Mathematica Homework #3

Email notebook to corbin@physics.ucla.edu
with a subject line: [Physics 105A]
by on or about Friday, 8 Nov

In a general sense, there are three steps one engages in to solve a problem using Lagrangian dynamics:

- Pick an inertial coordinate system in which to evaluate the kinetic and potential energies of the system and write the Lagrangian in this inertial coordinate system.
- Recognizing that the coordinates you used may not be the most convenient way to describe the state of the system, choose a more convenient set. The beauty part is these new *'generalized coordinates'*) do not need to be tied to an inertial frame! Re-write each of the initial coordinates as functions of the new generalized coordinates and use those equations to re-write the Lagrangian (in terms of the generalized coordinates).
- Apply the Euler-Lagrange equation to the Lagrangian (recast in generalized coordinates). The resulting differential equations will be the equations of motion in the generalized coordinates! Easy-peasy.

Now on to our regularly scheduled program...

- In the first cell, enter your **name**, **student ID**, **email address** and the **assignment identifier** (eg. "HW 3") as text.

- 1) A nail is driven into a wall. A circular hoop of mass M_1 and radius R is hung over the nail so that it is free to swing back-and-forth (without friction) in a plane parallel to the surface of the wall . A small bead of mass M_2 is attached around the wire that makes up the hoop - it is free to move along the hoop without friction.

  In a single cell...

  - *i*) In addition to starting the cell with an invocation of **Remove**[], you may want to use **SetAttributes**[] to help Mathematica come to grips with reality.
  - *ii*) Attach horizontal and vertical coordinate axes to the nail and obtain the Lagrangian for this system in terms of these coordinates.

- *iii*) Pick a more convenient set of coordinates, write the code to convert the original set to this new set, and **Print**[] the Lagrangian so-obtained.
- *iv*) Obtain the (differential) equations of motion in these new coordinates.
- *v*) Try to use **DSolve**[] to obtain the solution to the system of equations that describe your system. You will probably discover that the information returned by **DSolve**[] is a bit less than you'd hoped for. What does that mean? (Your expectations were too high? No, I mean, in practical terms).
- *vi*) To use **NDSolve**[], you will have to provide (reasonable) numerical values for all the relevant parameters. Give it a try and see if you can generate plots for x and y as functions of time. The result isn't as exciting as the fact that you've already laid most of the groundwork in your code.
- *vii*) Use **ParametricPlot**[] to show the trajectory of the center of the hoop and the trajectory of the bead. You will have to make some reasonable choices for the physical parameters (don't forget initial conditions!).
- *viii*) An animation would be *really* cool. Maybe even necessary, if you want to understand the ParametricPlot you just obtained. Give it a shot! (**Graphics**[] has ready-made circles and points)

- 2)
  - a) Define the following function: $osc = \ddot{x}(t) + 2\beta\dot{x}(t) + \omega^2 x(t)$
  - b) Solve the differential equation $osc = 0$ for $x(t)$ subject to the initial conditions $\dot{x}(0) = 0$ and $x(0) = 1$
  - c) Plot $x(t)$ from $t = 0$ to $t = 5$ for the following cases: *i*) $\omega = 2\pi$, $\beta = 0$ *ii*) $\omega = 2\pi$, $\beta = 4\pi$, and *iii*) $\omega = 2\pi$, $b = 2\pi$. Discuss the significance of each case.

  Tip: **Plot** likes to complain about unplotable points. Usually this is caused by vacant assignments - variables in the plotted expression that haven't been given a numeric value. Sometimes, you can have everything right and **Plot** will still complain. If this happens (and it probably will in this problem), *surround the expression to be plotted with the function* **Evaluate[...]** . If you execute **Plot[Evaluate**[$f[x]$]**,{**x**,**0**,**100**}**], Mathematica will obtain the points to be plotted by actually evaluating $f[x]$ first.

  Tip: The third case in part c will probably give you trouble. It has to do with an assumption Mathematica makes in its general solution. You will have to find a way to circumvent this assumption.

- 3) Plot $x(t)$ for a damped oscillator, using β as a tunable parameter in **Manipulate**. Take $A_0 = 1$, $V_{0x} = 0$ and $\omega_0 = 2$ and plot over $0 < t < 10$. Let β be tunable in steps of 0.1 over $0 < \beta < 4$.

  This is not as easy as it looks (is it ever?). You will probably run into issues as β passes through ω_0. The trick is going to be to see if you can get **DSolve** to re-evaluate with each new setting for β. Careful use of assignments and perhaps some smart argument sets should take care of this.

- 4) **Marion 3-35**. Not as bad as it looks. There are really only two temporal intervals, and Mathematica is going to do the hardest parts for you. Solve the first temporal interval as a driven oscillator, and use the relevant values at the end of that interval as initial conditions for the next interval. When you're done, take $\beta = \pi/2$, $\omega = \omega_0 = 2\pi$ and $a = 1$, and (on **one** plot), plot $x(t)$ over the range $0 < t < 8\pi/\omega_0$. Hint: **Show[...]**.