Evaluation of Friction Models with a Haptic Interface

Andrew Gosline

December 15, 2001
## Contents

1 Introduction 3

2 Proposal 4
   2.1 Methodology ......................................................... 4
   2.2 Timeline .............................................................. 5
   2.3 Resources ............................................................ 5

3 Literature Survey 6
   3.1 Presentations of New Models ...................................... 6
   3.2 Implementation of Friction Models ............................... 7
   3.3 Survey Papers ........................................................ 7
   3.4 Conclusions .......................................................... 8

4 Experimental Apparatus 9

5 Implementation of Friction Models 11
   5.1 Karnopp Model ....................................................... 11
   5.2 Hayward Model ..................................................... 12
   5.3 Hollerbach Model ................................................... 13
   5.4 Code Details ........................................................ 13

6 Results 14
   6.1 Karnopp Model ....................................................... 14
   6.2 Hollerbach Model ................................................... 17
   6.3 Hayward Model ..................................................... 19

7 Conclusions 21
   7.1 Future Work .......................................................... 22
   7.2 Acknowledgements .................................................. 22

8 Appendix 25
   8.1 S-Function for Friction Models ................................... 25
   8.2 Apparatus - Full View .............................................. 31
List of Figures

4.1 The 3 DOF, Twin-Pantograph Haptic Interface. ............................................. 9
4.2 The Simulation Block Diagram ................................................................. 10

5.1 Velocity vs. Friction Force in the Karnopp Model .............................. 11
5.2 Velocity vs. Friction Force in the Karnopp Model, with Dead Band Modification. 12
5.3 The Virtual Spring Formed in the Hayward Model ............................... 12

6.1 Karnopp Results with No Deadband, DV = 0.01m/s .............................. 15
6.2 Karnopp Results with No Deadband, DV = 0.001m/s ............................. 15
6.3 Karnopp Results with Deadband, DV = 0.01m/s ..................................... 15
6.4 Hollerbach Results with Velocity Threshold 0.0001m/s ...................... 17
6.5 Hollerbach Results with Velocity Threshold = 0.0025m/s ..................... 17
6.6 Hayward Results with $z_{\text{max}} = 0.0001\text{m}$ ................................. 19
6.7 Hayward Results with $z_{\text{max}} = 0.001\text{m}$ ....................................... 19
Chapter 1

Introduction

Friction is an important physical phenomenon that contributes reactive force to virtually all mechanical systems. When simulating a mechanical system, it is important to include the effects of friction. There exist multiple mathematical models of friction in open literature, each of which has certain advantages and disadvantages. This project aims to determine how well certain models of friction are suited for use with haptic interfaces.

Haptic technology is a relatively new and exciting topic in robotics research. The addition of tactile feedback to computer interaction enables simulations to give the user a very rich experience, which can be very useful for instances where tele-operation is required.

The Robotics and Control Laboratory (RCL) in the Electrical and Computer Engineering Department has several twin-pantograph haptic interfaces with which friction simulation can be performed. These devices have been developed by RCL, and are a continuing topic of research. This offers an excellent opportunity to investigate the mathematical models of friction on a very new and versatile haptic device that has been locally developed.

3 Models of Friction were implemented into an existing haptics demo for comparison. The 3 models were the Karnopp, Hollerbach, and Hayward Models. No clear winner could be determined. The main conclusion that can be drawn from the research done is that these friction models are very sensitive to their characteristic parameters. As such, some models will excel over others in situations where it is easy or possible to control these parameters.
Chapter 2

Proposal

2.1 Methodology

The proposed project will consist of the following activities (milestones):

1. Literature survey - At least 3 computational models of friction will be collected from open literature. Each model will be critiqued on its advantages and disadvantages for simulation with a haptic interface.

2. Familiarization with the 3DOF planar twin-pantograph haptic interface - The related papers, reports, code, and hardware will be studied such that a demo of friction with this haptic device can be created. Particular attention will be paid to the calculation and feedback of force information so that each model of friction may be implemented into a simulation with this haptic device.

3. Implementation of Friction Models - Each of the chosen models will be written into a module of code such that it can be tested prior to incorporating it into a virtual environment with the haptic device.

