# Integrating Co-linear Helium Dynamics

A quick and dirty solution to this problem is to use mathematica. This should not be a replacement for doing it properly, but can be useful as a check. We want to find the dynamics defined by the Hamiltonian

$$H_{co} = \frac{1}{8}(x_2^2 p_1^2 + x_1^2 p_2^2) - Z R_{12}^2 + (x_1 x_2)^2 (1 + \frac{1}{R_{12}^2}) = 0 \quad (1)$$

where

$$R_{12} = (x_1^2 + x_2^2)^{1/2} \quad (2)$$

A simple mathematica programme that does this is

```
tmax=35  (* time of run *)
Z = 2  (* Let's do Helium *)

(* The Hamiltonian *)
H = (x2[t]^2 p1[t]^2 + x1[t]^2 p2[t]^2)/8 - Z (x1[t]^2+x2[t]^2) +
  (x1[t] x2[t])^2 (1+ 1/(x1[t]^2+x2[t]^2))

(* Hamilton's Equations *)
HE = {x1'[t] == D[H, p1[t]], x2'[t] == D[H, p2[t]],
p1'[t] == -D[H, x1[t]], p2'[t] == -D[H, x2[t]]}

(* Initial Conditions *)
IC = {x1[0] == 2.0, x2[0]==-1.0, p1[0]==2.0, H==0 /.t->0}

(* The Solution *)
Solution = NDSolve[Join[HE,IC], {x1[t], x2[t], p1[t], p2[t]},
{t, 0, tmax}, MaxSteps->10000]

(* Four Graphs *)
(* x1^2 versus x2^2 *)
ParametricPlot[Evaluate[{x1[t]^2, x2[t]^2} /. Solution], {t,0,tmax}]

(* The trajectory projected into the space x1, p1, x2 *)
Poincare = ParametricPlot3D[Evaluate[{x1[t], p1[t], x2[t]} /. Solution],
 {t,0,tmax}]

(* A Poincare Section of the Trajectory *)
Show[Poincare, PlotRange->{-0.01,0.01}]

(* Check the Hamiltonian to see how well the integration program does *)
Plot[Evaluate[H /. Solution], {t,0,tmax}, PlotPoints->5,PlotDivision->10]
```

Putting this code into a file "helium.m", say and running on mathematica using

```
In[0]:= << helium.m
```

will integrate the equations of motion. Note that there are two solutions to the initial conditions ($p_2[0]$ can be either positive or negative). This corresponds to either going forward or backwards in time. Mathematica knows that there are two solutions and computes both trajectories. Of course the two trajectories will look like one long trajectory. Mathematica uses a routine which estimates the error of the final point. Because this is a highly non-linear problem keeping this error small becomes rapidly harder and so the algorithm cannot calculate very long trajectories — this will not be the case in a simple Runge-Kutta routine which doesn't care about its error.

The program produces four graphs these are

1. A graph of $x_1^2$ versus $x_2^2$

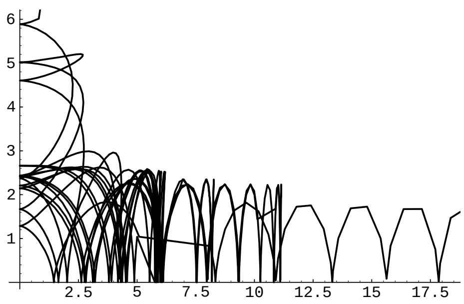


Figure 1: $x_1^2$ versus $x_2^2$

2. A graph of the trajectory projected into the space $x_1$, $x_2$ and $p_1$
3. A Poincaré section made by taking a cut in the trajectory at $x_2 = 0$
4. A plot of $H$ versus time. This is a measure of how well the integration is doing — if it was perfect $H$ would be constant. Note that there are two lines as there are two trajectories one going forward and one going backwards in time.

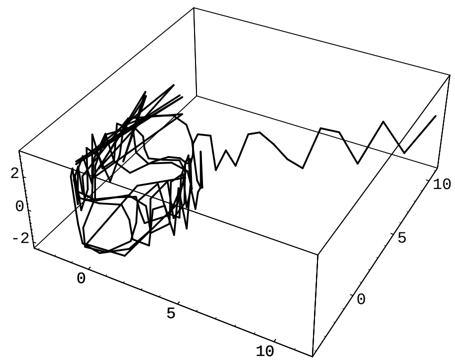


Figure 2: The trajectory projected into the space $x_1$, $x_2$ and $p_1$

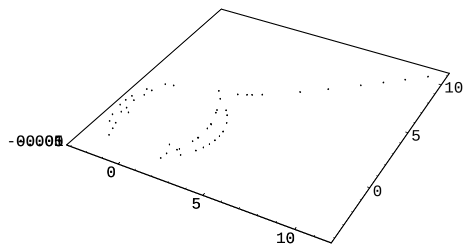


Figure 3: Poincaré section

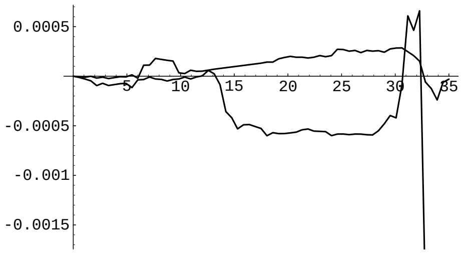


Figure 4: The Hamiltonian versus time.