Hwk #7 Solution Notebook

Morin 5.9

If a particle moves in one dimension under the influence of a force F(x) = -V'(x), then at any place where the force drops to zero (i.e. any x0 such that F(x0)=0) it is a possible "motion" for the particle to stay put (i.e x(t) = x0). If the particle is displaced a small amount from x0, though, what happens?
Expanding F(x) in a Taylor series near x0: F(x) = F(x0) + F'(x0)(x-x0) + ... = 0 + F'(x0)(x-x0) + ...
We obtain the force law of a spring with equilibrium point x0 and spring constant -F'(x0).
So a very common question to be asked is: given V(x) where are its minima (i.e. find the x0 such that V'(x0)=0), and for each minimum what is the frequency of small oscillations about that minimum
(i.e. what is k_eff = V"(x0), and then ω=Sqrt[k_eff/m]).

Here we are asked these questions for

```
V[x_] = - C x^n Exp[-α x]
```

First let's plot this thing:

```
p = {C → 1, n → 3, α → 1}
```

```
plotV = Plot[V[x] /. p, {x, 0, 10}]
```

Next locate the minimum:

```
eqn0 = V'[x] == 0
```

```
soln0 = Solve[eqn0, x]
```

And compute the 2nd derivative there:

```
V''[x] /. soln0[[1]]
```

```
keff = Simplify[%]
```

So to answer the question posed, the (angular) frequency of small oscillations is

```
ω0 = Sqrt[keff / m]
```

And the period would then be

```
T0 = 2 π / ω0
```

So let's choose parameters and actually find the motion. First atttempt:

```
p = {C → 1, n → 3, α → 1, m → 1}
```

```
eqnFma = m x''[t] == -V'[x[t]]
```

```
eqnFma /. p
```

Wait as long as you like for this one to come back, then hit Alt-plus to abort the calculation:

```
DSolve[{eqnFma /. p}, x[t], t]
```

Even when the differential equation is too hard for *Mathematica*, we can get a numerical solution.
Let's launch the motion from the equilibrium point with some choice of initial velocity:

```
x0 = x /. soln0[[1]] /. p
```

```
sN = NDSolve[{eqnFma /. p, x[0] == x0, x'[0] == .1}, x[t], {t, 0, 2 T0 /. p}]
```

```
x1[t_] = x[t] /. sN[[1]]
```

```
Plot[x1[t], {t, 0, 2 T0 /. p}]
```

So for a "small" velocity (.1) this looks pretty much like a sine.
Let's find the exact period and compare

```
FindRoot[x1[t] == x0, {t, 9}]
```

```
T0 /. p // N
```

For larger energies, the motion will not look so much like a sine:

```
sN2 = NDSolve[{eqnFma /. p, x[0] == x0, x'[0] == 1}, x[t], {t, 0, 2 T0 /. p}]
```

```
x2[t_] = x[t] /. sN2[[1]]
```

```
Plot[x2[t], {t, 0, 2 T0 /. p}]
```

Let's animate the 1D motion on top of the graph of the potential.
For maximum artistry, let's choose to draw this motion at the appropriate energy level:

```
E0 = (V[x0] + (1 / 2) m x2'[0]) /. p
```

```
plotE0 = Plot[E0, {x, 1, 9}]
```

```
Animate[Show[plotV, plotE0, ListPlot[{{x2[t], E0}}, PlotStyle → {Red, PointSize[.02]}]],
 {t, 0, 2 T0 /. p}]
```

Morin 5.15 and 5.16

Between time t and time t+Δt a snowball of mass Δm==σ*Δt moving at velocity u collides with a car of mass M moving at v(t).
In the car frame the speed of the incoming snowball is u-v. Assuming the collision is elastic (not so realistic for a snowball),
the outgoing speed is (v-u) in the car frame, and therefore 2v - u in the ground frame. Conservation of momentum then reads:

```
Δp = (M (v + Δv) + Δm (2 v - u)) - (M v + Δm u)
```

```
s = Solve[Δp == 0, Δv][[1]]
```