4. Incorporation and testing of Friction models - Each of the chosen models will be incorporated into a simple virtual environment using the haptic interface. The aim of this virtual environment will be to investigate the benefits of using each friction model with the haptic device, and to compare each model’s ’feel’.

5. Graphics (Time permitting) - A friction map with graphical representation will be created.
2.2 Timeline

<table>
<thead>
<tr>
<th>Dates</th>
<th>Description of Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 18 - Nov 9</td>
<td>Literature survey and selection of Models of friction</td>
</tr>
<tr>
<td>Oct 18 - Nov 16</td>
<td>Familiarization with 3 DOF planar twin-pantograph haptic interface</td>
</tr>
<tr>
<td>Nov 2 - Nov 20</td>
<td>Coding of friction models into separate modules and testing</td>
</tr>
<tr>
<td>Nov 20</td>
<td>Submission of Progress Report</td>
</tr>
<tr>
<td>Nov 20 - Dec 7</td>
<td>Incorporation of friction model codes with haptic interface and testing</td>
</tr>
<tr>
<td>Dec 15</td>
<td>Submission of Report</td>
</tr>
</tbody>
</table>

2.3 Resources

The following resources will be required to perform this project:

1. 3 DOF planar twin-pantograph haptic interface - supplied by Dr. Tim Salcudean in MCLD 348.

2. Vx Works workstation - supplied by Dr. Tim Salcudean in MCLD 348.
Chapter 3

Literature Survey

A survey of the open literature regarding friction models and haptics was performed. In general, the prior work that was surveyed can be subgrouped into papers that introduce new friction models, papers that compare and contrast multiple friction models, and papers that discuss implementation of friction models to specific applications (e.g., haptics, or control).

3.1 Presentations of New Models

The "Classical" model of friction, that it is proportional to normal force and opposes motion, stems back to Leonardo da Vinci. According to Armstrong-Helouvry et al. [1], Da Vinci’s model was rediscovered by Amontons in 1699, and then developed by Coulomb in 1785. As used in the terminology today, Coulomb friction is described by a constant force that opposes motion for any non-zero velocity.

In 1968, Dahl introduced a model of solid friction in the form of a first order differential equation. In 1976, [5] he used this model in a mathematical study of a friction damper to an oscillatory system. His conclusion was that the model behaves like Coulomb friction at large amplitudes, and "structural damping" at low amplitudes.

In 1985, Karnopp [9] proposed what is now referred to as the Karnopp model of stick-slip friction. This model is mindful of the fact that it is rare for a digital computer to calculate a velocity of exactly zero. This model uses a small region surrounding zero velocity (−DV < V < DV) where the force is proportional to the velocity to model the stick region, and a constant Coulomb force in the slip region.

In 1991, Haessig et al. [7] introduced the Bristle and the Reset Integrator Models. These two new models were compared with 3 other friction models, namely the classical Coulomb, Dahl and Karnopp models with various simulations. Although the Bristle Model performed with the highest degree of accuracy, it also performed very sluggishly, taking the longest computational
time by at least a factor of two in the simulations. The authors concluded that the Karnopp, Dahl, and Reset Integrator were best suited for simulation of dynamic processes in which friction is a large factor. Additionally, they pointed out that the Dahl and Reset Integrator Models are application independent, and can be implemented with the relative velocity as the only input.

In 2000, Hayward et al. [8] presented a friction model for haptics that depends only on displacements and accounts for 2D or 3D vector motions. This model was tested experimentally with a consumer haptic interface. The results indicate that this model is well suited for implementation in even-based computer interactions, such as haptics. The model is given in a discrete time formulation that could be easily implemented into a haptic simulation. Additionally, Hayward stated that this model is drift free. This is an advantage when compared to the Dahl model, which drifts significantly.

3.2 Implementation of Friction Models

In 1994, Salcudean et al. [11] used a slightly modified version of the Karnopp [9] model on the UBC maglev haptic interface. The stick-slip nature of the Karnopp Model allowed the joystick to be positioned in a stable manner, which is very important for position controlled teleoperation and virtual environment manipulation.

In 1997, Tafazoli [2] used the model proposed by Caundas de Wit et al. [3] and a modified Karnopp [9] model to compensate for friction in a cartesian manipulator that was being used as the master in a teleoperated excavator setup with good results.

