# Scientific Computing for DPhil Students II
# Assignment 1 Solutions

Here are two solutions. First we do it in real arithmetic with standard Matlab, introducing $\theta(t)$ as an additional variable. Then we do it in complex arithmetic in Chebfun, deriving $\theta(t)$ via `unwrap(angle)`.

1. There is one moving particle here in the plane, which has a position and a velocity. So we can set the problem up as a first-order system with four variables, plus a fifth variable to track the angle $\theta(t)$. Here is my code. Writing it as `function a1()` enables me to put it all in a single file `a1.m`.

```
% a1.m  A particle orbiting around three fixed points
%       We use a system of 5 ODEs: positions x and y, velocities u and v, and angle theta

  function a1()
% Set up the fixed points and the moving point:
  fp = [cos(0)      sin(0)
        cos(2*pi/3) sin(2*pi/3)
        cos(4*pi/3) sin(4*pi/3)];
  x = [2 -2]; u0 = [x 0 0 atan(x(2)/x(1))]';   % initial conditions

% Solve the problem with ode113:
  tol = input('tol?  (for best accuracy take 3e-14) ');
  opts = odeset('reltol',tol,'abstol',tol);
  [tt,uu] = ode113(@p5fun,[0 40],u0,opts);

% Print the requested numbers:
  fprintf('Tol = %5.2e    Position at t=40: (%14.11f,%14.11f)\n',tol,uu(end,1),uu(end,2))

% Plot the orbit:
  FS = 'fontsize'; fs = 16; LW = 'linewidth'; lw = 1;
  subplot(1,2,1), plot(fp(:,1),fp(:,2),'.k','markersize',12)
  hold on, plot(uu(:,1),uu(:,2),LW,lw), grid on
  axis([-3 2.6 -2.7 3.3]), axis square
  set(gca,'xtick',-3:3,'ytick',-3:4)
  title('orbit',FS,fs)

% Plot the angle theta(t):
  subplot(1,2,2), plot(tt,uu(:,5),'r',LW,lw), grid on
  axis([0 45 -2 15]), axis square
  set(gca,'ytick',0:pi:4*pi)
  xlabel('time t',FS,fs), ylabel('\theta',FS,fs)
  title(sprintf('angle'),FS,fs)
  set(gca,'fontname','symbol')
  set(gca,'yticklabel',{'0' 'p','2p','3p','4p'})

  function v = p5fun(t,u)
  x = u(1:2);
  force = zeros(2,1);
  for i = 1:3
    f = fp(i,:)'; force = force + (f-x)/norm(f-x)^3;
  end
  vel = u(3:4);
  dthdt = vel'*[-x(2);x(1)]/norm(x)^2;
  v = [vel; force; dthdt];
  end
  end
```

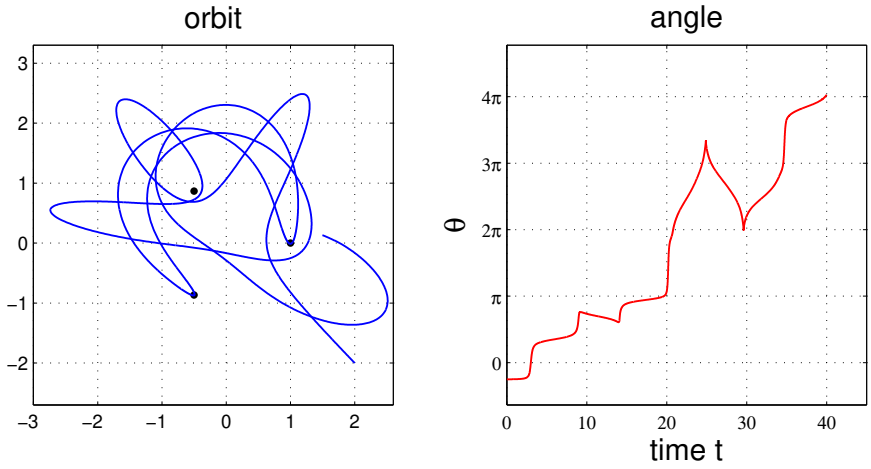
Now suppose we try various tolerances `tol`. Here are some results:

```
Tol = 1.00e-04    Position at t=40: (-2.23885690,-1.81134614)
Tol = 1.00e-06    Position at t=40: (-0.43796140,-0.77340695)
Tol = 1.00e-08    Position at t=40: ( 1.53505394, 0.07133092)
Tol = 1.00e-10    Position at t=40: ( 1.49699326, 0.13081708)
Tol = 1.00e-12    Position at t=40: ( 1.49671841, 0.13121419)
Tol = 3.00e-14    Position at t=40: ( 1.49671316, 0.13122175)
```

Until the tolerance is below $10^{-8}$, we don't get any accuracy at all! The reason is that this system is chaotic—small perturbations grow exponentially with time. The total growth over 40 time units is about a factor of $10^7$, and that is why we can manage to get some accuracy at the tighter tolerances.

To get chaos in a dynamical system, the state space must have dimension at least 3 (so that the orbits can get tangled up). Here, that's exactly the effective dimension. We have four variables—two positions and two velocities—but because of conservation of energy, the motion is confined to a 3-dimensional manifold.

Here is a plot computed with tolerance $3 \times 10^{-14}$. Note the sudden changes of the curve near the two very close approaches around $t = 25$ and $t = 30$. To someone sitting on one of those planets, it looks like a return of Halley's comet.




1'. Now we do it again compactly in Chebfun using complex arithmetic. Here's the code:

```
p = exp((0:2)*2i*pi/3);
x0 = 2-2i;
N = chebop(0,40);
N.lbc = [x0; 0];
N.op = @(x) diff(x,2) - (p(1)-x)/abs(p(1)-x)^3 - (p(2)-x)/abs(p(2)-x)^3 - (p(3)-x)/abs(p(3)-x)^3;
x = N\0;
subplot(2,2,1), plot(x,'b'), grid on, axis([-3 2.6 -2.7 3.3]), axis square
theta = unwrap(angle(x));
subplot(2,2,2), plot(theta,'r'), grid on, axis([0 45 -2 15]), axis square
```