solution, in a SIMULINK model.

TABLE 2

Kinematics Properties of the Block When it leaves the Cylindrical Surface  
No Friction Case ( $\mu_k = 0.4$ )

| Approach                 | Time (s) | Angle $\theta$ (deg) | Position (ft) | Velocity (ft/s) |
|--------------------------|----------|----------------------|---------------|-----------------|
| Finite Difference        | 0.2913   | 34.1204              | 2.9776        | 11.5456         |
| Fourth order Runge-Kutta | 0.2912   | 34.1093              | 2.9766        | 11.5453         |
| MATLAB ode45 Function    | 0.2912   | 34.1093              | 2.9766        | 11.5453         |
| SIMULINK Model           | 0.2912   | 34.1105              | 2.9774        | 11.5456         |

As it is seen from the results, friction holds the block to the cylinder for a longer time compared with the no friction case. This is expected as friction lowers the block velocity, which from equation (13) predicts a bigger contact angle  $\theta$ .

## Conclusion

It is the authors' belief that with the availability of powerful programming tools such as MATLAB and SIMULINK, the students of dynamics and differential equations benefit tremendously from integrating similar models, as the one in the above, in their course work. SIMULINK<sup>®</sup> software especially looks very promising in dynamics classes, where students do not have a differential equation course in their background.

This model lead to homework problems and projects which helped students learn how to solve a system of non-linear, time dependent, governing differential equations of motion in 4 different ways. It introduced students to the state form of these differential equations, a topic which they will deal with in their future course work in vibrations and control systems. It also sharpened our students math and programming skills by extending their differential equation knowledge to new dimensions; namely, stable numerical methods for solving nonlinear systems. SIMULINK also helped students to develop block diagram skills. A skill they use in their control course later.

Students' feedback regarding this model and its follow-up homework assignment and a project, was very positive. They indicated that the model, homework and project combination helped them a great deal to further understand the topics learned in the dynamics and differential equation classes. They enjoyed, in particular, programming with SIMULINK because of its user graphical interface character and relative ease to use.

## Bibliography

1. Tongue, B.H., and S.D., Sheppard: Dynamics: Analysis and Design of Systems in Motion, John Wiley, 2004.
2. Beer, F. P., and J. E. Russell: Vector Mechanics for Engineers- Dynamics, 6<sup>th</sup>. Edition, McGraw Hill, 2004.
3. Meriam, J.L., and L.G. Kraige: Engineering Mechanics- Dynamics, 5<sup>th</sup>. Edition, John Wiley, 2001.
4. Riley, W.F., and L.D. Sturges: Engineering Mechanics- Dynamics, 2<sup>nd</sup> Edition John Wiley, 1995.
5. Hibbeler, R.C.: Dynamics, 10<sup>th</sup> Edition, Prentice Hall, 2004.
6. Mohammadzadeh, A.R., and S. Haidar: Analysis and Design of Automotive Suspension System, ASEE Annual Proceedings, 2006.
7. Ayyub, B.M., and R.H. McCuen: Numerical Methods for Engineers, Prentice Hall, 1996.
8. Chapra, S.C.: Applied Numerical Methods with MATLAB for Engineers and Scientists, McGraw Hill, 2005.
9. Dabney J.B., and T.L. Harman: Mastering SIMULINK, Prentice Hall, 2004.

## APPENDIX A

### MATLAB File FiniteDifferenceFric.m for Finite Difference Approach

```
% Finite Difference Approach To solve ODE System of Equations
i=1;
% Specifying Constants
r=5; muk=0.4;dt = 0.0001;g=32.2;
% Creating Vectors
Theta= []; V = []; t= [];
% Initializing Vectors
V(i)=10; Theta(i)=0; t(i)=0; s(i)=0;
u=t(1);
% Testing Whether the Block Leaves the Surface and Evaluating
% The New Positions, Angle, and Velocity of the Block
while cos(Theta(i))-(V(i))^2/(r*g)>eps
    i= i+1;
    V(i) = V(i-1)+dt*(g*(sin(Theta(i-1)))-muk*cos(Theta(i-1)))+muk*(V(i-
1))^2/r);
    Theta(i)= Theta(i-1)+dt/(2*r)*(V(i)+V(i-1));
    s(i)=s(i-1)+ dt*(V(i)+V(i-1))/2;
    t(i)= u+(i-1)*dt;
end
% Evaluating the angle, Position, and velocity at which the block leaves
% the surface
Theta(i)=(Theta(i)+Theta(i-1))/2;
s2 = r*Theta(i);
Theta(i) = Theta(i)*180/pi;
V(i) = (V(i)+V(i-1))/2;
s(i) = (s(i)+s(i-1))/2;
% Printing the Results
fprintf('Time = %5.4f sec\n',t(i))
fprintf('Theta = %5.4f deg\n',Theta(i))
fprintf('Position= %5.4f ft\n',s(i))
fprintf('Position using r*Theta = %5.4f ft\n',s2)
fprintf('Velocity = %5.4f ft/s\n',V(i))
```

Running FiniteDifference.m for the case  $\mu_k = 0$ .