Next divide by Δt to get the equation of motion:

```
Δv / Δt /. s
```

```
eqn = v'[t] == (Δv / Δt /. s /. {Δm → (σ (u - v[t]) / u) Δt, v → v[t]} // Simplify)
```

And now we can ask for the general solution:

```
DSolve[eqn, v[t], t]
```

And specifically if we start from rest at time t=0 here is the velocity:

```
sv1 = DSolve[{eqn, v[0] == 0}, v[t], t][[1]]
```

And position:

```
x1[tf_] = Integrate[v[t] /. sv1, {t, 0, tf}, Assumptions → {M > 0, tf > 0, σ > 0}]
```

Choosing numerical parameters we can plot:

```
p = {u → 1, M → 1, σ → 1}
```

```
Plot[v[t] /. sv1 /. p, {t, 0, 3}]
```

```
Plot[x1[t] /. p, {t, 0, 3}]
```

In problem 5.16 we open the window and let the snowballs accumulate.

```
Δp = (m (v + Δv) + Δm (v)) - (m v + Δm u)
```

```
s = Solve[Δp == 0, Δv][[1]]
```

```
eqnv = v'[t] == (Δv / Δt /. s /. {Δm → (σ (u - v[t]) / u) Δt, v → v[t], m → m[t]} // Simplify)
```

Now the mass is a function of time, which obeys the differential equation:

```
eqnm = m'[t] == σ (u - v[t]) / u
```

Solving for M[t] and v[t] together:

```
soln = Simplify[DSolve[{eqnv, eqnm, m[0] == M, v[0] == 0}, {v[t], m[t]}, t], Assumptions → u > 0]
```

Plainly we are interested in the solution with positive mass. To get the position,

```
sv2 = soln[[2]]
```

```
pos = Integrate[v[t] /. sv2, t]
```

Starting from x=0 at t=0:

```
x2[t_] = pos - (pos /. t → 0)
```

And finally plotting both 5.15 and 5.16 together, we see that for the same parameters, the superballs make a better propellant:

```
Plot[Evaluate[{v[t] /. sv1, v[t] /. sv2} /. p],
 {t, 0, 3}, PlotStyle → {Red, Blue}, PlotRange → {0, 1}]
```

Let's take a look at the velocity *difference*, and note it has a max a some finite time:

```
Plot[Evaluate[((v[t] /. sv1) - (v[t] /. sv2)) /. p], {t, 0, 10}]
```

Falling Chain (Morin 5.28)

A chain of length L and mass density σ is held vertically with the bottom end attached to a support. The upper end is then released and the chain falls. First let's animate the motion. Each link is assumed just to fall at g until it reaches its resting place, so at time t a length (1/4) g t^2 is at rest:

Here R is a parameter to control the size of the semicircular loop. A fancy version would stop drawing this bit at the end of the motion, but we aren't fancy.

```
p = {L → 1, g → 1, σ → 1, R → .02}
```

```
tf = t /. Solve[(1 / 4) g t^2 == L, t][[2]]
```

Here y(t) is the length which is currently dangling:

```
y[t_] = (1 / 4) g t^2
```

```
draw[t_] := Show[ParametricPlot[(R {Cos[α], Sin[α]} + {0, -y[t]}) /. p,
   {α, 0, -π}, Axes → False, PlotRange → Evaluate[{-L, L} /. p]], Graphics[
   {Line[{{R, 0}, {R, -y[t]}} /. p], Line[Evaluate[{{-R, L - 2 y[t]}, {-R, -y[t]}} /. p]]}]]
```

```
Animate[draw[t], {t, 0, tf /. p}]
```

The homework question is to find the force on the chain applied at the fixed support.
Obviously part of this is the force needed to hold up a chain of length y(t), namely σ y g.
The other part is the impulsive force needed to stop the next little bit of chain.
Between time t and t+Δt a length

```
Δy = y'[t] Δt
```

comes to rest. Since it was moving at speed g*t, the momentum change is:

```
Δp = σ Δy g t
```

And the impusive force is

```
Fimp[t_] = Δp / Δt
```

which makes the total force:

```
F[t_] = Piecewise[{{Fimp[t] + σ y[t] g, t <= tf}, {σ L g, t > tf}}]
```

```
Plot[F[t] /. p, {t, 0, 1 + tf /. p}]
```

Speedy Travel (Morin 5.63)

Take an idealized spherical planet of uniform mass density ρ. Drill a straight tube through and make it frictionless.
Drop a mass down the tube. Describe the motion, and find out how long you have to wait for the mass to come back.

