MIT OpenCourseWare
http://ocw.mit.edu

12.010 Computational Methods of Scientific Programming
Fall 2008

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.

12.010 HW04 2008

- **Question (1): (25-points)**
  (a) Write, compile and run a Mathematica program that generates two tables of the Gamma function. The Gamma function satisfies the following equation

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$$

where z is the argument of the gamma function and can be complex. (See (See http://mathworld.wolfram.com/GammaFunction.html for information on the Gamma function). Gamma functions are rarely computed by directly integrating the equation above. Generally they are evaluated by series expansions.
For one table, table will be generated for z between 1 and 10 in increment of 1 when z is an integer. This table should give values of Γ(z), Γ(z+1/3), Γ(z+1/2) and Γ(z+2/3).
For the second table, Γ(z) should be generated for z between -1 and +1 in increments of 0.1.
Results should be tabulated with at least 6-significant digits.

The submission should include
(a) A discussion of the algorithms used in the program with rationales for the choices (in this case Mathematica has a built in gamma function)
(b) The tables generated by the program and
(c) The Mathematica Notebook.

- **Solution: Given that the gamma function is defined in *Mathematica*, the most tricky part of the this problem is formarting the output. There are many ways to approach the output problem. Basic formated table output is set with the TableForm function. In the solution we are using the Append function. Table can also be used for this. Using the Print command means the results are printed in a one cell.**

In[182]:=
```
Off[General::spell1]; (* Stops message about head looking like a command*)
z := {n, n + 1 / 3, n + 1 / 2, n + 2 / 3}; args = Range[1, 10];
head = {"gamma(n)", "gamma(n+1/3)", "gamma(n+1/2)", "gamma(n+2/3)"};
lall = {};
For[n = 1, n <= 10 , n++,
 {l1 = N[Gamma[z]],
  lall = Append[lall, l1]}
]
full = Insert[lall, head, 1];
Print["Table of Gamma Functions for positive n integers"]

Print[PaddedForm[TableForm[lall, TableHeadings → { args, head}, TableAlignments → Right], 7]]
(* Now do the fraactioal Gamma table*)
fargs = Range[-1.0, 1.0, 0.1];
fhead = "Gamma(z)";
Print["\nTable of Gamma Functions for non-integer arguments"]

(*Print[PaddedForm[
  TableForm[Gamma[fargs],TableHeadings →{fargs,fhead},TableAlignments→Right],7]]*)
Print[PaddedForm[TableForm[Transpose[{Gamma[fargs]}],
   TableHeadings → {fargs, {fhead}}, TableAlignments → Right], {8, 7}]]
```

Table of Gamma Functions for positive n integers

	gamma(n)	gamma(n+1/3)	gamma(n+1/2)	gamma(n+2/3)
1	1.	0.8929795	0.8862269	0.9027453
2	1.	1.190639	1.32934	1.504575
3	2.	2.778158	3.323351	4.012201
4	6.	9.260528	11.63173	14.7114
5	24.	40.12896	52.34278	68.65322
6	120.	214.0211	287.8853	389.0349
7	720.	1355.467	1871.254	2593.566
8	5040.	9940.091	14034.41	19884.01
9	40320.	82834.09	119292.5	172328.1
10	362880.	773118.2	1.133278×10^{6}	1.665838×10^{6}

Table of Gamma Functions for non-integer arguments

	Gamma(z)
-1.0000000	ComplexInfinity
-0.9000000	-10.5705640
-0.8000000	-5.7385546
-0.7000000	-4.2736700
-0.6000000	-3.6969326
-0.5000000	-3.5449077
-0.4000000	-3.7229806
-0.3000000	-4.3268511
-0.2000000	-5.8211486
-0.1000000	-10.6862870
0.0000000	ComplexInfinity
0.1000000	9.5135077
0.2000000	4.5908437
0.3000000	2.9915690
0.4000000	2.2181595
0.5000000	1.7724539
0.6000000	1.4891922
0.7000000	1.2980553
0.8000000	1.1642297
0.9000000	1.0686287
1.0000000	1.0000000

- **Question (2): (25-points).**
  Write a Mathematica Notebook that reads a file containing text, determines
  (1) The average and root-mean-square (RMS) scatter about the mean of the number of characters per word and
  (2) The average and root-mean-square (RMS) scatter about the mean of the number of words per sentence. A sentence can end with a period or question mark.

