

# A Dynamics Workbench Model of a Pendulum

*This is an example of using the Dynamics Workbench to model a pendulum, perform a variety of analyses, and perform a simulation, all within the Mathematica environment*

## ■ Step 1: Building the Model

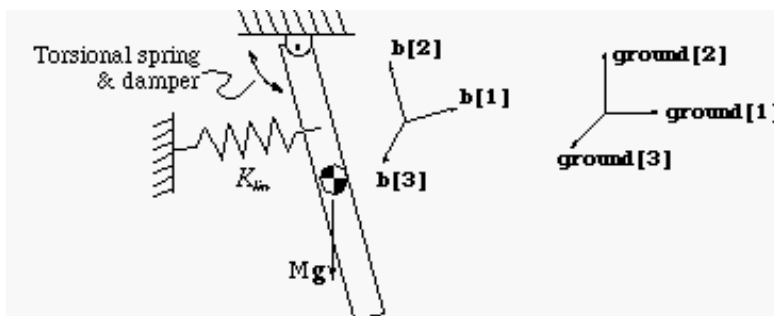
First, it is necessary to load the Dynamics Workbench package. Make sure it is in your path before you execute this command.

```
Needs["DynamicsWorkbench`"]
```

Now, initialize the modeling workspace:

```
NewModel[]
```

The system to be modeled is a pendulum, with a linear spring attached to the body, and a torsional spring and damper acting at the hinge joint. The inertial reference frame is called **ground**, and the pendulum, body **b**, has its own reference frame as well.



The **AddBody** command specifies that body **b** will be added, attached to ground via a **Hinge** joint. Specified are the mass, and the diagonal elements of the inertia matrix. The vector from the center of mass of the pendulum **b** to the **Hinge** joint is 1.5 m, in the direction of the **b[2]** axis. We also choose for the **Hinge** joint to be located at the origin of the **ground** reference frame. If it were located elsewhere, a vector **InbToJnt** would specify that location.

Moments of inertia are given relative to the three mutually orthogonal axes of a body's reference frame. If the inertia matrix has cross-terms, it may optionally be specified as a matrix or as a dyadic.

```
AddBody[ b, ground, Hinge, Mass->M, Inertia->{5,1,5},
  BodyToJnt->1.5 b[2] ]
{{1}, {1}}
```

**AddBody** returns two lists: the first is a list of the generalized coordinate indices, and the second is the list of the generalized speed indices. These coordinates and speeds, labeled  $\mathbf{q}[i]$  and  $\mathbf{u}[i]$ , describe the states of the body. Their definition depends on the type of joint used. In the case of the **Hinge** joint, which defaults to rotation about the **ground[3]** axis,  $\mathbf{q}[1]$  is the angle (in radians) of body **b**, measured counter-clockwise with respect to ground.  $\mathbf{u}[1]$  is the time derivative of  $\mathbf{q}[1]$ .

### ■ Gravitational Force

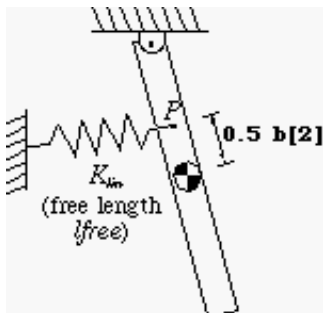
Next we apply a force to body **b** with the **AppFrc** command. This force is the weight vector applied at the center of mass. The third argument is the vector from the center of mass to the point of application of the force. In this case, the vector **0**. **AppFrc** returns the number of forces applied thus far.

```
AppFrc[ b, M grav, 0 ]
1
```

Gravity is defined as a vector in the negative **ground[2]** direction.

```
grav = -9.8 ground[2];
```

### ■ Linear Spring Force



We now define the point **P** where the linear spring applies its force. Point **P** is located 0.5 m from the pendulum's center of mass, along the **b[2]** axis. The **PosPnt** function gives this position relative to the origin of the inertial reference frame. Note that this position need not be specified using the **ground** frame's axes.

```
P = PosPnt[ 0.5 b[2], b ]
-1. b[2]
```

We now define a vector **R** between the point **P** and the fixed attachment for the spring. The fixed attachment is located at **-1 ground[1] - 1 ground[2]**, or 1 m to the left of and 1 m below the **Hinge** joint.

```
R = P - (-1 ground[1] -1 ground[2])
-1. b[2] + ground[1] + ground[2]
```

The **Dist** function can be used to find the distance between **P** and the attachment point. Equivalently, the **Norm** function could be used to return the magnitude of **R**.

```
length = Dist[ P, -1 ground[1] -1 ground[2] ]
Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]]
```

We next define the force produced by the spring, which is proportional to the stretched length (**length - lfree**) with spring constant **Klin**, along the unit vector given by **Normed[R]** (normalized vector).

```
F = -Klin (length - lfree) Normed[ R ]
-((Klin (-lfree + Sqrt[3. - 2. Cos[q[1]] +
2. Sin[q[1]]])) /
Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]]) ground[1] \
+ -(Klin (-lfree +
Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]])) /
Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]]) ground[2] \
+ (1. Klin (-lfree +
Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]])) /
Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]] b[2]
```

