UNIVERSITY OF ALABAMA
Department of Physics and Astronomy

PH 125 / LeClair Spring 2009

Problem Set 5 Solutions

1. Problem 7.51 from your textbook

(a) We are given the initial and final position vectors $\vec{\mathbf{d}}_i$ and $\vec{\mathbf{d}}_f$ of an object acted upon by a force $\vec{\mathbf{F}}$. In this case, the force is constant, so work is simply calculated from the scalar product of the force and displacement:

$$W = \vec{\mathbf{F}} \cdot \Delta\vec{\mathbf{d}}$$

The displacement is just the difference between initial and final position vectors:

$$\Delta\vec{\mathbf{d}} = \vec{\mathbf{d}}_f - \vec{\mathbf{d}}_i$$

Since the scalar product is distributive,

$$W = \vec{\mathbf{F}} \cdot \Delta\vec{\mathbf{d}} = \vec{\mathbf{F}} \cdot \left(\vec{\mathbf{d}}_f - \vec{\mathbf{d}}_i\right) = \vec{\mathbf{F}} \cdot \vec{\mathbf{d}}_f - \vec{\mathbf{F}} \cdot \vec{\mathbf{d}}_i$$

With the vectors given,

$$\begin{aligned} W &= \left(3\,\hat{\boldsymbol{\imath}} + 7\,\hat{\boldsymbol{\jmath}} + 7\,\hat{\mathbf{k}}\right) \cdot \left(-5\,\hat{\boldsymbol{\imath}} + 4\,\hat{\boldsymbol{\jmath}} + 7\,\hat{\mathbf{k}}\right) \mathrm{N \cdot m} - \left(3\,\hat{\boldsymbol{\imath}} + 7\,\hat{\boldsymbol{\jmath}} + 7\,\hat{\mathbf{k}}\right) \cdot \left(3\,\hat{\boldsymbol{\imath}} - 2\,\hat{\boldsymbol{\jmath}} + 5\,\hat{\mathbf{k}}\right) \mathrm{N \cdot m} \\ &= -15 + 28 + 49 - (9 - 14 + 35)\,\mathrm{J} = 62 - 30\,\mathrm{J} = 32\,\mathrm{J} \end{aligned}$$

(b) Given the work done and the elapsed time Δt, power is readily calculated:

$$\mathscr{P} = \frac{W}{\Delta t} = \frac{32\,\mathrm{J}}{4\,\mathrm{s}} = 8\,\mathrm{W}$$

There is, of course, another way:

$$\mathscr{P} = \vec{\mathbf{F}} \cdot \vec{\mathbf{v}} = \vec{\mathbf{F}} \cdot \frac{\Delta\vec{\mathbf{d}}}{\Delta t} = \frac{1}{\Delta t}\vec{\mathbf{F}} \cdot \Delta\vec{\mathbf{d}}$$

You should be able to verify that this gives the same result.
(c) The angle θ between initial and final distances is most easily found from the scalar product:

$$\vec{\mathbf{d}}_f \cdot \vec{\mathbf{d}}_i = |\vec{\mathbf{d}}_f||\vec{\mathbf{d}}_i| \cos\theta$$

Rearranging,

$$\theta = \cos^{-1}\left[\frac{\vec{\mathbf{d}}_f \cdot \vec{\mathbf{d}}_i}{|\vec{\mathbf{d}}_f||\vec{\mathbf{d}}_i|}\right] = \cos^{-1}\left[\frac{-15 - 8 + 35}{\sqrt{25 + 16 + 49}\sqrt{9 + 4 + 25}}\right] \approx 78^\circ$$

2. Problem 8.27 from your textbook

(a) Since the stone is stationary, it has zero net force. The only force present are the object's weight and the spring force. Take $+y$ as upward, and let Δy be the distance the spring has been compressed from equilibrium *in total*, namely, $0.40\,\mathrm{m}$.

$$\sum F_y = k\Delta y - mg = 0 \qquad \Longrightarrow \qquad k = \frac{mg}{\Delta y} \approx 785\,\mathrm{N/m}$$

(b,c) The potential energy stored in the spring at the total compression distance of $\Delta y = 0.4\,\mathrm{m}$ must be

$$U_s = \frac{1}{2}k\left(\Delta y\right)^2 \approx 62.8\,\mathrm{J}$$

Conservation of mechanical energy dictates that this original potential energy must be the same as the total energy at the highest point.
(d) The maximum height h from the release point can be found by relating the original potential energy in the spring to the final potential energy of the mass:

$$K_i + U_i = \frac{1}{2}k\left(\Delta y\right)^2 = K_f + U_f = mgh \qquad \Longrightarrow \qquad h = \frac{k\left(\Delta y\right)^2}{2mg} \approx 0.8\,\mathrm{m}$$