  The text below is contained in the file Q2_text.txt.

  The basic analysis of spacecraft tracking data requires relating the position and velocity of the spacecraft to the position and velocity of the tracking system. The coordinate system used in spacecraft navigation is shown in Figure 1. The basic measurements are of r and its time derivative and sequences of these measurements, combined with knowledge of the tracking station location and equations of motions of the spacecraft, allow the position of

the spacecraft denoted here by distance r, right ascension, a, and declination, d, and its velocity to be determined as a function of time. In addition to knowing the coordinates of the spacecraft in an inertial coordinate system, the coordinates of solar system bodies are also needed in this frame. Tracking data collected on spacecraft near planets can also be used to improve the ephemerides the planets through the gravitational perturbations of the spacecraft motions. Large combined analysis of tracking data and direct measurements of planets (radar and optical positions) are used to generate planetary ephemeredes.

- **Solution: The solution this case involves a number of different aspects of *Mathematica*. First we need to open a file and read it.**

In[194]:=
```
(* The SetDirectory command needs to be changed for specific cases
 or start Mathematica in the directory where the fille is located*)
SetDirectory["/Users/tah/TAH_docs/12.010/HW_2008/HW02_soln/"];
file = "Q2_text.txt";
lst = Import[file]
```

Out[196]=
```
The basic analysis of spacecraft tracking data requires
relating the position and velocity of the spacecraft to the
position and velocity of the tracking system.  The
coordinate system used in spacecraft navigation is shown in
Figure 1.   The basic measurements are of r and its time
derivative and sequences of these measurements, combined
with knowledge of the tracking station location and
equations of motions of the spacecraft, allow the position
of the spacecraft denoted here by distance r, right
ascension, a, and declination, d, and its velocity to be
determined as a function of time.  In addition to knowing
the coordinates of the spacecraft in an inertial coordinate
system, the coordinates of solar system bodies are also
needed in this frame.  Tracking data collected on
spacecraft near planets can also be used to improve the
ephemerides the planets through the gravitational
perturbations of the spacecraft motions.   Large combined
analysis of tracking data and direct measurements of
planets (radar and optical positions) are used to generate
planetary ephemeredes.
```

- **Having read in the file, we need to do some counting. The first step is decide how to handle numbers. The usenum variable is set True to use numbers, and False not to use them. When not used, all the numbers are replaced by spaces. Because of the way ReadList works, we need one version of the text that also removes the periods. (If this is not done, the period is treated like a word)**

  The characters in word count is done by extracting the list of words in the text. The total number of words is just the Length of this list.
  The ReadList line separates each word and then StringCount counts number of letters in a-z range.
  Mean and StandardDeviation are used to compute the corresponding quantities.

  The sentences need to be handled slightly differently. Here we find sentences as those elements that are separated by ".","?" or "!". The Split functions breaks the sentences into separate strings that can be accessed with the Part function. Lists are built that contain the number of words in each sentence and Mean and StandardDeviation are used to compute the corresponding quantites. Print is use to output the results.