In 1998, Nahvi et al. [10] used a spring based stick-slip model of friction to display the friction characteristics of a human finger pad in a virtual environment. This model depends on velocity, transitioning from slip to stick when the velocity crosses the minimum velocity threshold, and transitioning back to slip when the spring reaches it’s rupture limit.

3.3 Survey Papers

In 1994 Armstrong-Helouvry et al. [1] performed a survey of models, analysis tools and compensation methods for friction in machines. This survey gives a detailed description of the mechanics of friction, including a detailed description of the stick-slip transition phenomenon know as the Stribeck or "negative viscous" effect. This effect appears to be most prevalent in situations where there is a lubrication layer between an otherwise metal to metal contact, and is very important for the friction modelling and compensation of hydraulic actuators. This paper also discusses the "Seven Parameter" Friction Model, which is a very complex model comprised of equations that describe the four velocity regimes.
3.4 Conclusions

A fairly in depth literature survey was performed on computational models of friction from which several conclusions can be drawn.

1. The Seven Parameter [1] and the Dahl [5] models are better suited to compensate for friction in a control problem, while the Karnopp [9], Hayward [8], spring based stick-slip [10], and reset-integrator [7] models are better suited to haptic rendering. The reason why the latter models are well suited for haptics is because they are depend only on velocity and/or position. This offers two benefits. Firstly, position and velocity are readily available parameters, requiring at most a single numerical differentation. This makes computation of friction cheap. Secondly, the models do not consist of PDEs or ODEs (the Dahl model is in the form of a first order ODE) and there is no need for numerical integration, which can lead to instability and inaccuracy. Instead, they consist of simple comparative statements and algebra for both the stick and slip states.

2. The Hayward, Karnopp, reset-integrator, and spring-based friction models will most likely not capture the Stribeck effect. However, the simulation will not involve a lubricated metal to metal contact, so the Stribeck effect will not be a factor that needs to be considered.

The three friction models that were chosen to code, test, and implement with the planar twin-pantograph haptic interface are:

1. Hayward, because it is one of the newest models in the literature, and is specifically formulated for haptics.

2. Karnopp, because Salcudean et al. [11] showed that it integrates well into haptic rendering and it is very efficient.

3. Hollerbach, because it has an efficient algorithm that should give a good stick-slip effect.
Chapter 4

Experimental Apparatus

This chapter will explain the details of the twin-pantograph haptic interface that was used for this project.

Figure 4.1: The 3 DOF, Twin-Pantograph Haptic Interface.

Figure 4.1 shows the 3 DOF twin-pantograph haptic interface that was used for the project. From the figure, it can be seen that there are two parallel linkages that are connected by an offset, passive wrist, giving the interface 3 independent degrees of freedom, x-translation, y-translation, and z-rotation.

The interface is driven by 4 identical 90W DC motors. Each motor is linked directly (ie. no gear reduction) to one of the limbs of the device. Each motor shaft is fitted with an optical encoder (resolution 0.09 degrees), for feedback of angular position. The limbs are constructed of carbon fiber tubes, and the joints and base are machined from aluminum stock. These materials, combined with low friction bearings at the joints make the interface very light and easy to maneuver, which is
ideal for haptics.

Control and simulation are run on a PC running Vx Works 1.2¹. This PC is responsible for the 'master' of the teleoperation system. The 'slave' is a separate PC that displays the virtual environment. The two PC’s communicate using udp packets over a serial connection.

Control and simulation code for the twin-pantograph are written with MATLAB² in a complex Simulink model. This model is compiled using the MATLAB 'Realtime Workshop' and then downloaded to the Vx Works target PC.

Figure 4.2: The Simulation Block Diagram

Figure 4.2 shows the simulation block diagram of the twin-pantograph setup. The friction S-Function was put in as a separate block right after the main simulation block. The main simulation block deals with inertia, and wall collision detection.

[6] [4] [12]

¹VxWorks is TM by Wind River Systems product, www.windriver.com
²MATLAB, Simulink, Realtime Workshop are TM by Mathworks, www.mathworks.com
Chapter 5

Implementation of Friction Models

5.1 Karnopp Model

The Karnopp Model of friction is simplest of the 3 models and was tested first.

