```
// simple harmonic oscillator using external Runge-Kutta function
// Corey Mutnik 3/15
// modified from: P. Gorham, originally 3/6/2003 for UH Physics 305, updated 3/10/2013


using namespace std;
#include <iostream>
#include <iomanip>
#include <fstream>
#define USE_MATH_DEFINES
#include <cmath>
#include <cstdlib>

#define Tmax  50                    // seconds


extern double FRK2xv(int, double (*)(double, double, double),
                                    double (*)(double, double, double),
                                    double ,double ,double ,double );
double  f_x(double, double, double), f_v(double, double, double);

main( int argc, char *argv[])
{
  double xt0,t0,vt0,vt,xt,t,dt,xtold,vtold, dE, E, dEperE;
  double k,m,w,xtrue,vtrue;

   ofstream outfile;

        outfile.open("oscRK1.dat");

// program wont run unless proper command life parameters are set
if(argc<2){
                cerr<< "usage: harmonic oscillator [time interval, dt]"<<endl;
                exit(0);
                }
// modify program to accept time interval as a command line parameter
        dt = atof(argv[1]);

        k = 1.0;                    // spring constant
        m = 1.0;                    // mass in kg
        w = sqrt(k/m);

        xt0 = 1.0;                  // initial position
        t0 = 0.0;                   // initial time
```

```
45          vt0 = 0.0;                  // initial velocity
    //      dt = 0.1;
47
            xtold = xt0;                // set & print the initial conditions
49          vtold = vt0;
            t = t0;
51
            dEperE = 0.0;
53          dE = 0.0;
            double Ei = (0.5*m*pow(vt0,2.0)) + (0.5*k*pow(xt0,2.0));
55          double T = 2.0*M_PI/w;

57          outfile << t << "\t" << xt0 << "\t" << vt0 << "\t" << xtrue << "\t" << vtrue <<
59
61          for(t=t0; t<Tmax; t+= dt){

63                  xt = xtold + FRK2xv(0,f_x,f_v,t,xtold,vtold,dt);
                    vt = vtold + FRK2xv(1,f_x,f_v,t,xtold,vtold,dt);
65
                    double dv = vt - vtold;
67                  double dx = xt - xtold;

69                  xtrue = xt0*cos(w*t);
                    vtrue = vt0  + -sin(w*t);
71
                    xtold = xt;
73                  vtold = vt;
                    E = (0.5*m*pow(vtrue,2.0)) + (0.5*k*pow(xtrue,2.0));
75  //              dE = (0.5*m*pow(dv,2.0)) + (0.5*k*pow(dx,2.0));
77
                    if(t>=(Tmax-T)){
79                    double Etot = (0.5*m*pow(vt,2.0)) + (0.5*k*pow(xt,2.0));
                      double Ef = abs(E-Etot)/E;
81                    dEperE += Ef;
                    }
83
                    outfile << t << "\t" << xt << "\t" << vt << "\t" << xtrue << "\t" << vt
85              }

87                  // modification to plot (fractional energy change in one period of the
    //              cout << (dt/T) << "\t" << ((dE/E)/T) << endl;
89                  cout << (dt/T) << "\t" << ((dEperE)/T) << endl;
```

```
}

/* the following is just the formal definition: f(x) = dx/dt == v */
double f_x(double t, double x, double v)
{
        return(v);
}

/* this is where the action is: should contain the force law: dv/dt = f(t,x,v) */
double f_v(double t, double x, double v)
{
        double k = 1.0, m=1.0;

        return( -k/m*x );
}


}
```