Notes and Reflections

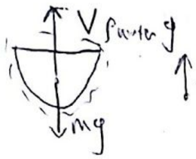
I was able to make very effective use of study groups this Int-set, and working through the problems with Siddhartan helped me understand them better. I now know the basic ideas behind this, although the nasty algebra can still get nasty at times, and I'm not sure I can reliably solve many-body versions of these problems.

Table of Contents

Solutions begin on next page

More Boats!

$M = A\,\rho_{H_2O} \cdot m^2$



$\uparrow + \qquad \sum F = \frac{dp}{dt} = ma$

mg

$-mg + V\rho g = ma$

$-\cancel{A}\cancel{\rho} g + \cancel{A}(2-d)^2\cancel{\rho} g = \cancel{A}\cancel{\rho} a$

$$\boxed{((2-d)^2 - 1)g = a}$$

Bobbing

$$\boxed{\ddot{d} = g(2-d)^2 - g}$$

Rocking

Assume d const.

$$\boxed{V = A \cdot m^2}$$

$\sum \tau = \frac{dL}{dt} = I\dot{\omega}$

$$\boxed{F_B = g\rho A(m^2)}$$

$F_B RA = I\dot{\omega}$

$$\boxed{I\dot{\omega} = g\rho A(m^2) \cdot C(m)\sin(2\theta) = I\ddot{\theta}}$$

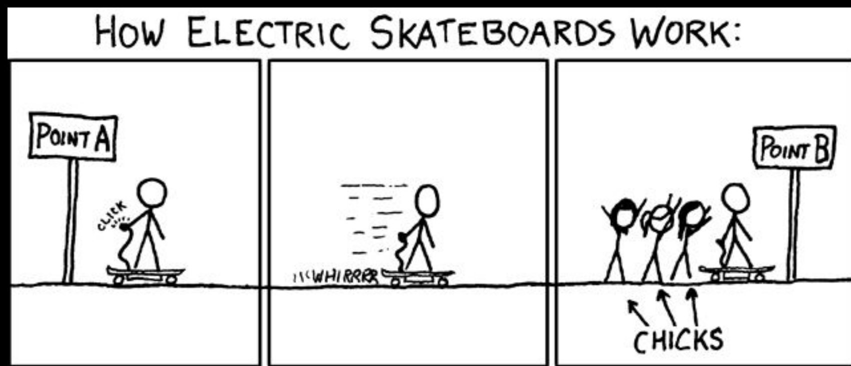
Both

$F_B RA = I\ddot{\theta}$

$$\boxed{\begin{gathered} g\rho A(2-d)^2 \cdot C(2-d)\sin(2\theta) = I\ddot{\theta} \\ \ddot{d} = g((2-d)^2 - 1) \end{gathered}}$$

Electric Skateboards

Obligatory xkcd



Step 1: FBD

Step 2: Parameters

```
In[125]:= params = <| m → 1, g → 9.8, motor[t] → -r[t], i → 3 |>
Out[125]= <| m → 1, g → 9.8, motor[t] → -r[t], i → 3 |>
```

Step 3: Define forces

```
In[126]:= fn = normal[t] θhat;
          fmotor = motor[t] rhat;
          fg = m g RotationMatrix[θ[t]].(-rhat);

In[129]:= (* Torques on ramp *)
          tn = -r[t] normal[t];
```

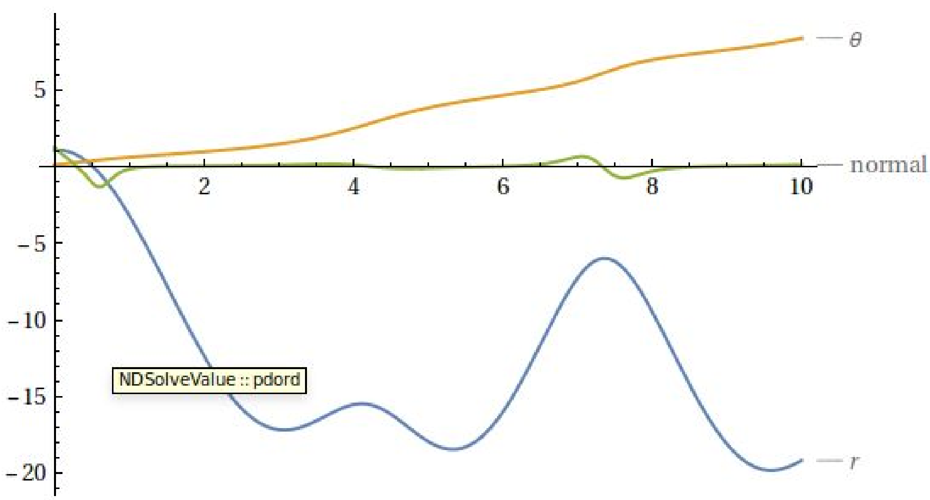
Step 4: Equations of motion

```
In[130]:= (* Mass motion *)
          eq1 = fn + fmotor + fg == m polaraccel[];
          (* Ramp rotation *)
          eq2 = tn == i θ''[t]

          eqns = Append[splitVectorEqn[eq1], eq2]
                  NDSolveValue :: pdord
Out[131]= -normal[t] r[t] == i θ''[t]
Out[132]= {-g m Cos[θ[t]] + motor[t] == m (-r[t] θ'[t]^2 + r''[t]),
           normal[t] - g m Sin[θ[t]] == m (2 r'[t] θ'[t] + r[t] θ''[t]), -normal[t] r[t] == i θ''[t]}
```