Figure 5.1: Velocity vs. Friction Force in the Karnopp Model

Figure 5.1 shows the relationship between friction force and velocity. When the velocity is $-DV < v < DV$, the friction force is treated as linear with the velocity. This region is used to compensate for the difficulties that digital computers can have with near zero values. Once the velocity is safely away from zero, the coulomb friction is used.

In the first implementation of the Karnopp Model, there were substantial vibrations when the haptic interface was still. This was due to noise in the velocity measurements that the haptic interface calculating. It can be shown in Figure 5.1 that even the near zero velocities are used in a friction force calculation, resulting in these vibrations. To remedy these vibrations a 'dead band' filter approach was taken. Figure 5.2 shows the modified force vs velocity graph with a 'deadband' modification.

This modification prevented much of the noise in the velocity measurements from being used in a friction force calculation, and subsequently reduced the vibrations. This modification was
implemented with a simple conditional statement.

5.2 Hayward Model

The Hayward Model is a spring based stick-slip model. The basic equation that governs the friction force generated by the Hayward Model is

\[ F_f = -Kz \tag{5.1} \]

where \( z \) is the stretched length of the spring, defined as:

\[ z = x_k - w_k \tag{5.2} \]

and \( K \) is the stiffness coefficient of the virtual spring. Figure 5.3 shows a simplified diagram of the modelled physical geometry. The point \( x \) is measured from the haptic device, and \( w \) is the point where the spring meets touches the ground. When slipping, \( w \) stays a constant distance \( z_{max} \) away from \( x \). While when the mass is stuck, \( w \) is fixed.

The formulation of the Hayward model is as follows:
\[ W_k = \begin{cases} X_k - \frac{X_k - W_{k-1}}{|X_k - W_{k-1}|} z_{max}, & \text{if } \alpha(z)|X_k - W_{k-1}| > 1; \\ W_{k-1} + Y_k \alpha(z)(X_k - W_{k-1}), & \text{otherwise.} \end{cases} \] (5.3)

Where \( \alpha(z) = \alpha(X_k - W_{k-1}), Y_k = X_k - X_{k-1} \) and

\[ \alpha(X_k - W_{k-1}) = \begin{cases} 0, & \text{if } |z| \leq z_{\text{stick}}; \\ \frac{1}{z_{\text{max}}}, & \text{elsewhere.} \end{cases} \] (5.4)

According to Hayward [8], this formulation gives a very rich friction behaviour, as it has 3 regions including stick, slip, slide. The parameter \( z_{\text{stick}} \) allows the model to have relaxation oscillations when \( X_k - W_{k-1} \) is close to \( z_{\text{stick}} \). This relaxation oscillation behaviour describes the "bumpy" phenomenon that occurs when the transition from stick to slide happens.

### 5.3 Hollerbach Model

The Hollerbach Model [10] is also based on a virtual spring force. This has two distinct states, stuck and sliding.

In the sliding state, the friction force is defined as

\[ F_f = \mu_d |n| \frac{v}{|v|} \] (5.5)

Where \( n \) is the normal force of the mass on the sliding surface, \( \mu_d \) is the coefficient of dynamic friction, and \( v \) is the velocity of the mass.

The sliding state is retained until \( |v| \leq v_{\text{min}} \) at which point the stuck state is initialized. As soon as \( |v| \leq v_{\text{min}} \) is true, the present location, \( a \), and a center point, \( c \), defined as

\[ c = a - \frac{\mu_d}{k_f} \frac{v}{|v|} \] (5.6)

are recorded. The mass is now stuck, with the friction force calculated by

\[ F_f = k_f |n|(x - c) \] (5.7)

Where \( x \) is the location of the center of mass. While in the stuck state, the friction force is monitored. As soon as \( F_f \geq \mu_s |n| \), the state is changed from stuck to sliding, where \( \mu_s \) is the coefficient of static friction.

### 5.4 Code Details

The 3 friction models were coded into a single S-Function that was added to the Simulink model. This S-Function is found in REFER TO APPENDIX.
Chapter 6

Results

Data were collected for many different haptic experiments with each of the Friction models using different model parameters. For each run, 10 seconds of real time data was stored into memory, and dumped to a file once the run was completed. Each experiment consisted of 3 seconds of calibration, and approximately 7 seconds of realtime interaction with the haptic interface. To test the friction models, the haptic controller was moved back and forth along the x-axis (NOTE: this was done by hand, so the amplitudes and directions are not exactly the same for each run, this is characteristic of haptic interaction as the user is in control.)

