Worksheet #6 – PHY102 (Spring 2011)

Generating and plotting lists of numbers,
"Do" loops and animation

We often need to ask the computer to do an operation many times. There are several ways of doing these "iterative" tasks in Mathematica. The two that you will use this week are **Table** and **Do**. Look these up in the online help. Especially if you are familiar with C++, you may also want to learn **For** at this time.)

This week you will also learn how to make plots based on lists of numbers, using **ListPlot** and **ListPlot3D**.

Finally, *animation* is very simple to do in Mathematica. You will learn the commands **Animate** and **Manipulate** to do this.

Problem 1. *(This problem really belongs to the last week's worksheet, but that worksheet was in danger of becoming too long; and anyway, reviewing is a good thing!)*

You lift a box of mass 30 kg vertically to a height of h meters by sliding it up an inclined plane that makes an angle θ with respect to the horizontal.

(i) Assuming there is no friction, use Mathematica to evaluate the integral $\int \vec{F} \cdot d\vec{r}$ to calculate the work done, and check that the result agrees with the change in gravitational potential energy.

(ii) Now include a frictional force $f = c\,x\,N$, which increases as the box moves up the plane. Here x is the distance measured along the inclined plane, N is the force normal to the inclined plane, and c is a constant. How much additional work does it now take to slide the box to the top of the ramp? Solve this problem analytically using Mathematica. (In most physics courses, the coefficient of friction is assumed to be a constant. But in this problem it is not: here the coefficient of friction is $\mu = c\,x$, i.e., the inclined plane is very smooth at the bottom, but it gets gets progressively stickier, the farther up you go.)

Problem 2.

(i) ListPlot plots a list of numbers on the y axis of a graph. To see how this works, enter the following code

```
    sintable=Table[Sin[x], {x,  0,  20,  .1 } ]
ListPlot[sintable]
```

(ii) Three dimensional plots are just as easy. Enter and run the following code

```
sintable3D=Table[Sin[x*y],{x,0,4,.1},{y,0,4,0.1}]
ListPlot3D[sintable3D]
```

Notice that if you left-click on the pretty picture, you can rotate the point of view!

(iii) Using the Table function, generate points to represent a circle, and plot these data using ListPlot. You can use the qualifier **AspectRatio** $\rightarrow$ **1** in ListPlot to force it to choose equal axes, so that your circle looks round. (**AspectRatio** $\rightarrow$ **Automatic** seems to work also.)

Problem 3.

Here is some code that computes the sum of the first n integers, with n running from 1 to 100. (The **Range** command is used here to set up an array to store the sums. It also sets the elements of that array to the integers, but that aspect of **Range** is not relevant here, because each element of array is overwritten inside the loop.)

```
sumintegers=Range[100];
sumintegers[[1]]=1;
Do[
{sumintegers[[n]]=sumintegers[[n-1]]  +  n},
{n,2,100,1}
]
ListPlot[sumintegers]
```

The Riemann zeta function is defined by $\zeta(p) = \sum_{n=1}^{\infty} 1/n^p$. This sum converges for all $p > 1$ (Why? – Hint: the behavior at large n is similar to the behavior of the integral $\int dx/x^p$ at large x).

Write a program to find the approximate value of $\zeta(p)$ as a function of the number of terms, N, that are included in the sum. Plot the value of this sum for $p = 3$ as a function of N. How many terms do you need to take (i.e., how big does N need to be) to get the approximation correct to 4 digit accuracy? Note, you will have to make up an alternative name for this N, because Mathematica already uses `N` for its numerical value function.

Problem 4. First type in and run the following code, which animates circular motion:

```
tstep = 0.05*Pi;

x = Cos[t];

y = Sin[t];

Animate[ParametricPlot[{x, y}, {t, tstart, tstart + tstep}, PlotRange -> {{-1, 1},
{-1, 1}}] , {tstart, 0, 2*Pi}]
```

(Another useful command is illustrated by replacing "Animate" by "Manipulate" in the above code. This generates a graph with a mouse-controlled slider to vary the parameter.)

Modify this code to animate the following projectile motion problem: A mass of 20 kg is fired from a height of 2000 meters, with initial angle above the horizontal of 60 degrees and initial speed of 500m/s (ignore drag). Your animation should begin at firing and end when the mass hits ground level.