The **AppFrc** is now used to apply the spring force **F** to the pendulum, body **b**, at the specified location of **P** in **b**.

```
AppFrc[ b, F, 0.5 b[2] ]
2
```

## ■ Torsional Spring & Damper

The torsional spring has spring constant **Kr**, and a viscous friction constant **Ffric**. Similar to the free length of the linear spring, it has a resting angle **qbias**. The **AppTrq** command is used to apply this torque, a function of the generalized coordinate **q[1]**, which is the angle of the pendulum, to body **b**. The axis of this torque is coincident with the **ground[3]** axis.

```
AppTrq[ b,
(-Kr (q[1] - qbias) - Ffric u[1]) ground[3] ]
1
```

## ■ Form equations of motion

The **EOM** command forms the equations of motion for this system. In this case, there is a single equation in terms of  $\mathbf{u}[1]$ , the time derivative of  $\mathbf{u}[1]$ .

```
eom = EOM[
{(-1. Klin Cos[q[1]] (-lfree +
  Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]])) /
  Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]] +
  Sin[q[1]] ((0.5 Klin
    (-lfree + Sqrt[3. - 2. Cos[q[1]] +
      2. Sin[q[1]])) /
    Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]] +
    1.5 (-9.8 M - (Klin
      (-lfree +
        Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]]) \
      / Sqrt[3. - 2. Cos[q[1]] + 2. Sin[q[1]]])) -
    Kr (-qbias + q[1]) - Ffric u[1] +
    (-5 - 2.25 M) (u[1])' == 0}
```

One convenient way of assigning parameters is to put them in the form of rules, which makes it easy to perform analyses using different sets of parameters.

```
parms = {M->10, qbias->0.3, Klin->10, Ffric->10, Kr->100,
lfree -> 1 };
```

## ■ Static Equilibrium Analysis

Let's first perform a static analysis to see where the pendulum rests, given the default parameters shown above. The **FindEquil** command will find the equilibrium for any variable, given a solvable set of equations and sufficient fixed parameters. Here, the equation of motion is given, with its parameters set to default, and velocity and acceleration,  $\mathbf{u}[1]$  and  $\mathbf{u}[1]$ ', respectively, set to zero. An equilibrium is sought for  $\mathbf{q}[1]$ , with a provided initial guess of 0 as its value.

```
static = FindEquil[ eom /. parms /. {u[1]->0,u[1]'->0},
  {q[1],0}]
{q[1] -> 0.116921}
```

Note that the **FindEquil** command is not confined to static analysis. For example, it can also be used to perform dynamic analysis. Here, the acceleration is sought for the condition where the pendulum is momentarily at rest in its home position ( $\mathbf{q}[1]$ ,  $\mathbf{u}[1]$  equal to 0).

```
FindEquil[ eom /. parms /. {q[1]->0,u[1]->0},
  {u[1]',0} ]
{(u[1])' -> 1.09091}
```

## ■ Dynamic Analysis on Equilibrium Point

One way to determine whether the static equilibrium point found above is stable is to integrate the equations of motion, starting at a configuration slightly different from the equilibrium point. First, we define a perturbed value for  $q[1]$ , by adding 0.001 to the static equilibrium value:

```
perturb = q[1] + 0.0001 /. static
0.117021
```

The **AtT0** function is used to define a set of initial conditions. Here,  $q[1]$  is defined equal to the perturbed value, and  $u[1]$  equal to 0.

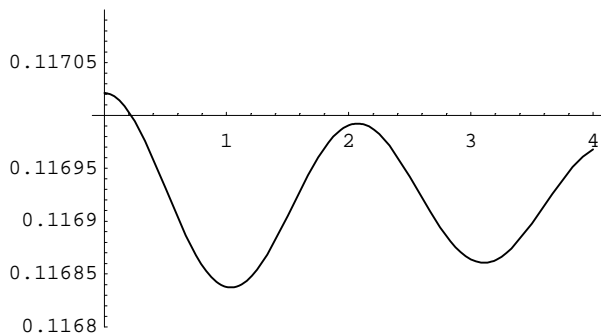
```
initcond = AtT0[ q[1] == perturb,
  u[1]==0]
{q[1][0] == 0.117021, u[1][0] == 0}
```

The **NDSolve** command then integrates the equations of motion. The full set of equations includes the equations of motion with parameters set, the kinematical differential equations provided by the Dynamics Workbench (and stored in **Kinematics**), and the initial conditions. **NDSolve** will solve for  $q[1]$  and  $u[1]$ , from time 0 to 0.5 sec.

```
trajectory = NDSolve[ Join[ eom/.parms, Kinematics, initcond],
  {q[1],u[1]}, {t,0.,4}];
```

We can then plot the trajectory vs. time, and verify that the pendulum oscillates about and gradually approaches the equilibrium point, verifying that it is stable.

```
Plot[ Evaluate[ q[1] /. trajectory ],
  {t, 0, 4}, PlotRange->{Automatic,{.1168,.1171}} ]
```



-Graphics-