In[197]:=
```
(* To aviod counting 1. as a word we should remove and number period combinations *)
usenum = True; (* Count numberic values as words, False not counted*)
uselstp = If[ usenum ,
   lst, StringReplace[lst, {".", RegularExpression["[0-9]"]} -> " "]];
uselstn = If[ usenum ,
   lst, StringReplace[lst, RegularExpression["[0-9]"] -> " "]];

If[usenum,
 Print["12.010 HW04_02: In file " file " when numbers counted, There are:"],
 Print["12.010 HW04_02: In file " file " when numbers not counted, There are:"]]

wrdlist = ReadList[StringToStream[uselstp], Word]; (* Make string into list of words *)
(* wrdlist is a list of words and we can use this list to get total number,
mean number of characters per word, and the standard deviation (RMS about the mean)*)
totalWords = Length[StringLength[wrdlist]];
meanchars = N[Mean[StringCount[wrdlist, _?LetterQ]], 4];
rmschars = N[StandardDeviation[StringCount[wrdlist, _?LetterQ]], 4];
Print["Mean characters per word ", meanchars,
 " with RMS ", rmschars, " in ", totalWords " words;"]
(* Now do sentences*)
sentences = StringSplit[StringJoin[" ", uselstn], "."];
numsent = Length[sentences];
splitSen = Split[sentences];
wrdcount = {};
For[is = 1, is  < numsent, is++,

 numword = StringCount[Part[splitSen, is], Whitespace];
 wrdcount = Append[wrdcount, numword];
]
meanwrds = N[First[Mean[wrdcount]], 4];
rmswords = N[First[StandardDeviation[wrdcount]], 4];
Print["Mean words per sentence  ", meanwrds,
 " with RMS ", rmswords, " in ", numsent - 1, " sentences"]
```

```
12.010 HW04_02: In file  Q2_text.txt  when numbers counted, There are:

Mean characters per word 5.347 with RMS 3.172 in 167  words;

Mean words per sentence  27.83 with RMS 15.66 in 6 sentences
```

Question (3): (50-points) Write a Mathematica program that will compute the motions of a set of particles undergoing mutual gravitational attraction. The program should generat the trajectories of each of the particles with an error tolerance that is proportional to the separation of the particles. The program should be able to handle large numbers of particles and thought should be given as to how to input the initial positions and velocities of particles when there are a large number of particles.

As a test of your program: Evaluate the trajectories of the 6 particles below with an error tolerance of 1.e-6. (This case is similar to the collision of two Sun-Earth-Moon systems) The integration should be run for 515-days and the positions and velocities at the end of 515 days should be included in the output.

The values below give the mass (kg), X and Y position (km) and X and Y velocities (km/s) of the 6 particles to be evaluated.
2.0e+30 kg XY 0.0e+00 0.0e+00 (km) VXY 0.000e+00 0.000e+00 (km/sec
8.8e+28 kg XY 1.5e+08 0.0e+00 (km) VXY 0.000e+00 2.857e+01 (km/sec
7.3e+22 kg XY 1.5e+08 1.0e+07 (km) VXY -2.424e+01 2.857e+01 (km/sec
2.0e+30 kg XY -1.0e+09 0.0e+00 (km) VXY 1.000e+01 0.000e+00 (km/sec
8.8e+28 kg XY -8.5e+08 0.0e+00 (km) VXY 1.000e+01 2.857e+01 (km/sec
7.3e+22 kg XY -8.5e+08 1.0e+07 (km) VXY -1.424e+01 2.857e+01 (km/sec)

Your answer to this question should include:
(a) The algorithms used and the design of your program
(b) The Mathematica program Notebook (I will compile and run your programs). If yo program does not run or takes more than few minutes to run, let me know so that I will treat it with caution.
(c) The results from the test case above with positions and velocities at the end of 515 days. You could also explore the effects of changing the accuracy tolerance on the results.
(d) A plot of the final locations of the bodies and an optional plot of there trajectories.

- **Solution:**
  First we read in the body information from the NBody.dat file. Import is used to read the data and the separate elements such as mass, xpi, ypi, xvi and yvi are separated using the Part and Table functions. The first line is a header and ignored
  (Non-numeric lines can be detected with test Head[Part[Part[alldata,i],1] == Real.)

In[213]:=
```
SetDirectory["/Users/tah/TAH_docs/12.010/HW_2008/HW02_soln/"];
file = "NBody.dat";
alldata = Import[file];
(* Divide up the data by Body.  File have one header line *)
nb = Length[alldata] - 1;
mass = Table[Part[Part[alldata, i], 1], {i, 2, nb + 1}]; (* Mass g *)
xpi = Table[Part[Part[alldata, i], 2], {i, 2, nb + 1}]; (* X pos initial km *)
ypi = Table[Part[Part[alldata, i], 3], {i, 2, nb + 1}]; (* Y pos initial km *)
xvi = Table[Part[Part[alldata, i], 4], {i, 2, nb + 1}]; (* X vel initial km/s *)
yvi = Table[Part[Part[alldata, i], 5], {i, 2, nb + 1}]; (* Y vel initial km/s*)
```

- **Now define the functions that we will need for solving the gravitonal problem**
  Here dacx and dacy define the contributions to the accelerations from body 2 numbered m2 at Body 1. Notice in dis function that computed distance, we all a small number. With this addition, the force of a body on itself is zero (due to coordinate differences being zero) Without the small number, the force is undefined because 0/0 is generated.