3. Problem 8.21 from your textbook

(a) Let the y axis run vertically, with $+y$ upward and the origin at the *bottom* of the pendulum's motion. The pendulum then starts at rest a height L above the origin. Applying conservation of mechanical energy,

$$K_i + U_i = 0 + mgL = K_f + U_f = \frac{1}{2}mv_{\text{bott}}^2$$
$$\Longrightarrow \quad v_{\text{bott}} = \sqrt{2gL} \approx 4.85\,\mathrm{m/s}$$

(b) After reaching the bottom of its motion, the pendulum will hit the peg at a distance r above the origin. The string will wind around the peg, and the mass will reach a maximum height of $2r$. Applying conservation of energy from the bottom of the pendulum's motion to the point when the mass is at height $2r$, and noting $r = L - d$,

$$K_i + U_i = \frac{1}{2}mv_{\text{bott}}^2 + 0 = mgL = K_f + U_f = \frac{1}{2}mv_f^2 + mg\left(2r\right)$$
$$mgL - 2mgr = \frac{1}{2}mv_f^2$$
$$\Longrightarrow \quad v = \sqrt{2g\left(L - 2r\right)} = \sqrt{2g\left(2d - L\right)} \approx 2.43\,\mathrm{m/s}$$

4. Problem 8.63 from your textbook

Let the the y axis run vertically, with $+y$ upward, and the x axis horizontally, with $+x$ to the right. Let $y = 0$ be the level of the flat region.

Initially, we release the mass m from a height h. Since there is no friction on the curved surfaces, we

can use conservation of mechanical energy to find the mass' speed at the beginning of the flat surface:

$$K_i + U_i = 0 + mgh = K_f + U_f = \frac{1}{2}mv_{\text{bott}}^2$$
$$v_{\text{bott}} = \sqrt{2gh}$$

As the mass moves across the flat surface, which *does* have kinetic friction characterized by μ_k, energy will be transferred away from the block by friction. The amount of work done by the frictional force depends linearly on the distance traveled along the flat region.

If it still has nonzero velocity after reaching the end of the flat region, of length L, it will go up the right-hand curved surface, come back down again, and proceed along the flat surface in the opposite direction. Now we can use a slight trick. If the curved surface has no friction, the mass will go up the track to a height $h' < h$, and come back down *with exactly the same speed*. No mechanical energy is lost on the curved surface, so the curved ramp does nothing more than change the particle's direction. The same will happen when the mass gets back to the original curved surface – all that will happen is that the particle will change its direction, its speed will remain the same.

If that is the case, we just need to figure out how far in total the mass can travel under the influence of kinetic friction, d_{tot}. From that total distance, we only need to find how many times the mass can go a distance L (how many times back and forth along the flat surface), and the remainder tells us where the block stops within the flat region. In other words, we want to find d_{tot} *modulo*[i] L: $d_{\text{tot}} \mod n$.[ii]

The frictional force on the block is $f_s = \mu_k mg$, and thus the work done over a distance d is $W_f = \mu_k mgd$. An energy balance between the top of the first ramp and the block's final resting spot, noting $2h = L$ is then

$$K_i + U_i = K_f + U_f + W_f$$
$$0 + mgh = 0 + 0 + \mu_k mgd_{\text{tot}}$$
$$d_{\text{tot}} = \frac{h}{\mu_k} = \frac{L}{2\mu_k} = \frac{0.4\,\text{m}}{0.4} = 1\,\text{m}$$

This is the total distance the block can travel against the given frictional force. Since the block actually goes back and forth over a distance L, we want to find out how many multiples of L are congruent with d_{tot} and the remainder.[iii]. First, divide d_{tot} by L and ignore the remainder:

$$\text{number of transits} = \left\lfloor \frac{d_{\text{tot}}}{L} \right\rfloor = \left\lfloor \frac{1}{0.4} \right\rfloor = \left\lfloor 2.5 \right\rfloor = 2$$

The stopping position is the remainder:

$$\text{stopping position} = d_{\text{tot}} \mod L = d_{\text{tot}} - L\left\lfloor \frac{d_{\text{tot}}}{L} \right\rfloor = 1\,\text{m} - 0.4\,\text{m}\,(2) = 0.2\,\text{m}$$

In short, the block can travel 1 m against the frictional force, which means 2.5 trips back and forth across the flat region. During the third trip, it will stop halfway.