Soln: Let's use {x,y} coordinate system and orient our tube at constant y(t) = y0. Our goal is then to find x(t), starting from rest at x0=sqrt[R^2-y0^2]. Note that the problem statement didn't specify y0 or R---they had better not matter in the end, or the problem was not well-posed.
We'll find it useful along the way to introduce these two functions of x: r = Sqrt[x^2+y0^2] and θ = ArcSin[x/r]
There are two forces acting on our mass: gravity and a normal force which enforces the constraint y=y0.
The y-component of F=ma tells us what N needs to be, while the x-component reads:

```
m x''[t] == - Fg Sinθ
```

Here Fg is the magnitude of the gravity force when the mass is at location x, i.e. at radius r. To nail that down, recall Gauss's law for gravity, in particular that for a hollow spherical shell of radius R the gravitional field on the exterior is the same as that from a point mass, while the field inside vanishes. Building our uniform sphere out of a collection of hollow spheres, we see that the gravity force at radius r is that due to a point mass of size

```
M[r_] = ρ (4 / 3) π r^3
```

```
Fg[r_] = G M[r] m / r^2
```

Since sin(θ) = x/r, our equation of motion is:

```
eqn = m x''[t] == - Fg[r] x[t] / r
```

which we regcognize as good old simple harmonic motion:

```
DSolve[eqn, x[t], t]
```

with frequency

```
ω0 = Sqrt[(4 / 3) π G ρ]
```

and thefore period

```
T = 2 π / ω0
```

Note that indeed the period doesn't depend on the details such as the planet's radius, or even where we drill the tube. Numerically,

```
Needs["PhysicalConstants`"]
```

```
ρ0 = EarthMass / ((4 / 3) π EarthRadius^3)
```

```
T /. {G → GravitationalConstant, ρ → ρ0} /. {Newton → Kilogram Meter / Second^2}
```

```
Convert[5068.49 Second, Hour]
```

which is indeed a speedy trip to the other side of the world and back.
(We note that the book problem asks for the one-way time, i.e. half of this).
Amusinly, this is the same period one find for a satellite orbiting just above the atmosphere:

```
eqnFma = - m ω^2 R == (- G M m / R^2 /. M → (4 / 3) π ρ R^3)
```

```
Solve[eqnFma, ω]
```

Displaying the motions together. Choose a planet of radius 1 and y0=1/2

```
circle = ParametricPlot[{Cos[α], Sin[α]}, {α, 0, 2 π}];
```

```
y0 = 1 / 2
```

```
x0 = Sqrt[1 - y0^2]
```

```
tube1 = Graphics[Line[{{x0, y0}, {-x0, y0}}]];
```

```
tube2 = Graphics[Line[{{x0, y0}, {-x0, -y0}}]];
```

```
Show[circle, tube1, tube2]
```

```
pt1[t_] = {x0 Cos[t], y0};
pt2[t_] = {x0 Cos[t], y0 Cos[t]};
ϕ = ArcTan[y0 / x0];
pt3[t_] = {Cos[t + ϕ], Sin[t + ϕ]};
```

```
Animate[Show[circle, tube1, tube2,
  ListPlot[{pt1[t], pt2[t], pt3[t]}, PlotStyle → {Red, PointSize[.02]}]], {t, 0, 2 Pi}]
```

Morin 5.73

A one dimensional collision: a mass 2m moves to the right at v and a mass m move to the left at v. They collide elastically. Find the final velocities (a) working in the lab frame and (b) in the CM frame.