In[222]:=
```
Guniv = 6.67428×10^(-20); (* Set constant*)
 (* Distance between points*)
dis[xp1_, yp1_, xp2_, yp2_] := Sqrt[(xp1 - xp2)^2 + (yp1 - yp2)^2] + 10^-9;
dacx[xp1_, yp1_, xp2_, yp2_, m2_] := Guniv * mass[[m2]] * (xp2 - xp1) / dis[xp1, yp1, xp2, yp2]^3;
dacy[xp1_, yp1_, xp2_, yp2_, m2_] := Guniv * mass[[m2]] * (yp2 - yp1) / dis[xp1, yp1, xp2, yp2]^3;
```

- **Now define the equations that will be solved. We have 2*nb second order differential equations (nb in x, nb in y) called eqnx and eqny. These are formed by Sum. Mathemathica is smart to remove the zero, body on itself contribution.**

In[225]:=
```
eqnx = Table[
   xp[j]''[t] == Sum[dacx[xp[j][t], yp[j][t], xp[i][t], yp[i][t], i], {i, 1, nb}], {j, 1, nb}];
eqny = Table[yp[j]''[t] == Sum[dacy[xp[j][t], yp[j][t], xp[i][t], yp[i][t], i], {i, 1, nb}],
   {j, 1, nb}];
```

- **Now define the boundary counditions. There are 4nb of these, Also define the variables to solved for varx and vary**

In[227]:=
```
bcxp = Table[xp[i][0] == xpi[[i]], {i, 1, nb}];
bcyp = Table[yp[i][0] == ypi[[i]], {i, 1, nb}];
bcxv = Table[xp[i]'[0] == xvi[[i]], {i, 1, nb}];
bcyv = Table[yp[i]'[0] == yvi[[i]], {i, 1, nb}];
varx = Table[xp[i], {i, nb}];
vary = Table[yp[i], {i, nb}];
```

- **Now set up the NDSolve solution. Here we set AccuracyGoal → 9, PrecisionGoal → 9, WorkingPrecision → 16, **
  MaxSteps -> Infinity to ensure that we get the needec accuracy. These goals are not specified the reaults differ from those generated by C and Fortran. (The commented line shows default run).
  There is a precision warning issued because of the number of deciminal places used in the inputs. Adding zeros to make the numbers more precise may make this message go away (not tested). The precw message is not present when the default accuracy case in run.

In[233]:=
```
dur = 1000; (*Length of integration in days*)
(*soln = NDSolve[{Join[eqnx,eqny],Join[bcxp,bcyp,bcxv,bcyv]},
   Join[varx,vary],{t,0,86400*dur}, WorkingPrecision → 10];*)
soln = NDSolve[{Join[eqnx, eqny], Join[bcxp, bcyp, bcxv, bcyv]},
  Join[varx, vary], {t, 0, 86 400 * dur}, AccuracyGoal → 9,
  PrecisionGoal → 9, WorkingPrecision → 16, MaxSteps -> Infinity];
```

NDSolve::precw : The precision of the differential equation

$$\left(\left\{\text{xp[1]}''[t] == \frac{5.87337\times 10^{9}\,(-\text{xp[1][t]}+\text{xp[2][t]})}{\left(\frac{1}{1000000000}+\sqrt{\text{Plus[≪2≫]}}\right)^{3}}+\frac{4872.22\,(-≪1≫+≪1≫)}{\left(\frac{1}{≪10≫}+≪1≫\right)^{3}}+\frac{≪1≫}{≪1≫}+\frac{≪13≫\,(≪1≫)}{(≪1≫\,≪1≫^{3}}+\right.$$