6.1 Karnopp Model

The Karnopp Model has several parameters that influence it’s dynamic behaviour, the most important being the width of the $-DV < v < DV$ window. As previously mentioned in the implementation of friction models section, this model is also sensitive to noise in velocity measurements in this window.

For this set of experiments, the static friction limit was set for 1.2 newtons, while the dynamic friction limit was 1.0 newtons. The effect of this setting can be seen in all three figures as spikes that peak at 1.2 or -1.2 newtons just before a flatline of 1.0 newtons during non-zero velocity window.

Figure 6.1 shows response with no deadband, and $DV = 0.01 \text{ m/s}$. When the position is constant, the velocity noise can be seen as the friction force is non-zero, when in fact it should be. When the user takes his/her hand off of the device in this implementation, it vibrates slightly as a small force is being generated by the Karnopp Model, even though the controller is virtually still. The displacement of this vibration is not noticeable to the naked eye, but the output an obvious chatter that indicates that they are being driven.

Figure 6.2 is very similar to Figure 6.1 although this time, the $DV$ window is 0.001 m/s. This change caused even more vibration as the window is even smaller and more sensitive to velocity
Figure 6.1: Karnopp Results with No Deadband, DV = 0.01 m/s

Figure 6.2: Karnopp Results with No Deadband, DV = 0.001 m/s

Figure 6.3: Karnopp Results with Deadband, DV = 0.01 m/s
measurement noise. Figure 6.2 shows this response, as the force oscillations during constant position are similar but noticeably more violent than in Figure 6.1. Although the difference between Figures 6.2 and 6.1 is subtle, the vibrations definitely decrease when $DV$ is 0.01 m/s as opposed to 0.001.

Figure 6.3 shows the effect of the deadband modification. During the constant position periods, there are still small oscillations, but the friction force does, in fact, flatline for short periods. Again, the difference is subtle in the figures, but the audible noise and vibration are noticeably lower with the deadband modification.
6.2 Hollerbach Model

The Hollerbach Model has several model parameters, the most important of which is the velocity threshold. This is the parameter that determines whether the mass is in a stuck or sliding phase. In Figures 6.4 and 6.5 show the response when different velocity thresholds are used. It should be noted that Figure 6.4 uses a static friction level of 1.5 newtons, and a dynamic friction level of 1.0 newtons. This is the cause of the large spikes that occur during a stuck to sliding transition. Figure 6.5 uses a static and dynamic of 1.0 newton, which is why there is no spike during the stuck to sliding transitions in this experiment.

Figure 6.4: Hollerbach Results with Velocity Threshold 0.0001m/s

![Figure 6.4: Hollerbach Results with Velocity Threshold 0.0001m/s](image)

Figure 6.5: Hollerbach Results with Velocity Threshold = 0.0025m/s

Figure 6.4 shows the response when the velocity threshold is set to 0.0001m/s. It can be seen from this figure that 0.0001 m/s is too low a setting for the threshold, as the position oscillates rather than flatlines at the peaks. This occurs because transitions from stuck to sliding are induced by the noise in velocity measurements. The haptic interface actually oscillated when the user was
not touching it, indicating a 'marginally stable' or 'oscillatory' nature of the model with these parameters.

Figure 6.5 shows the response when the threshold is 0.0025. Even in this experiment it can be seen that the velocity threshold causes problems in the when the device is stationary. Large spikes of maximal friction force are present when the position is constant. To credit this model, however, the friction force does flatline during some regions of constant position. This spiking response due to velocity noise caused vibrations similar to that of the Karnopp Model when the $DV$ was set too low.
6.3 Hayward Model

The Hayward Model has one major parameter, \( z_{\text{max}} \). The tweaking of this parameter has a large effect on the response of the interface.