Time = 0.2393 sec  
Theta = 29.1095 deg  
Position= 2.5403 ft  
Position using  $r*\Theta$  = 2.5403 ft  
Velocity = 11.8598 ft/s

Running FiniteDifference.m for the case  $\mu_k = 0.4$

Time = 0.2913 sec  
Theta = 34.1204 deg  
Position= 2.9776 ft  
Position using  $r*\Theta$  = 2.9776 ft  
Velocity = 11.5456 ft/s

## APPENDIX B

### MATLAB File FrictionIntegRun to call on Runge-Kutta Integration Method

```
%%Provide the Initial Conditions
y0 =[0;0;10]; r=5; g=32.2;

%Call on Runge-Kutta Function to Perform Integration
[t,y]=rkgen('fric',[0 1],y0,0.0001);

%Test Wether the Block Has left the Surface
for i=1:10001
    s= cos(y(2,i))-((y(3,i))^2)/(r*g);
    if s<eps
        y(:,i)=(y(:,i)+y(:,i-1))/2;
        y(2,i)=y(2,i)*180/pi;

%Print the Result
fprintf('Time = %5.4f sec\n',t(i))
fprintf('Theta = %5.4f deg\n',y(2,i))
fprintf('Position = %5.4f ft\n',y(1,i))
fprintf('Velocity = %5.4f ft/s\n',y(3,i))
break
    end
end
```

### MATLAB Function rkgen to Implement Runge-Kutta Integration Method

```
% Fourth Order Runge Kutta Method for Solving Simultaneous first order
% Differential Equations

function[tvals,yvals]= rkgen(f,tspan,startval,step)

% Creating Coefficient Vectors
b=[];d=[];
b=[1/6 1/3 1/3 1/6]; d =[0 0.5 0.5 1];

% Indicating the Number of Time Steps and Initial values
steps = (tspan(2) -tspan(1))/step +1;
y=startval; t=tspan(1);
yvals=startval; tvals=tspan(1);

% Calculating k1, k2, k3, and k4
for j=2:steps
    k(1,:) = step*feval(f,t,y);
    for i=2:4
        if (i==2 | i==3)
            cc=0.5;
        else
            cc=1;
        end
        k(i,:)= step*feval(f, t+step*d(i),y+(cc*k(i-1,:))');
    end
end

% Updating Function Values and Time
```

```

y1 = y+(b*k)';
t1=t +step;
tvals=[tvals, t1]; yvals = [yvals, y1];
t = t1; y =y1;
end

```

MATLAB Function fric.m to provide derivatives to Runge-Kutta Integration Method

```

function ydot = fric(t,y)
muk=0.4; r=5; g=32.2;

% Provide the derivatives
ydot=[ y(3); y(3)/r; g*(sin(y(2))-muk*cos(y(2)))+muk*((y(3))^2)/r];

```

Running Runge-Kutta Approach for the case  $\mu_k = 0$ .

Time = 0.2393 sec  
Theta = 29.1106 deg  
Position = 2.5404 ft  
Velocity = 11.8606 ft/s

Running Runge-Kutta Approach for the case  $\mu_k = 0.4$

Time = 0.2912 sec  
Theta = 34.1093 deg  
Position = 2.9766 ft  
Velocity = 11.5453 ft/s

## APPENDIX C

### MATLAB File FrictionInteg.m to call on ode45 Integration Method

```
%Provide Time Span of Integration
for j=1:10000
    tspan(j)= 0+(j-1)*0.0001;
end

%Provide the Initial Conditions
y0 =[0;0;10]; r=5; g=32.2;

%Call on ode45 to Perform Integration
[t,y]=ode45(@fric,tspan,y0);

%Test Whether the Block Has left the Surface
for i=1:10000
    s= cos(y(i,2))-((y(i,3))^2)/(r*g);
    if s<eps
        y(i,:)=(y(i,:)+y(i-1,:))/2;
        y(i,2)=y(i,2)*180/pi;
    end
end

%Print the Result
fprintf('Time = %5.4f sec\n',t(i))
fprintf('Theta = %5.4f deg\n',y(i,2))
fprintf('Position = %5.4f ft\n',y(i,1))
fprintf('Velocity = %5.4f ft/s\n',y(i,3))
break
end
end
```

### MATLAB Function fric.m to provide derivatives to ode45 Integration Method

```
function ydot = fric(t,y)
muk=0.4; r=5; g=32.2;

% Provide the derivatives
ydot=[ y(3); y(3)/r; g*(sin(y(2))-muk*cos(y(2)))+muk*((y(3))^2)/r];
```

Running MATLAB built in Function ode45 Approach for the case  $\mu_k = 0$ .

Time = 0.2393 sec  
Theta = 29.1107 deg  
Position = 2.5404 ft  
Velocity = 11.8606 ft/s

Running MATLAB built in Function ode45 Approach for the case  $\mu_k = 0.4$

Time = 0.2912 sec  
Theta = 34.1093 deg  
Position = 2.9766 ft  
Velocity = 11.5454 ft/s

# APPENDIX D

## SIMULINK Model for $\mu_k = 0.4$