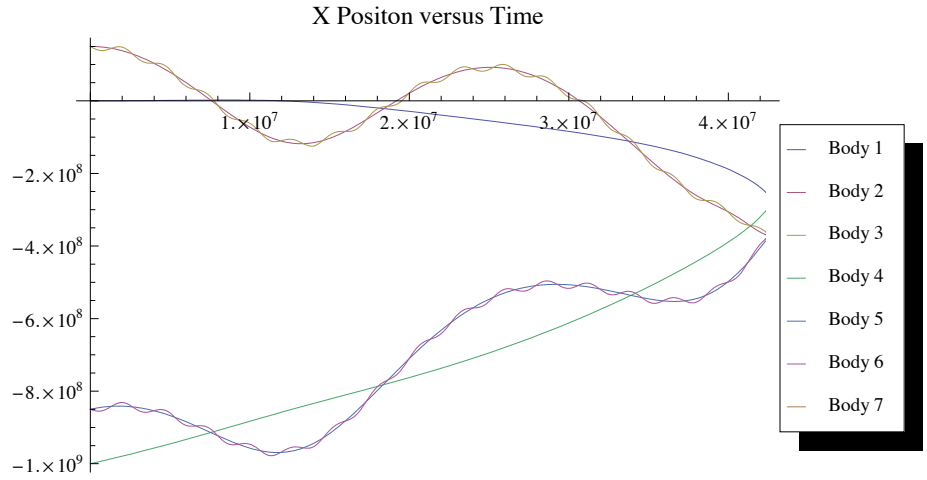
$$\left.\frac{4872.22\,(-\text{xp[1][t]}+\text{xp[6][t]})}{\left(\frac{1}{10\,≪6≫\,00}+\sqrt{\text{Plus[≪1≫]}}\right)^{3}},\ ≪9≫,\ ≪26≫\right\})$$

is less than WorkingPrecision (16.`). ≫

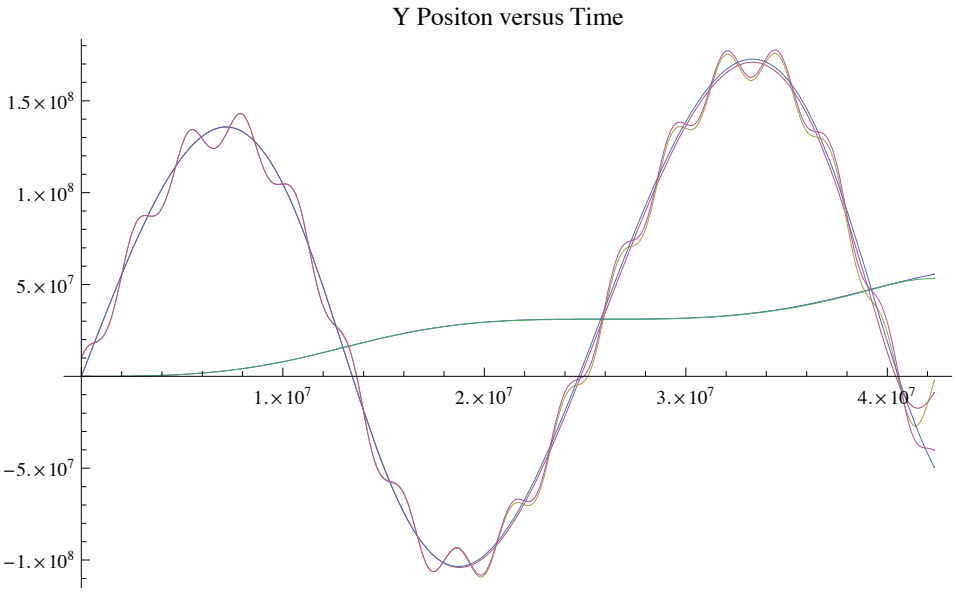
- **Now generate graphics: x and y plots as functions of time (the first with Legend) and then x versus y plot.**