![Graph of Hayward Model results with \( z_{\text{max}} = 0.0001 \) m](image)

**Figure 6.6: Hayward Results with \( z_{\text{max}} = 0.0001 \) m**

![Graph of Hayward Model results with \( z_{\text{max}} = 0.001 \) m](image)

**Figure 6.7: Hayward Results with \( z_{\text{max}} = 0.001 \) m**

Figure 6.6 shows the response with \( z_{\text{max}} \) at 0.0001m while Figure 6.7 shows \( z_{\text{max}} \) at 0.001m. As we’ve seen before with other models, the Hayward Model is oscillatory when the position is supposed to be constant (ie. when the user is not touching the interface). This oscillatory behaviour seems proportional to the \( z_{\text{max}} \) setting, as with \( z_{\text{max}} \) of 0.0001, the oscillations are much smaller and faster than those at \( z_{\text{max}} \) of 0.001m. In fact, the oscillations when \( z_{\text{max}} \) is set to 0.0001m are so small that it is difficult to see the movement, and the user can hardly feel them. That being said, it is not ideal for the interface to be oscillating back and forth between maximum and minimum friction forces while it’s supposed to be stationary. That can’t be good for the motors.
The Hayward Model is a spring-mass based model, however, it does not have any damping in the formulation. This is why it is oscillatory. One of the major benefits of the Hayward Model is that it is ‘time free’. Because of this characteristic, velocity cannot be computed (as it is the time derivative of position), and damping (which is usually applied to a velocity term) is unfit for this model if it to keep its ‘time free’ nature.
Chapter 7

Conclusions

Each of the 3 friction models discussed in this report is capable of simulating friction for haptic interaction in realtime. However, none of these models is perfect. In fact, none of these models could simulate static friction adequately. The haptic interface was never held stationary until the static friction limit was met, it always moved slightly when touched by the user. This, unfortunately, is a limitation of haptic rendering. The interface is purely reactive, and so must wait for an action from the user before it can react with the appropriate force.

The Karnopp Model is the simplest of the 3 models, and is most likely the most computationally efficient (since it has no 'states' and very short, simple mathematical expressions. NOTE: no test was done to compare numerical speed between the models). It is very sensitive to the accuracy of velocity measurement. The addition of a deadband filter can aid in removing vibrations when the interface is near zero. For simplicity, assuming the availability of accurate velocity measurement, the Karnopp Model is the most likely candidate the three.

The Hollerbach model requires states, which serves as a relative disadvantage compared to the other two models because it requires information to be saved from previous timesteps. On top of this, this model can be oscillatory if the threshold is too high. These two hits make the Hollerbach model an unlikely candidate for implementation in a haptic interface.

The Hayward model is 'time free' which can be advantageous when the sampling rate of a haptic system fluctuates (for example, with a 'soft' real time operating system when events are allowed to be missed, the sampling rate may not be constant). However, the results of this paper show that it is by nature an oscillatory model. The only way to reduce or eliminate these oscillations is to use a damping term in the force formulation, which would in turn take away this model’s 'time free' feature. If $z_{max}$ is set very small, and there is some physical damping in the haptic interface (friction in bearings, joints, gears..etc), these oscillations may end up being small enough so that human touch could not detect them.
7.1 Future Work

This study is by no means conclusive. These friction models have only been tested on one haptic apparatus, and have only been tested with a small set of parameter settings. Additionally, the person who coded these models is by no stretch of imagination a good C programmer. To make more definitive conclusions on the overall strengths, weaknesses, and benefits of these models, each would have to their code optimized, responses tested for a wide range of parameter settings, and performances tested on different haptic devices.

7.2 Acknowledgements

I would like to acknowledge the following people for their support, and guidance.

- Daniela Constantinescu, Simon DiMaio, Purang Abolmaesumi, and Shahin Siropour for their help with the twin-pantograph interface in RCL.
- Dima Brodsky and Adrian Secord for many late night latec and C tutorials.
- Shahram Tafazoli for help with friction model research.
- Tim Salcudean for resources in RCL
Bibliography


Chapter 8

Appendix

8.1 S-Function for Friction Models

The following S-Function was written for the Simulink block diagram. It contains the 3 friction models

/
* sfuntmpl.c: C template for a level 2 S-function.
* 
* | See matlabroot/simulink/src/sfuntmpl.doc for a more detailed template |
* |------------------------------------------------------------------------|
* Copyright (c) 1990-1998 by The MathWorks, Inc. All Rights Reserved.
* $Revision: 1.19$
* /

/*
* You must specify the S_FUNCTION_NAME as the name of your S-function
* (i.e. replace sfuntmpl with the name of your S-function).
*/