Step 5: Solve

```
In[137]:= sol = Quiet@NDSolveValue[eqns /. params, {r[t], θ[t], normal[t]}, {t, 0, 10}];

In[135]:= Plot[Evaluate@sol, {t, 0, 10}, PlotLabels → {r, θ, normal}]
```

Out[135]=

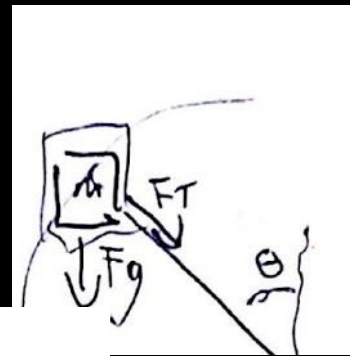


Note that a proper solution would define initial conditions. In this case, I just let *Mathematica* guess some.

Segways

Just a pendulum

Step 1: FBD



Step 2: Parameters

```
In[158]:= params = <| m → 1, g → 9.8 , l → 2|>
Out[158]= <|m → 1, g → 9.8, l → 2|>
```

Step 3: Define forces

```
In[182]:= fg = m g RotationMatrix[θ[t]].(-rhat);
          ftension = tension[t] rhat;
```

Step 4: Equations of motion

```
In[184]:= (* Mass motion *)
          eq1 = fg + ftension == m polaraccel[];
          eqns = splitVectorEqn[eq1]
```

Out[185]= $\{-g\, m \operatorname{Cos}[\theta[t]] + \operatorname{tension}[t] == m\,(-r[t]\,\theta'[t]^2 + r''[t]),$
$-g\, m \operatorname{Sin}[\theta[t]] == m\,(2\, r'[t]\,\theta'[t] + r[t]\,\theta''[t])\}$

```
          (* Constraints *)
In[194]:= constraint = r[t] == l;
```

Step 5: Solve

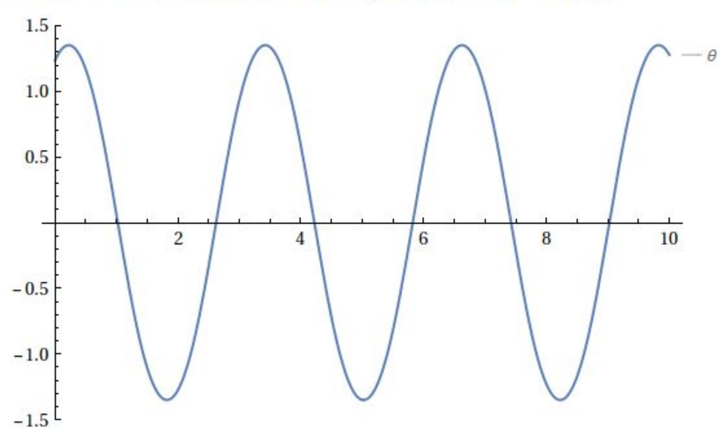
```
In[189]:= sol = NDSolveValue[{eqns, constraint} /. params, {θ[t]}, {t, 0, 10},
            Method →
             {"IndexReduction" → {"Pantelides", "ConstraintMethod" → "Projection"}}]
```

NDSolveValue: Structural analysis indicates that 2 initial conditions are needed to fix the state of the system. Currently only 0 initial conditions are specified. NDSolve may return one of a family of solutions.

```
Out[189]= {InterpolatingFunction[ Domain: {{0., 10.}} Output: scalar ][t]}
```

```
In[192]:= Plot[Evaluate@sol, {t, 0, 10}, PlotLabels → {θ}]
```

Out[192]=



A perfect segway

All that needs to change is replacing $\frac{d^2}{dt^2}\vec{r}$ with $\frac{d^2}{dt^2}(\vec{r}+\vec{x})$ on the rhs of Newton's equation.