In[234]:=
```
nday = 490;
Needs["PlotLegends`"];
Plot[Evaluate[Table[xp[i][t], {i, 6}] /. soln], {t, 0, 86 400 * nday},
 PlotLegend → {"Body 1", "Body 2", "Body 3", "Body 4", "Body 5", "Body 6", "Body 7", "Body 8"},
 LegendPosition → {1, -0.5}, PlotLabel → "X Positon versus Time"]
Plot[Evaluate[Table[yp[i][t], {i, 6}] /. soln],
 {t, 0, 86 400 * nday}, PlotLabel → "Y Positon versus Time"]
Plot[Evaluate[Table[xp[i]'[t], {i, 6}] /. soln],
 {t, 0, 86 400 * nday}, PlotLabel → "X velocity versus Time"]
Plot[Evaluate[Table[yp[i]'[t], {i, 6}] /. soln],
 {t, 0, 86 400 * nday}, PlotLabel → "Y velocity versus Time"]
ParametricPlot[Table[{First[Evaluate[xp[i][t] /. soln]], First[Evaluate[yp[i][t] /. soln]]},
  {i, 6}], {t, 0, 86 400 * nday}, PlotLabel → "X versus Y position"]
```

Out[236]=

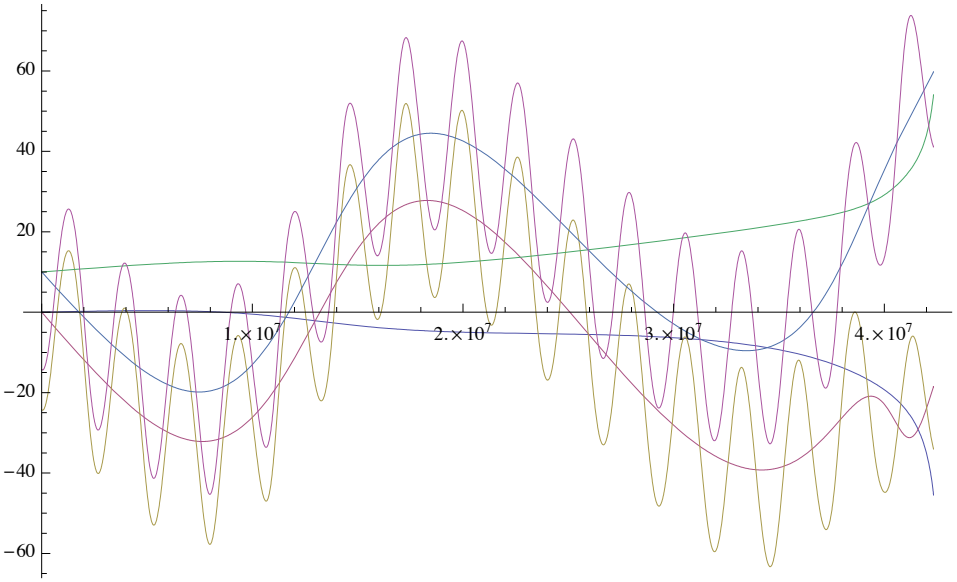


Out[237]=



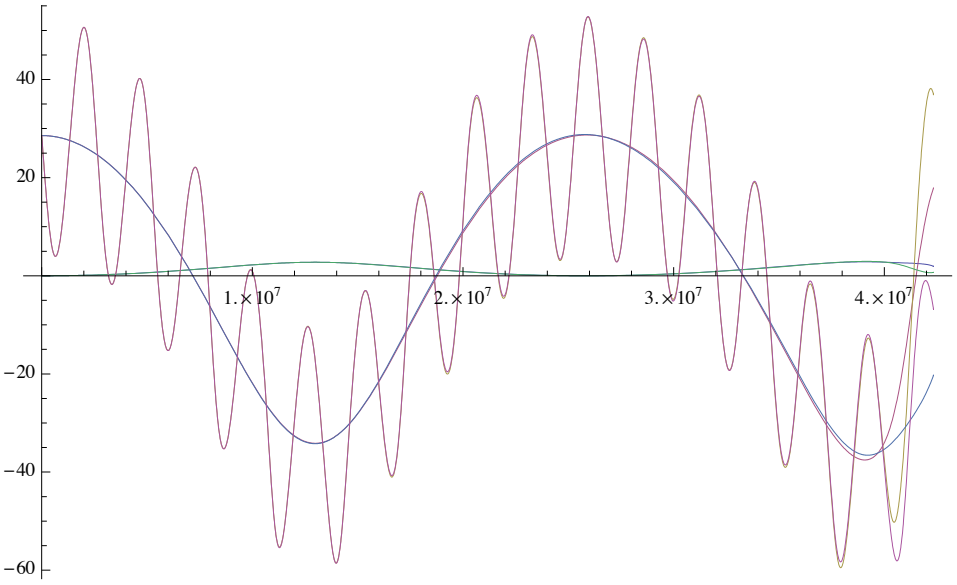
X velocity versus Time

Out[238]=



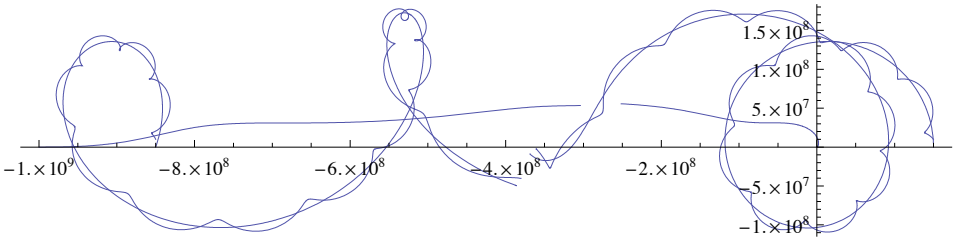
Y velocity versus Time

Out[239]=



X versus Y position

Out[240]=



In[241]:=
```
(* Generate Outputs at 515 and 490 days*)
nday = 515;
Print["12.010 HW04 Question 3\nResults at time ", nday, " days"]
For[i = 1, i <= nb, i++,
 Print["Body ", i, " PX PY ",
  First[Evaluate[xp[i][86 400 * nday] /. soln]], " ",
  First[Evaluate[yp[i][86 400 * nday] /. soln]], " VX VY ",
  First[Evaluate[xp[i]'[86 400 * nday] /. soln]], " ",
  First[Evaluate[yp[i]'[86 400 * nday] /. soln]]]
]
```

```
12.010 HW04 Question 3
Results at time 515 days