[i] See `http://en.wikipedia.org/wiki/Modulo_operation` for a short introduction to modular arithmetic.

[ii] one may also notate this $d_{\text{tot}} \% n$.

[iii] Here we will use the *floor* function, which for an argument x gives the greatest integer not greater than x. See. `http://en.wikipedia.org/wiki/Floor_function`

5. Problem 8.53 from your textbook

There is no friction until the mass reaches the right-most flat region, a height h above the starting point. Once motion on that flat region begins, the frictional force transfer energy away from the block. The magnitude of the frictional force is $f_s = \mu_k mg$, and the work done by the frictional force acting over a distance d is then $W_f = \mu_k mgd$. From the starting point until the point at which the block stops, we can balance the change in gravitational potential energy, the initial kinetic energy, and the work done by the frictional force.

$$K_i + U_i = K_f + U_f + W_f$$

Let the the y axis run vertically, with $+y$ upward, and the x axis horizontally, with $+x$ to the right. Let $y = 0$ be the level of the left-most flat region where the mass starts its motion, with initial velocity v_o.

$$\begin{aligned} K_i + U_i &= K_f + U_f + W_f \\ \frac{1}{2}mv_o^2 + 0 &= 0 + mgh - \mu_k mgd \\ 2\mu_k gd &= v_o^2 - 2gh \\ d &= \frac{v_o^2 - 2gh}{2\mu_k g} \approx 1.2\,\text{m} \end{aligned}$$

6. Problem 8.41 from your textbook

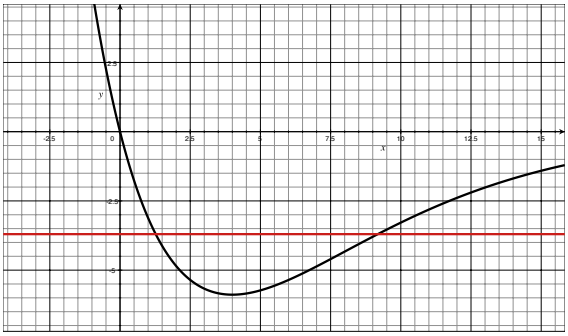
(a) The mechanical energy of the system at $x = 5\,\text{m}$ is just the given kinetic energy plus potential energy at 5 m, $U(5)$:

$$U(5\,\text{m}) = -4\,(5)\,e^{-5/4} \approx -5.73\,\text{J}$$

$$E_{\text{mech}}\Big|_{x=5\,\text{m}} = U(5\,\text{m}) + K = -5.73\,\text{J} + 2.0\,\text{J} = -3.7\,\text{J}$$

Technically, we should have been told that the $1/4$ in the exponent has units of meters, since overall exponents must be dimensionless.

(b) Here is a plot of $U(x)$. The solid red line shows the particle's total mechanical energy, E_{mech}.



(c,d) Here we are after the turning points of the particle. At these points, the particle's kinetic energy is zero, and thus its velocity must be zero, meaning the particle is changing directions ... hence the name. If $K=0$, then we must have $E_{\text{mech}}=U(x)$, the particle's energy is entirely potential.

$$\begin{aligned} E_{\text{mech}} &= U(x) \\ -3.73 &= -4xe^{-x/4} \end{aligned}$$

Either graphically (where the red line intersect the black curve above) or numerically, you should find $x=\{1.3, 9.1\}$.

(e,f,g) The maximum kinetic energy occurs when the potential is minimum, since $K = E_{\text{mech}} - U$. We don't really need to do this part graphically: we can maximize K analytically, given a total energy.

$$\begin{aligned} & K = E - U = E - 4xe^{-x/4} \\ & \frac{dK}{dx} = -\frac{dU}{dx} = F = -4e^{-x/4} - 4x\left(\frac{-1}{4}e^{-x/4}\right) = (x-4)\,e^{-x/4} = 0 \\ \implies \quad & K_{\text{max}} \quad \text{at} \quad x = 4.0 \\ & K_{\text{max}} = K(4\,\text{m}) = E - U(4\,\text{m}) \approx 2.2\,\text{J} \end{aligned}$$

From the graph, this is clearly a minimum, and the position seems correct. We have also found the force for free:

$$F = (x-4)\,e^{-x/4\,\text{m}}$$

(h) From the form of F given above, it is clear that $F=0$ only at $x=4.0$ and as $x \to \infty$.