Step 2: Parameters

```
In[215]:= params = <| m → 1, g → 9.8 , l → 2 |>
Out[215]= <| m → 1, g → 9.8, l → 2 |>
```

Step 3: Define forces

```
In[216]:= xhat = RotationMatrix[-θ[t]].(-θhat);

In[217]:= fg = m g RotationMatrix[θ[t]].(-rhat);
          ftension = tension[t] rhat;
```

Step 4: Equations of motion

```
In[219]:= (* Mass motion *)
          eq1 = fg + ftension == m polaraccel[] + x''[t] xhat;
          eqns = splitVectorEqn[eq1]
Out[220]= {-g m Cos[θ[t]] + tension[t] == m (-r[t] θ'[t]^2 + r''[t]) - Sin[θ[t]] x''[t],
           -g m Sin[θ[t]] == -Cos[θ[t]] x''[t] + m (2 r'[t] θ'[t] + r[t] θ''[t])}

In[221]:= (* Constraints *)

In[222]:= constraint = r[t] == l;

In[223]:= (* Driving functions *)

In[243]:= driver = x[t] == Sin[2 t];
```

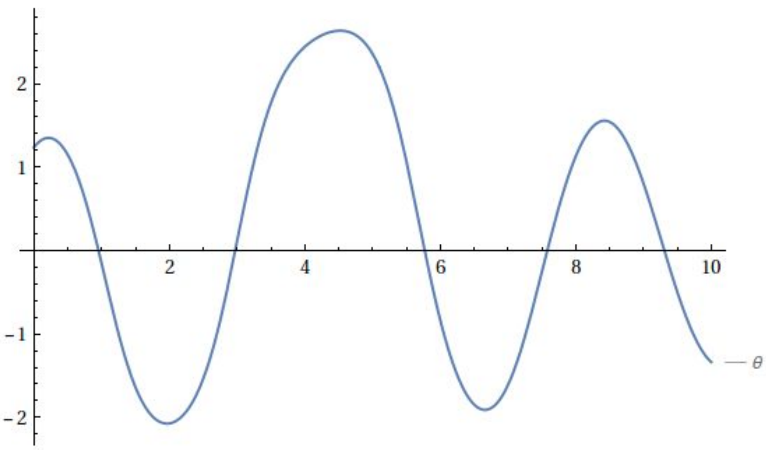
Step 5: Solve

```
In[244]:= sol = NDSolveValue[{eqns, constraint, driver} /. params, {θ[t]}, {t, 0, 10},
            Method →
             {"IndexReduction" → {"Pantelides", "ConstraintMethod" → "Projection"}}]
```

NDSolveValue: Structural analysis indicates that 2 initial conditions are needed to fix the state of the system. Currently only 0 initial conditions are specified. NDSolve may return one of a family of solutions.

```
Out[244]= {InterpolatingFunction[ Domain: {{0., 10.}} Output: scalar ][t]}

In[245]:= Plot[Evaluate@sol, {t, 0, 10}, PlotLabels → {θ}]
```

Out[245]=



Massy segway

Step 2: Parameters

```
In[40]:= params = <| m1 → 1, m2 → 2, g → 9.8 , l → 2|>
Out[40]= <|m1 → 1, m2 → 2, g → 9.8, l → 2|>
```

Step 3: Define forces

```
In[41]:= xhat = RotationMatrix[-θ[t]].(-θhat);

In[42]:= fg = m1 g RotationMatrix[θ[t]].(-rhat);
         ftension = tension[t] rhat;
         fdrive = drive[t] xhat;
```

Step 4: Equations of motion

```
In[74]:= (* Mass motion *)
         eq1 = fg + ftension == m1 (polaraccel[] + x''[t] xhat);
         (* Base motion *)
         eq2 = -ftension + fdrive == m2 x''[t] xhat;
         eqns = Join[splitVectorEqn[eq1], splitVectorEqn[eq2]]
Out[76]= {-g m1 Cos[θ[t]] + tension[t] == m1 (-r[t] θ'[t]^2 + r''[t] - Sin[θ[t]] x''[t]),
          -g m1 Sin[θ[t]] == m1 (2 r'[t] θ'[t] - Cos[θ[t]] x''[t] + r[t] θ''[t]),
          -drive[t] Sin[θ[t]] - tension[t] == -m2 Sin[θ[t]] x''[t],
          -Cos[θ[t]] drive[t] == -m2 Cos[θ[t]] x''[t]}

In[28]:= (* Constraints *)

In[48]:= constraint = r[t] == l;
```

Unfortunately, I wasn't able to get far enough to fully solve this one. I am confident I can, but I just haven't yet.