Body 1 PX PY -1.92707344515627×10^8 4.68022130985396×10^7 VX VY 31.374385144 -2.06187762508

Body 2 PX PY -3.55785137142781×10^8 5.7458565846332×10^7 VX VY -131.560900682 -87.855772845

Body 3 PX PY -3.48927453070772×10^8 5.1813750873396×10^7 VX VY 78.374854376 109.918973121

Body 4 PX PY -3.47201188224320×10^8 6.53128114888229×10^7 VX VY -18.1739724217 8.0931568005

Body 5 PX PY -2.43153363126753×10^8 -6.30237298745246×10^7 VX VY 68.824086093 7.9211745626

Body 6 PX PY -2.34462004757339×10^8 -6.4932094546093×10^7 VX VY 78.191599665 33.496022290
```

In[244]:=
```
nday = 490;
Print["12.010 HW04 Question 3\nResults at time ", nday, " days"]
For[i = 1, i <= nb, i++,
 Print["Body ", i, " PX PY ",
  First[Evaluate[xp[i][86 400 * nday] /. soln]], " ",
  First[Evaluate[yp[i][86 400 * nday] /. soln]], " VX VY ",
  First[Evaluate[xp[i]'[86 400 * nday] /. soln]], " ",
  First[Evaluate[yp[i]'[86 400 * nday] /. soln]]]
]
```

```
12.010 HW04 Question 3
Results at time 490 days

Body 1 PX PY -2.51555457795619×10^8 5.55984132891273×10^7 VX VY -45.449259661 1.9141995646

Body 2 PX PY -3.68758470167809×10^8 -8.901114171017×10^6 VX VY -18.4746363621 17.9021159007

Body 3 PX PY -3.60885635710179×10^8 -2.198626301723×10^6 VX VY -34.035992377 36.908456217

Body 4 PX PY -3.04046831809862×10^8 5.34148222302302×10^7 VX VY 54.071948400 0.70407727142

Body 5 PX PY -3.86012628734055×10^8 -4.9591321827816×10^7 VX VY 59.777127239 -20.2683851519

Body 6 PX PY -3.80336182278255×10^8 -4.0240384178693×10^7 VX VY 41.115354114 -6.8639276101
```