7. You're driving your car on the highway at 75 mph, and you notice a sign that says you are 75 miles from your destination. So if you continue driving at that speed, you'd be there in an hour. But, you're not going to do that, because then it wouldn't be a puzzler. When you have driven one mile and you are now 74 miles from your destination, you drop your speed down to 74 mph.

So, you drive that first mile at 75 mph; when you are 74 miles from your destination, you drop your speed down to 74 mph; and then 73 mph, 72 mph ... and so on. Until, finally, you get down to 1 mile from your destination and you're driving at one mile per hour.

And the question is, if you do this, how long is it going to take you to travel the entire 75 miles?

This was a problem originally posted on the radio show *Car Talk*, you can find their response here: `http://www.cartalk.com/content/puzzler/transcripts/200642/answer.html`
Incidentally, the hosts Tom and Ray Magliozzi both have degrees from MIT. Anyway: we can figure this out mile-by-mile. At constant speed, the time t taken to travel a distance d at velocity v is just

$$t = \frac{d}{v}$$

If we traveled a total distance of 75 mi at 75 mi/h, the time would simply be

$$t_o = \frac{75\,\text{mi}}{75\,\text{mi/h}} = 1\,\text{h}$$

Of course, the problem presented is not this simple. For the first mile, we travel 1 mi at 75 mi/hr, for the second, we travel 1 mi at 74 mi/h, and so on. This leads to a series of times for each mile, the sum of which is the total trip time. After mile n, our speed reduces from v_n to $v_n - 1$, or from $75 - n$ to $75 - n + 1$ miles per hour.

$$t_{\text{tot}} = t_1 + t_2 + \ldots + t_{75} = \frac{1}{v} + \frac{1}{v-1} + \ldots \frac{1}{1} = \sum_{k=75}^{1} \frac{1}{v-k} = \sum_{k=1}^{75} \frac{1}{k}$$

Looking at the series we end up with, it is nothing more than the harmonic series $1/n$ summed from $n = 1$ to $n = 75$. There is no closed-form solution for the harmonic series through n terms, but there is a nice approximation:

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k} = \ln n + \gamma$$

Here $\gamma \approx 0.57721$ is the Euler-Mascheroni constant. For sufficiently large n, we can approximate the harmonic sum:

$$\sum_{k=1}^{n=75} \frac{1}{k} \approx \ln k + \gamma \approx 4.3175 + 0.57721 \approx 4.895$$

Thus, the trip should take approximately 5 hours. One can also sum this series by brute-force, ideally using a computer (see code examples below). The exact result is:[iv]

$$\sum_{n=1}^{75} \frac{1}{n} = \frac{6707589817681415714496242622181 33}{136851726813476721146087646859200} \approx 4.901355630553047$$

The approximation is good in this case to about 1%, and it gets better for larger series. The trip takes just under 5 hours.

[iv] Note that the LISP program in the next section is unique will actually output this improper fraction. An elegant weapon indeed. `http://xkcd.com/297/`

Randomness

I was bored on Saturday, so I whipped up solutions to this problem in various programming languages for comparison. I made the C versions a bit fancier, taking command line arguments and such, but all versions below work: standard C (iterative and recursive), LISP, pascal, fortran, java, perl, postscript, python, and even a bash shell script.

Postscript was probably the most interesting to figure out. Since it is mostly for talking to printers, it is wildly inefficient for the current task, but it does work. Since it is stack-based, it also looks much different than the other languages. The output is very nice looking, however. Really - copy and paste the code below into a file and save it as (say) "harmonic.ps." Open that file in, e.g., Preview, Acrobat Distiller, Ghostview and you'll have a nicely printed table of the harmonic series.

C:

```
//sums the partial harmonic series H(n) = 1 + 1/2 + ... + 1/n
//usage: pass "n" as a command line argument
//e.g., harmonic 75 <enter>
//returns H(N), the sum of the first N terms
//BY THE WAY: the input is restricted to N<=1e5
//(c) P. LeClair 2009

#include <stdio.h>

int main (int argc, const char * argv[])
{
        int i=0, DIST;
        double t=0;

        if (argc!=2) {
                fprintf(stderr,"\nUsage: %s NUM\n\n",*(argv));
                return(-1);
        }

        else DIST= atoi(*(argv+1));

        if (DIST>1e5) {
                printf("N out of range; try a number less than 1e5.\n");
                return(-1);
        }

        printf("\nPartial sums of the harmonic series through n=%i terms:\n",DIST);

        do      {
                t+=1.0/((double) ++i);
                printf("n=%i, H(%i)=%g\n",i,i,t);
        }
        while(i<DIST);

        return(0);
}
```

