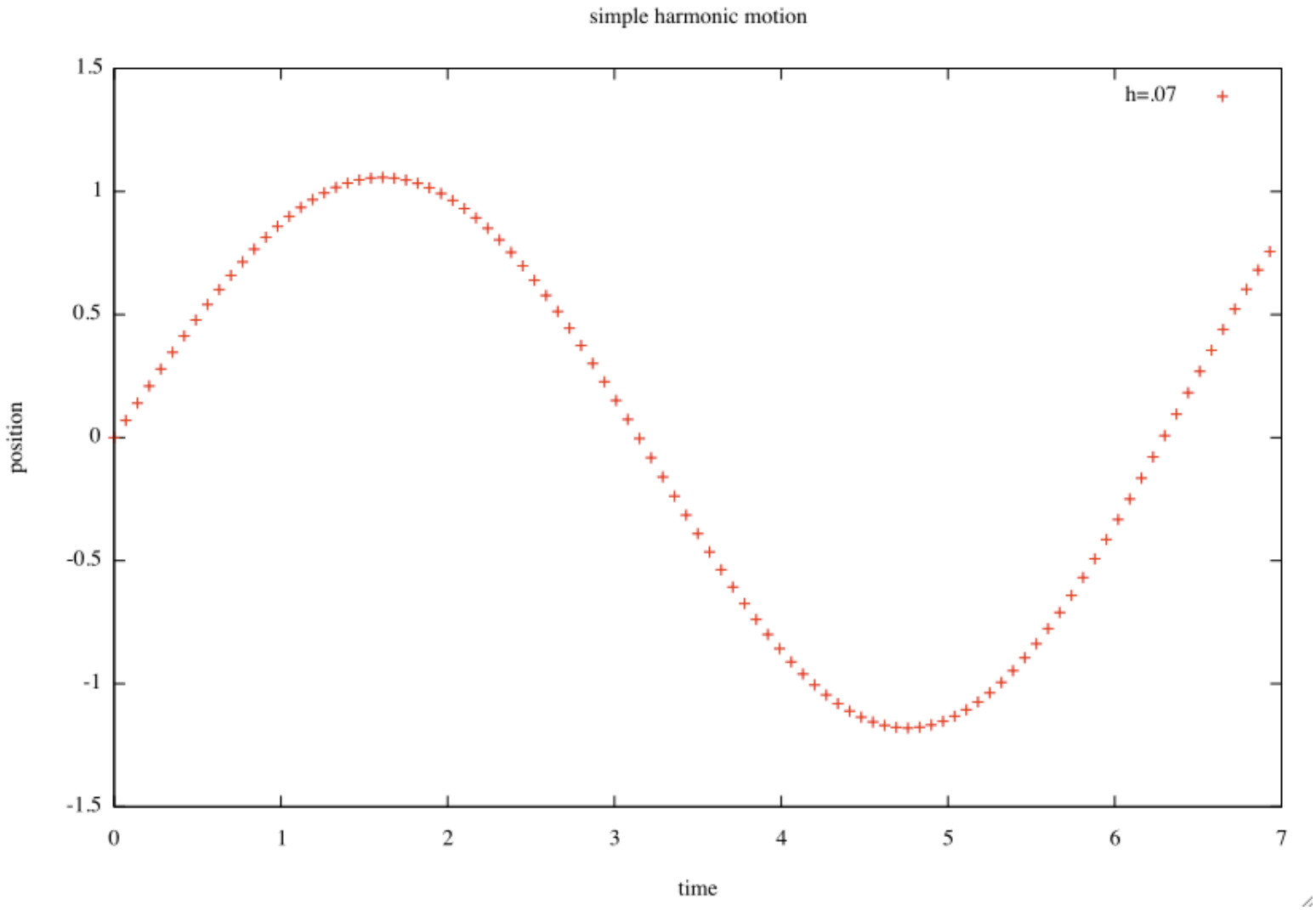
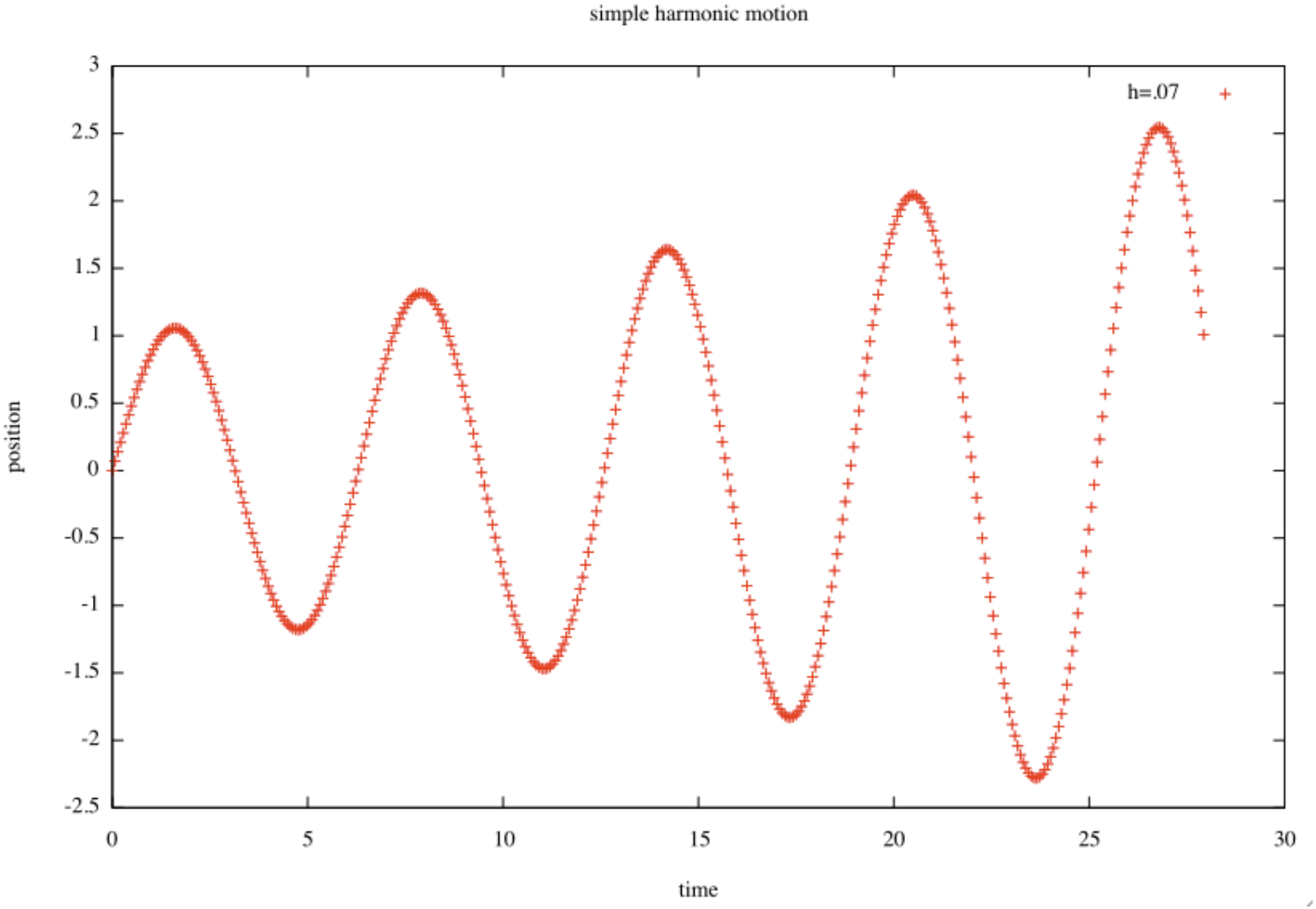
