Computer Lab Assignment 11

Radioactive Decay, Keplerian Orbits, and Runge-Kutta Method

(1) In lecture 16, we discussed a numerical algorithm to solve the following ordinary differential equation to calculate how fast the unstable isotope carbon-14 decays:

$$\frac{dc_{14}}{dt} = -k\, c_{14}(t)$$

Please use $c_{14}(t=0)=1$ as initial condition and a decay constant of $k=0.00012097\ year^{-1}$. Write your own 10 lines of Matlab code to solve this equation using the discretized form:

$$c(t+\Delta t) = c(t)\ \left[1 - k\Delta t\right]$$

Use a step size of 10 years. At the end, make a plot that compares your numerical solution to the analytical solution, $c(t) = c(0)\ e^{-kt}$. What is half-life of carbon-14?

(2) Now we want to use the same method to solve Newton's equation for a planet orbiting the sun. Instead of a single variable c, we now have four variables that change with time,

$$\vec{r} = (x,y) \qquad \vec{v} = (v_x, v_y)$$

$$\frac{dv_x}{dt} = -\frac{Gm_S}{r^3}\ x$$

$$\frac{dv_y}{dt} = -\frac{Gm_S}{r^3}\ y$$

$$\frac{dx}{dt} = v_x$$

$$\frac{dy}{dt} = v_y$$

For simplicity, set $m_S=10^6$, $G=3\times10^{-6}$, $\Delta t=10^{-3}$ and integrate for 10 time units. As initial condition, I recommend $x=1$, $y=0$, $v_x=0$, and $v_y=2$. Make a plot $y(t)$ versus $x(t)$. If this is anything like an ellipse then you have succeeded in this part of the lab. Congratulations!

(please turn over)

(3) Now we re-write the code a bit introducing a Matlab function that computes $\frac{d\vec{y}}{dt} = f(t,\vec{y})$, where $\vec{y}$ = [x y v_x v_y]. Your code should still use the Euler method and should do *exactly* the same calculation as before. Review lecture 19. Now open a separate file "kepler_ode.m" and write a function:

```
function ydot = kepler_ode(t,y)

%obtain 'r' and 'v' from 'y'
r = …
v = …

% use the following three variables from the main worksheet
% must declare them 'global' here and there
global ms G M

% Now put your formula for the force and acceleration here
F = …
a = …

% set the column 'ydot' that will be returned to the calling routine
ydot = …
```

This is all you need in the file "kepler_ode.m". In your main code you call this function with

```
dydt = kepler_ode(t,y);
```

further below you update the vector 'y'

```
y = y+ dydt .* dt;
```

Remove the old way of updating vectors 'r' and 'v'. Instead generate 'r' and 'v' each time from the current vector 'y'. Now make sure that your code still works and the plots are the same. Well done if it does!

(4) Instead of calling "kepler_ode.m" only once per time step dt (called h below), we want to call it 4 times as specified in the Runge-Kutta algorithm:

$$\vec{F}_1 = \vec{f}(t_n,\vec{y}_n)$$
$$\vec{F}_2 = \vec{f}(t_n + \tfrac{h}{2},\vec{y}_n + \tfrac{h}{2}\vec{F}_1)$$
$$\vec{F}_3 = \vec{f}(t_n + \tfrac{h}{2},\vec{y}_n + \tfrac{h}{2}\vec{F}_2)$$
$$\vec{F}_4 = \vec{f}(t_n + h,\vec{y}_n + h\vec{F}_3)$$

Introduce intermediate vectors $F_1 \ldots F_4$ and compute the much more accurate new 'y' vector

$$\vec{y}_{n+1} = \vec{y}_n + \frac{h}{6}\left[\vec{F}_1 + 2\vec{F}_2 + 2\vec{F}_3 + \vec{F}_4\right]$$

Now run the new code and see if you still get an ellipse. For a given dt, this integration method will be much more accurate. If you have time, compare the accuracy of the Euler and the Runge-Kutta method for different time steps. How would you define a measure of accuracy?