/*
* Need to include simstruc.h for the definition of the SimStruct and
* its associated macro definitions.
*/
#ifndef MATLAB_MEX_FILE
#include <vxWorks.h>
#include <sysLib.h>
#include <math.h>
#endif
#define S_FUNCTION_NAME Body_Friction
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include "DynSim.h"

#define friction_model(S) (int)mxGetPr(ssGetSFcnParam(S,0))[0]
#define w_prev(i) RWork[i]

/* In alphabetical order 0=none, 1=hayward, 2=hollerbach, 3=karnopp */
Andrew's Friction Codes go here. 3 functions, hayward, hollerbach, karnopp

void hayward() {
    /* the function hayward will return a */
    float z_max = 0.0001;
    float z_stick = 0.00011;
    float w[2];
    float k = 10000;
    float alpha = 0;
    float decision[2];
    float y[2];
    float dectot;
    float ytot = 0;

    decision[0] = moving_body.pos[0] - w_prev[0];
    dectot = sqrt (decision[0]*decision[0] + decision[1]*decision[1]);
    y[0] = moving_body.pos[0] - moving_body.prev_pos[0];
    ytot = sqrt (y[0]*y[0] + y[1]*y[1]);

    if (dectot <= z_stick)
        alpha = 0;
    else
        alpha = 1/z_max;

    if (dectot > z_max) {
        w[0] = moving_body.pos[0] - (moving_body.pos[0]-w_prev[0])/dectot*z_max;
    }
    else {
        w[0] = w_prev[0] + ytot * alpha * (moving_body.pos[0] - w_prev[0]);
    }

    F_env[0] = F_env[0] - k * (moving_body.pos[0] - w[0]);

    w_prev[0] = w[0];
    w_prev[1] = w[1];
}

void karnopp() {
    float force;
    float slope = 120;
    float DV = 0.01;
    float coulomb = 1;
    float v;

    v = sqrt(moving_body.vel[0]*moving_body.vel[0] +
             moving_body.vel[1]*moving_body.vel[1]);

    if (v > DV)
        force = coulomb;
    else if (v < 0.001)
        force = 0;
    else
        force = slope * v;

    if (v > 0) {
        F_env[0] = F_env[0] - force*moving_body.vel[0]/v;
    }
}

26
```c
void hollerbach() {
    /*
    The Integer "state" is 0 for stuck, and 1 for sliding
    */

    float v_min = 0.0025;
    float coulomb = 1;
    float k = 1000;
    float disp[2];
    float v_mag=0;
    float threshold=0.0001;
    float disp_mag;

    v_mag = sqrt(moving_body.vel[1]*moving_body.vel[1] + moving_body.vel[0]*moving_body.vel[0]);
    disp[0] = moving_body.pos[0] - w_prev[0];
    disp_mag = sqrt(disp[0]*disp[0] + disp[1]*disp[1]);

    if ( andrew_state == 0 ) {
        if ( disp_mag > 0.0012 && v_mag > threshold ) {
            F_env[0] = F_env[0] - coulomb*moving_body.vel[0]/v_mag;
            andrew_state = 1;
        } else {
            F_env[0] = F_env[0] - k * (moving_body.pos[0] - w_prev[0]);
        }
    }

    if ( andrew_state == 1 && v_mag > threshold ) {
        F_env[0] = F_env[0] - coulomb*moving_body.vel[0]/v_mag;
        if ( v_mag <= v_min && v_mag > threshold ) {
            w_prev[0] = moving_body.pos[0] - coulomb*moving_body.vel[0]/v_mag/k;
            andrew_state = 0;
        }
    }

    /*-------------------*
    * S-function methods *
    *-------------------*/

    /* Function: mdlInitializeSizes ----------------------------------------
    * Abstract:
    *   The sizes information is used by Simulink to determine the S-function
    *   block’s characteristics (number of inputs, outputs, states, etc.).
    */
    static void mdlInitializeSizes(SimStruct *S) {
        /* See sfuntmpl.doc for more details on the macros below */
        ssSetNumSFcnParams(S, 1); /* Number of expected parameters */
        if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
            /* Return if number of expected != number of actual parameters */
            return;
        }
        ssSetNumContStates(S, 0);
        ssSetNumDiscStates(S, 0);
    }
```
if (!ssSetNumInputPorts(S, 1)) return;
ssSetInputPortWidth(S, 0, 1);
ssSetInputPortDirectFeedThrough(S, 0, 1);

