Physics 652: Assignment 4

(to be submitted by Thursday, April 11, 2024)

1. Consider a very general forced and damped harmonic oscillator, modelled as

$$ma = -kx - \eta v + f,$$

where the spring constant $k(t)$, the dissipation coefficient $\eta(t)$, and the external forcing term $f(t)$ all have arbitrary time dependence. For convenience, let's work in units where the mass has a value $m = 1$.

(a) Doubly integrate (from time 0 to time t)

$$\ddot{x} = \frac{d^2x}{dt^2} = -kx - \eta\dot{x} + f = \frac{d}{dt}(-\eta x) + (\dot{\eta} - k)x + f$$

to find the integral version of this differential equation. Assume that $x(0) = x_0$ and $\dot{x}(0) = v_0$ are known initial conditions. You should be able to derive a result of the form

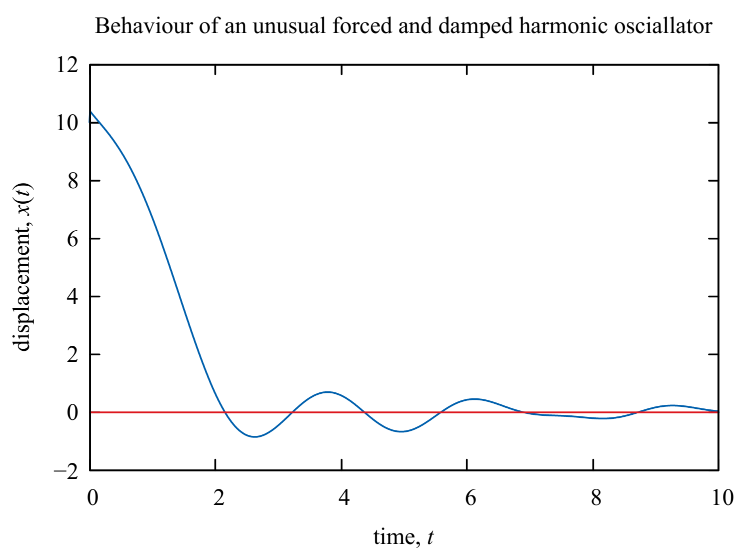
$$x(t) = x_0 + u_0 t + \int_0^t dt' \left[\lambda(t,t')f(t') + \kappa(t,t')x(t')\right].$$

(b) Sample the time on a uniform, discrete mesh and approximate the integral in the Simpson's-rule sense in order to show that

$$x_i = x_0 + \Delta t\left\{i(v_0 + \eta_0 x_0) + \sum_{j=0}^{i}\Big[\Delta t(i-j)f_j + \big[-\eta_j + \Delta t(i-j)(\dot{\eta}_j - k_j)\big]x_j\Big]\right\}$$

Here, $x_i = x(i\Delta t)$, $v_i = \dot{x}(i\Delta t)$, $f_i = f(i\Delta t)$, etc.

(c) Carry out the numerical integration with the model $f(t) = 3\cos 2t$, $k(t) = 1 + 2t$, and $\eta(t) = 4e^{-t/4}$ starting from $x(0) = 10$ and $v(0) = -1$. This is straightforward to implement in your favourite programming language. (I did it in Julia, but it's just as easy in Python, c++, or Mathematica; the calculation can even be implemented as an Excel spreadsheet.) Compute and plot the evolution of $x(t)$ in the interval $t \in [0, 10]$ with a step size of $\Delta t = 0.01$. You should be able to reproduce the following plot.



2. The nonlinear differential equation $u_t + uu_x = 0$, known as the inviscid Burgers' equation, is useful as a minimal model of a shock wave.

   (a) This equation has an unusual implicit solution. Try the Ansatz $u(x,t) = f(x - u(x,t)t)$. Show that

   $$u_t = \frac{-uf'}{1+tf'} \quad \text{and} \quad u_x = \frac{f'}{1+tf'},$$

   and hence that $u_t + uu_x = 0$.

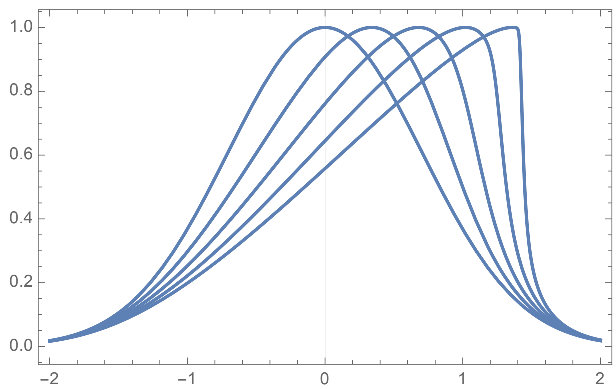
   (b) Argue that $dx/dt = u(x,t)$ is the local speed at every point in time and space.

   (c) Prove that every point in the field moves at a constant speed; i.e., $d^2x/dt^2 = 0$.

   (d) Suppose that the field is prepared in the initial state $u(x,0) = f(x) = e^{-x^2}$, which triggers a right-moving pulse. Determine the *overtaking time* or *breaking time* at which the pulse (whose shape changes) develops a vertical tangent and becomes multivalued.