Soln: in the lab frame we write the two relations coming from momentum and energy conservation:

```
eqnP = 2 m v + m (-v) == m v1 + 2 m v2
```

```
eqnK = (1 / 2) 2 m (v) ^2 + (1 / 2) m (-v) ^2 == (1 / 2) m v1^2 + (1 / 2) 2 m v2^2
```

```
Solve[{eqnP, eqnK}, {v1, v2}]
```

As always we get two solutions to our quadratic, where here the first solution corresponds to nothing happening,
i.e. the particles sliding right through each other with unchanged velocities. We are interested in the nontrivial 2nd solution.

To work in the center of mass frame we first need to find the velocity of the CM.
Since Ptot = Mtot vcm,

```
vcm = (2 m v - m v) / (2 m + m)
```

The initial state CM velocities are then obtained by subtracting off vcm

```
v2cmInit = v - vcm
```

```
v1cmInit = -v - vcm
```

As a cross check, note that in the CM frame the total momentum is zero:

```
m v1cmInit + 2 m v2cmInit
```

If the collision is elastic, then the final state consists of both masses just reversing their directions.
To show this the long way, though:

```
eqnPcm = m v1cmInit + 2 m v2cmInit == m v1cmFinal + 2 m v2cmFinal
```

```
eqnKcm = (1 / 2) m (v1cmInit) ^2 + (1 / 2) 2 m ( v2cmInit) ^2 ==
  (1 / 2) m (v1cmFinal) ^2 + (1 / 2) 2 m (v2cmFinal) ^2
```

```
s = Solve[{eqnPcm, eqnKcm}, {v1cmFinal, v2cmFinal}]
```

Which again has the two solutions: pass on through w/o interaction, or bounce off backwards.
Back in the lab frame we then find final velocities:

```
v1labFinal = (v1cmFinal /. s[[2]]) + vcm
```

```
v2labFinal = (v2cmFinal /. s[[2]]) + vcm
```

which happily enough agrees with the result above.

■ Morin 5.80

A 2D elastic collision: mass m2=2m collides with m1=m, and after the collision the two head off at angles +*θ* and -*θ*. Find *θ*..

Soln: write the three conservation law (for px, py and kinetic energy), then solve for the three unkowns (v1, v2 and *θ*)

```
eqnpx = 2 m v0 == 2 m v2 Cos[θ] + m v1 Cos[θ]
```

```
eqnpy = 0 == 2 m v2 Sin[θ] + m v1 Sin[-θ]
```

```
eqnK = (1 / 2) 2 m v0^2 == (1 / 2) 2 m v2^2 + (1 / 2) m v1^2
```

```
Solve[{eqnpx, eqnpy, eqnK}, {v1, v2, θ}]
```

Note how *Mathematica* gives us the same physical solution multiple times with differing values of the angles.
But there really are only four physically distinct solutions here: (1) nothing happens (stated 3 times),
(2) particles continue along the x-axis at appropriate speeds (stated 3 times as well),
(3) mass 2 scatters at $\pi/6$ (stated twice) and (4) mass scatters at $-\pi/6$. These last two are the ones we want.

Now repeat, allowing the m2 to be more general:

```
eqnpx = α m v0 == α m v2 Cos[θ] + m v1 Cos[θ]
```

```
eqnpy = 0 == α m v2 Sin[θ] + m v1 Sin[-θ]
```

```
eqnK = (1 / 2) α m v0^2 == (1 / 2) α m v2^2 + (1 / 2) m v1^2
```

```
s = Solve[{eqnpx, eqnpy, eqnK}, {v1, v2, θ}]
```

And we see that the when α exceeds 3 the argument of the ArcCos gets larger than 1 and we have no real solution.