C (recursive):

```
//sums the partial harmonic series H(n) = 1 + 1/2 + ... + 1/n
//usage: pass "n" as a command line argument
//e.g., harmonic 75 <enter>
//returns H(N), the sum of the first N terms
//BY THE WAY: the input is restricted to N<=1e5
//(c) P. LeClair 2009

#include <stdio.h>

double sum(double N)
{
        if (N == 0) return 0;
        return 1.0/N + sum(N - 1.0);
}

int main (int argc, const char * argv[])
{
        int i=0, DIST;
        double t=0;

        if (argc!=2) {
                fprintf(stderr,"\nUsage: %s NUM\n\n",*(argv));
                return(-1);
        }

        else DIST= atoi(*(argv+1));

        if (DIST>1e5) {
                fprintf(stderr,"N out of range; try a number less than 1e5.\n");
                return(-1);
        }

        printf("\nPartial sum of the harmonic series through n=%i terms:\n",DIST);
        printf("%g\n",sum(DIST));

        return(0);
}
```

Bash script:

```
#!/bin/bash
#simple bash script to calculate first 75 terms of the harmnoic series, approx
#since bash only does integer math, we make some allowances
#multiply everything by 1e6 and compute sum in millionths :-)
count=1
sum=0
while [ $count -lt 76 ] ; do
 sum=$(( $sum + 1000000/$count ))
 count=$(( $count + 1 ))
done
sum=$sum/1000000
echo "Sum = $sum"
```

Common LISP:

```
(defun sum (N)
  ( loop for i from 1 to N
         for sum = 1 then (+ sum (/ 1 i))
         finally (return sum)))

(format t "Sum␣of␣harmonic␣series␣through␣75␣terms␣is␣~A~%" (sum 75))
```

Postscript:

```
%!PS
/LM 72 def                                  %define left margin
/Times-Roman findfont 8 scalefont setfont
/nstr 7 string def

/newline
{ currentpoint 8 sub                        % move y down by 8 points
exch pop                                    % drop old x coordinate
LM exch                                     % replace it with left margin
moveto} def                                 % go there

/harmonic
{    /num 0 def
      /current 1 def                        % loop counter
     {/num 1 current div num add def        % add 1/current to running sum
     /current current 1 add def             %incr counter
   } repeat
   num
} def

/prt-n                                      %convert given value to a string
{ nstr cvs show } def                       %then print it

/prtharmonic                                %given N, calc first N harmonic sums
{ dup prt-n                                 %print N =
( = ) show
harmonic prt-n%                             %get current partial sum & print it
newline                                     %along with a newline
} def

% ---------- Program ----------
LM 700 moveto                               %go to upper left corner of page
(Partial sums of the first 75 terms of the harmonic series:)show
newline
newline
1 1 75 {prtharmonic} for                    %show every term from 1 to 75
showpage
```

Python:

```
print '\nSum of the first 75 terms of the harmonic series 1/n.'

sum=0

for i in range(1,76):
    sum = sum + 1/float(i)

print 'sum: ' + str(sum)
```

Java:

```
// sum first N terms of harmonic series
// compile: javac harmoic.java
// run: java harmonic N

import java.io.PrintWriter;

public class harmonic {

  public static void main(String[] args) {
        // command line argument gives how many terms to sum over
        int N = Integer.parseInt(args[0]);

        double sum = 0.0;
        for (int i=1; i<=N; i++)
            sum += 1.0/i;

        System.out.println("The sum of the first " + N
                        + " terms of the harmonic series is "  + sum);

    }

}
```

Fortran:

```
PROGRAM HARMONIC
REAL :: SUM = 0
INTEGER :: I=1

DO I=1,75
    SUM = SUM + 1.0/I
END DO

WRITE (*,*) "Sum of harmonic series through 75 terms: ",SUM

END PROGRAM harmonic
```

Perl:

```
#!/usr/bin/perl
$sum=0;
for ($counter = 1; $counter <= 75; ++$counter) {
    $sum = ($sum + 1/$counter);
}

print "Sum of harmonic series through n=75: ", $sum,"\n";
```

Pascal:

```
program harmonic;

var
   i    : Integer;
   sum :  Real;

begin
    sum := 0;
    i := 1;
    repeat
       sum := sum + 1.0/i;
       i := i + 1;
    until (i=76);

    Write('Sum of first 75 terms of the harmonic series: ');
    Writeln(sum);

end.
```