   (e) Given the following five (approximated) field snapshots at times $t \approx 0.34, 0.68, 1.02, 1.36$, use your intuition and judgement to sketch the sixth and seventh. Explain what's happening in the `Do` loops and how that solves the differential equation.

```
f[x_] = Exp[-x^2];
u[x_] := f[x];
dt = 0.017;
p1 = Plot[u[x], {x, -2, 2}, Frame -> True];
Do[u[x_] = f[x - u[x]*i*dt], {i, 1, 20}];
p2 = Plot[u[x], {x, -2, 2}];
Do[u[x_] = f[x - u[x]*i*dt], {i, 21, 40}];
p3 = Plot[u[x], {x, -2, 2}];
Do[u[x_] = f[x - u[x]*i*dt], {i, 41, 60}];
p4 = Plot[u[x], {x, -2, 2}];
Do[u[x_] = f[x - u[x]*i*dt], {i, 61, 80}];
p5 = Plot[u[x], {x, -2, 2}];
Show[p1, p2, p3, p4, p5]
```



3. Use *Mathematica* to generate the Legendre polynomials via Rodrigues's formula and its explicit, Leibnitz-rule-based expansion. These expressions should agree with one another and with the output from the built-in `LegendreP` function. (This is for your own edification. There's nothing to submit for this question.)

```
RodriguesP[n_] := Simplify[D[(x^2 - 1)^n, {x, n}]/(2^n n!)]
RodriguesP[1]
RodriguesP[2]
RodriguesP[3]
CombP[n_] := Simplify[Sum[Binomial[n, k]^2 (x - 1)^(n - k) (x + 1)^k, {k, 0, n}]/2^n]
CombP[1]
CombP[2]
CombP[3]
LegendreP[1,x]
LegendreP[2,x]
LegendreP[3,x]
```

4. Complete each of the following (either by hand or with computational help):

   (a) Find the expressions for $P_5(x)$ and $P_7(x)$ and compute $\lim_{x\to 0} P_7(x)/P_5(x)$.

   (b) Demonstrate that
$$P_n(0) = \frac{\sqrt{\pi}(-2)^n}{\Gamma\left(\frac{1-n}{2}\right)\Gamma\left(\frac{n+2}{2}\right)}.$$

   (c) Report the values $P_5(+1)$, $P_5(-1)$, $P_{10}(+1)$, and $P_{10}(-1)$.

   (d) Evaluate the three integrals,
$$\int_{-1}^{1} dx\, P_5(x)P_{10}(x), \quad \int_{-1}^{1} dx\, P_5(x)P_5(x), \quad \text{and} \quad \int_{-1}^{1} dx\, P_{10}(x)P_{10}(x).$$

5. Expand the generating function $g(x,t) = (1 - 2xt + t^2)^{-1/2}$ out to seventh order. Confirm that each coefficient of t^n in the expansion is the function $P_n(x)$ for all powers $n = 0, 1, \ldots, 6$. (Again, there's no need to submit anything for this question.)

```
g[x_, t_] := (1 - 2 x t + t^2)^(-1/2)
expansion7 = Map[Simplify, Series[g[x, t], {t, 0, 7}]]
Coefficient[expansion7, t, 2]
LegendreP[2,x]
Coefficient[expansion7, t, 3]
LegendreP[3,x]
```

6. The Legendre polynomials form a complete basis for the space of smooth functions defined on $[-1, 1]$. Consider the function $f(x) = \sin(\pi x/2)\cos(2\pi x)$. It has an expansion

$$f(x) = \sum_{n=0}^{\infty} \frac{2n+1}{2} f_n P_n(x) \text{ with coefficients } f_n = \int_{-1}^{1} dx\, P_n(x) f(x).$$

Produce plots of the approximate functions $f^{[m]}(x) = \sum_{n=0}^{m}(n+1/2) f_n P_n(x)$ corresponding to expansions truncated at some finite order m for each of $m = 2, 3, \ldots 10$; devise a measurement of the discrepancy between $f(x)$ and $f^{[m]}(x)$ that you can plot versus m. It should converge to zero as m gets large.

```
f[x_] = Sin[\[Pi] x/2]*Cos[2*\[Pi] x]
Plot[f[x], {x, -1, 1}]
ff[x_] = Sum[(2n+1)/2*LegendreP[n,x]*Integrate[LegendreP[n,x]*f[x],{x,-1,1}],{n,0,7}]
Plot[{f[x], ff[x]}, {x, -1, 1}]
```

7. In *Mathematica*, write a new function `RodriguesP[l,m]` that computes the associated Legendre polynomial according to

$$P_{l,m}(x) = \frac{(1-x^2)^{m/2}}{2^l l!} \frac{d^{l+m}}{dx^{l+m}} (x^2-1)^l.$$

(See `https://mathworld.wolfram.com/SphericalHarmonic.html`.) Confirm that

$$Y_l^m(\theta, \phi) = (-1)^m \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}} P_{l,m}(\cos\theta) e^{im\phi}$$

is identical to the built-in function `SphericalHarmonicY[l,m,\[Theta],\[Phi]]`.