if (!ssSetNumOutputPorts(S, 2)) return;
ssSetOutputPortWidth(S, 0, 1);
ssSetOutputPortWidth(S, 1, 2);

ssSetNumSampleTimes(S, 1);
ssSetNumRWork(S, 2);
ssSetNumIWork(S, 1);
ssSetNumPWork(S, 0);
ssSetNumModes(S, 0);
ssSetNumNonsampledZCs(S, 0);

ssSetOptions(S, 0);

/* Function: mdlInitializeSampleTimes ---------------------------------------
 * Abstract:
 * This function is used to specify the sample time(s) for your
 * S-function. You must register the same number of sample times as
 * specified in ssSetNumSampleTimes.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
  ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
  ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
/* Function: mdlInitializeConditions ---------------------------------------
 * Abstract:
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block. The initial states are placed
 * in the state vector, ssGetContStates(S) or ssGetRealDiscStates(S).
 * You can also perform any other initialization activities that your
 * S-function may require. Note, this routine will be called at the
 * start of simulation and if it is present in an enabled subsystem
 * configured to reset states, it will be call when the enabled subsystem
 * restarts execution to reset the states.
 */
static void mdlInitializeConditions(SimStruct *S)
{
}
#endif /* MDL_INITIALIZE_CONDITIONS */

#define MDL_START /* Change to #undef to remove function */
#if defined(MDL_START)
/* Function: mdlStart -----------------------------------------------
 * Abstract:
 * This function is called once at start of model execution. If you
 * have states that should be initialized once, this is the place
 * to do it.
 */
static void mdlStart(SimStruct *S)
{
  double    *RWork = ssGetRWork(S);
  printf("friction = %d
",friction_model(S));
}
#endif /* MDL_START */
w_prev[0] = moving_body.pos[0];
w_prev[1] = moving_body.pos[1];
andrew_state = 1;

andrewsindex = 0;
}
#endif /* MDL_START */

/* Function: mdlOutputs =======================================================
* Abstract:
* In this function, you compute the outputs of your S-function
* block. Generally outputs are placed in the output vector, ssGetY(S).
*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
    double *RWork = ssGetRWork(S);
    real_T *out_value = ssGetOutputPortRealSignal(S, 1);

    if ( friction_model(S) == 0 )
        friction_model(S) = friction_model(S);
    else if ( friction_model(S) == 1 )
        hayward(S);
    else if ( friction_model(S) == 2 )
        hollerbach();
    else if ( friction_model(S) == 3 )
        karnopp();

    if ( andrewsindex <= 5120 )
    {
        position[andrewsindex][0] = moving_body.pos[0];
        position[andrewsindex][1] = moving_body.pos[1];
        friction[andrewsindex][0] = F_env[0];
        friction[andrewsindex][1] = F_env[1];
        andrewsindex++;
    }
}

/* Function: mdlTerminate =====================================================
* Abstract:
* In this function, you should perform any actions that are necessary
* at the termination of a simulation. For example, if memory was
* allocated in mdlStart, this is the place to free it.
*/
static void mdlTerminate(SimStruct *S)
{
    FILE *fid;
    int i;

    fid = fopen("hollerbach_thresh0.0001.dat", "w");
    printf("Xposition Yposition Xforce Yforce\n");
    for (i=2000; i<andrewsindex-1; i++)
    {
        fprintf(fid, "test %d %9.6f %9.6f %9.6f\n",
            andrewsindex, position[2000][0], position[2000][1],
            friction[2000][0], friction[2000][1]);
    }
    fclose(fid);
/*====================================================================*
 * See sfuntmpl.doc for the optional S-function methods *
 *====================================================================*/

/*====================================*
 * Required S-function trailer *
 *====================================*/

#ifdef MATLAB_MEX_FILE
    /* Is this file being compiled as a MEX-file? */
#endif
#include "simulink.c" /* MEX-file interface mechanism */
#else
    /* Code generation registration function */
#endif
#include "cg_sfun.h" /* Code generation registration function */
#include "cg_sfun.h" /* Code generation registration function */
8.2 Apparatus - Full View