Let's animate!

```
x1[t_] = Piecewise[{{0, t < 0}, {v1 t Cos[-θ], t > 0}}]
```

```
y1[t_] = Piecewise[{{0, t < 0}, {v1 t Sin[-θ], t > 0}}]
```

```
x2[t_] = Piecewise[{{v0 t, t < 0}, {v2 t Cos[θ], t > 0}}]
```

```
y2[t_] = Piecewise[{{0, t < 0}, {v2 t Sin[θ], t > 0}}]
```

```
p = {v0 → 1, α → 1}
```

```
Animate[ListPlot[{{x1[t], y1[t]}, {x2[t], y2[t]}} /. s[[10]] /. p,
  PlotStyle → {Red, PointSize[.03]}, PlotRange → {{-2, 2}, {-2, 2}}], {t, -2, 2}]
```

And finally give the parameter α a knob:

```
Manipulate[Animate[Show[ListPlot[{{x1[t], y1[t]}} /. s[[10]] /. {v0 → 1, α → q},
    PlotStyle → {Red, PointSize[.03]}, PlotRange → {{-2, 2}, {-2, 2}}],
   ListPlot[{{x2[t], y2[t]}} /. s[[10]] /. {v0 → 1, α → q}, PlotStyle → {Blue, PointSize[.04]},
    PlotRange → {{-2, 2}, {-2, 2}}]], {t, -2, 2}], {q, 0, 3}]
```

Whirling mass

A couple of weeks ago we found the motion of a mass at the end of an ever-shortening rope:

```
r[t_] = r0 - v0 t
```

Since the force was purely in the radial direction the θ component of F=ma reads:

```
eqnθ = m (r[t] ω'[t] + 2 r'[t] ω[t]) == 0
```

which we DSolve, starting with angular speed ω0:

```
s = DSolve[{eqnθ, ω[0] == ω0}, ω[t], t][[1]]
```

The radial equation tells us what the the tension has to be:

```
T = -m (r''[t] - r[t] ω[t]^2)
```

```
T /. s
```

Notice that these expressions are all written as functions of time, but we can also think of them as functions of the radius:

```
ω[r_] = ω0 r0^2 / r^2
```

```
T =.
```

```
T[r_] = m r0^4 ω0^2 / r^3
```

Now suppose we reel in the mass from r0 to a smaller r1. The kinetic energy changes from

```
K0 = (1 / 2) m ((r'[0])^2 + (r[0] ω[r0])^2)
```

to

```
K1 = (1 / 2) m ((v0)^2 + (r1 ω[r1])^2)
```

Notice that the radial velocity is always v0 in this problem, so that piece of the kinetic energy is unchanging.
Our job is to compare with the work done by the winch reeling in the rope:

```
work = Integrate[-T[r], {r, r0, r1}, Assumptions → {r0 > 0, r1 > 0}]
```

Note that relative to the normal positive r directoin the force is -T, and the work done is *positive*.
Is it the same as the change in kinetic energy?

```
assertion = K1 - K0 == work
```

```
Simplify[assertion]
```

The Space Elevator (Morin 5.65)

(a) Describing the earth by its radius (R), density (ρ) and angular frequency (ω=2π/day), find the radius of *geosynchronous* orbit, i.e. that magical *r* at which we "park" satellites.
More precisely they orbit with with angular frequency precisely ω, so they are at rest relative to the spinning Earth.
We are asked to express the answer in terms of a dimensionless number η=r/R.

Well, first let's define the Earth's mass:

```
M = (4 / 3) π R^3 ρ
```

in terms of which the graviational force is the familiar G M m / r^2. Equating that with m a, where the acceleration is of course a-vec = (-r ω^2) r-hat,
Using the relation to express r:

```
eqn = - G M m / r^2 == - m r ω^2
```

```
s = Solve[eqn, r]
```

Obviously we want the 3rd of these cube roots (the others are complex).
And then we get our η cubed:

```
η3 = (r / R )^3 /. s[[3]]
```

To convert to a number, let's load up all the physical constants:

```
Needs["PhysicalConstants`"]
```

This defines things like

```
GravitationalConstant
```

and

```
EarthRadius
```

It doesn't define the density, so let's create our own:

```
ρ0 = EarthMass / ((4 / 3) π EarthRadius^3)
```

and then plug them in:

```
params = {G → GravitationalConstant, ω → (2 π / (3600 * 24 Second)), ρ → ρ0}
```

```
η3 /. params
```

Sometimes you have to help *Mathematica* along:

```
NewtonHelp = {Newton → Kilogram Meter / Second^2}
```

```
η3 /. params /. NewtonHelp
```

And our numerical expression for η is:

```
η1 = (%)^(1 / 3)
```

Next envision a very long rope of uniform mass density (call it λ, though they didn't specify it---hopefully we won't need it to respond to the questions asked).
It extends from radius R to η' R, where our job is to find η'. But we'll call our variable, say, α, since *Mathematica* is fussy about primes.
First let's get the tension on the structure applied at the contact with Earth.
Viewed as a system, the rope has mass m = λ (α-1) R, center of mass located at 1+(α-1)/2 in units of R:

```
m = λ (α - 1) R
```

```
rcm = R + R (α - 1) / 2
```

And the total gravitational force on the sytem is the sum of of the bits. This requires a quick little integral,
which provides as good a time as any to introduce the Assumptions option to Integrate---this is where you
can inform *Mathematica* of extra information it can assume is true.
Without it:

```
Integrate[ G M λ / r^2, {r, R, α R}]
```

With it:

```
Fg = Integrate[ G M λ / r^2, {r, R, α R}, Assumptions → {α > 0}]
```

Together, gravity and tension provide the force necessary for circular motion:

```
T0 =.
```

```
eqn = - m ω^2 rcm == -Fg - T0
```

So let's solve for the tension:

```
solnT0 = (Solve[eqn, T0][[1]] // Simplify)
```

If our elevator rope is to stand up, the tension can never drop below 0 anywhere, including at r=R.
We'll have to check this with the explict T(r) later on, but it seems plausible that if T0>0 the rest will be OK.
So for what value of α does T0 vanish?
Dividing by the various constants, the polynomial whose roots we want is:

```
q = -8 G π (-1 + α) ρ + 3 α (-1 + α^2) ω^2
```

Notice this cubic factorizes---the α=1 root is the trivial elevator of zero length:

```
Factor[q]
```

Adding in the roots of the quadratic, the list of all solutions is:

```
solnα = Solve[q == 0, α]
```

Plainly we are interesting in the positive solution, which numerically is:

```
numα = α /. solnα[[3]] /. params // N
```

And again we need to give a push:

```
α0 = Simplify[numα /. {Newton → Kilogram Meter / Second^2}, Assumptions → {Second > 0}]
```

(c) We are asked at what radius the tension is maximized. Well at such a radius, it is clear that T'(r) = 0, which means that the force on a tiny little piece
of string at that location is purely gravitational. If that bit is supposed to keep up with the rest of the elevator, it had better be located at the geosynchronous radius.

But to fully answer this question, we might as well go ahead and get the full T(r).
To get it, we derive and solve a simple differential equation:
Consider the small bit of rope between r and r+Δr. Its mass is

```
Δm = λ Δr
```

F = ma for that bit reads:

```
eqn = T[r + Δr] - T[r] - G M Δm / r^2 == - Δm ω^2 r
```

Taking Δr to 0, we get the differential equation

```
eqnT = T'[r] == -r λ ω^2 + G M λ / r^2
```

Solving with the boundary condition that the tension at the Earth end is T0:

```
solnT = (DSolve[{eqnT, T[R] == (T0 /. solnT0)}, T[r], r][[1]]) // Simplify
```

As a check that we are on the right track, let's get the tension at the far end:

```
T[r] /. solnT /. r → α R
```

In fact we could have used this condition to find T(r):

```
solnT = (DSolve[{eqnT, T[α R] == 0}, T[r], r][[1]]) // Simplify
```

Now let's put in numbers and plot. For plotting define the dimensionless parameter x = r/R:

```
q = (T[r] /. solnT /. α → α0 /. params /. NewtonHelp // Simplify)
```

```
F[x_] = q /. {r → x R, λ → 1 Kilogram / Meter} /. R → EarthRadius // Simplify
```

This is the force in Newtons. Let's divide out the units:

```
f[x_] = F[x] / Newton  /. NewtonHelp
```

```
Plot[f[x], {x, 1, α0}]
```

Where is that maximum? Is it in fact at ηR?

```
FindRoot[f'[r] == 0, {r, 7}]
```

Yup.