Harmonic Oscillator by Euler, Euler-Cromer and RK2 Algorithms

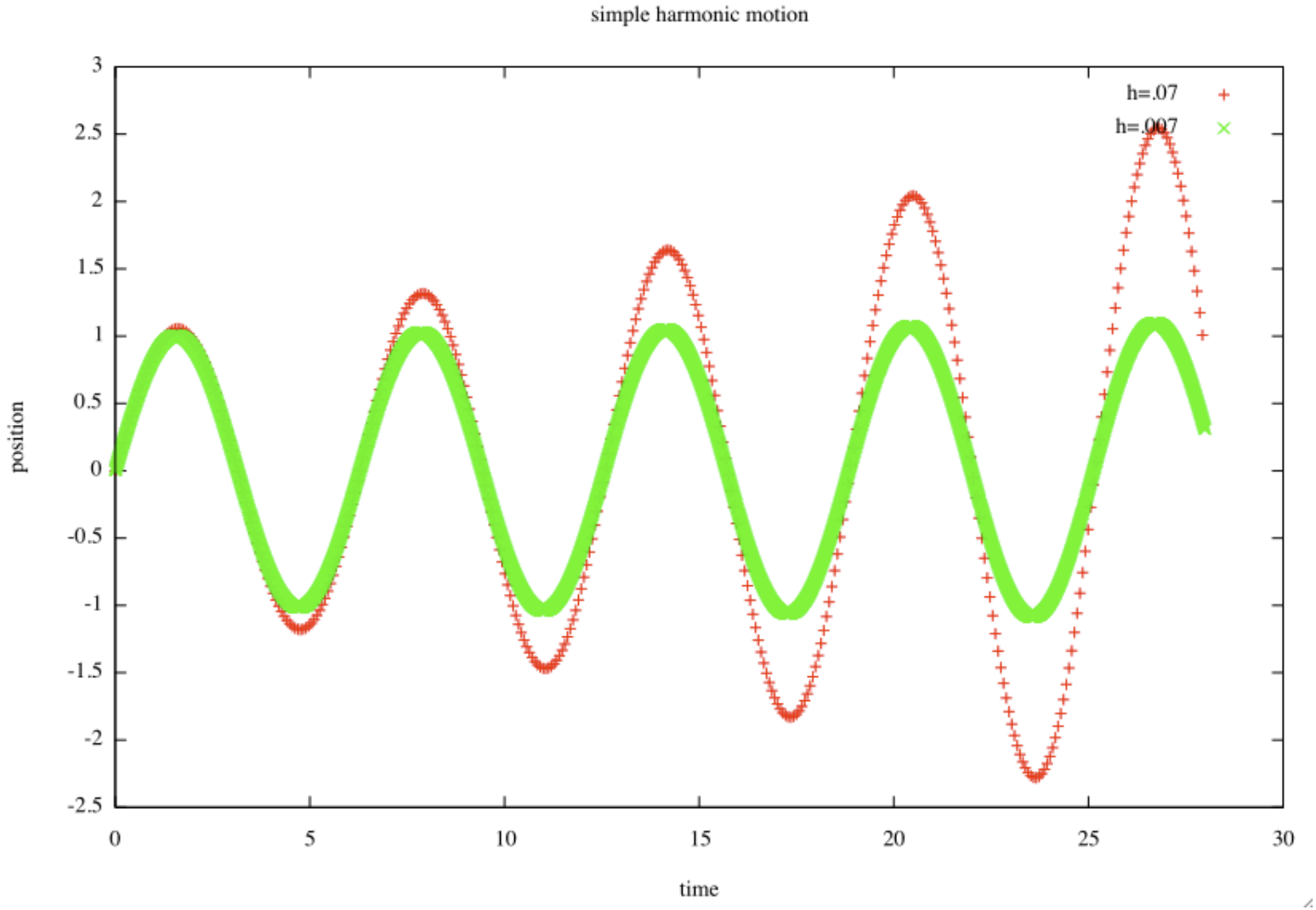
Harmonic Oscillator by Euler Algorithm

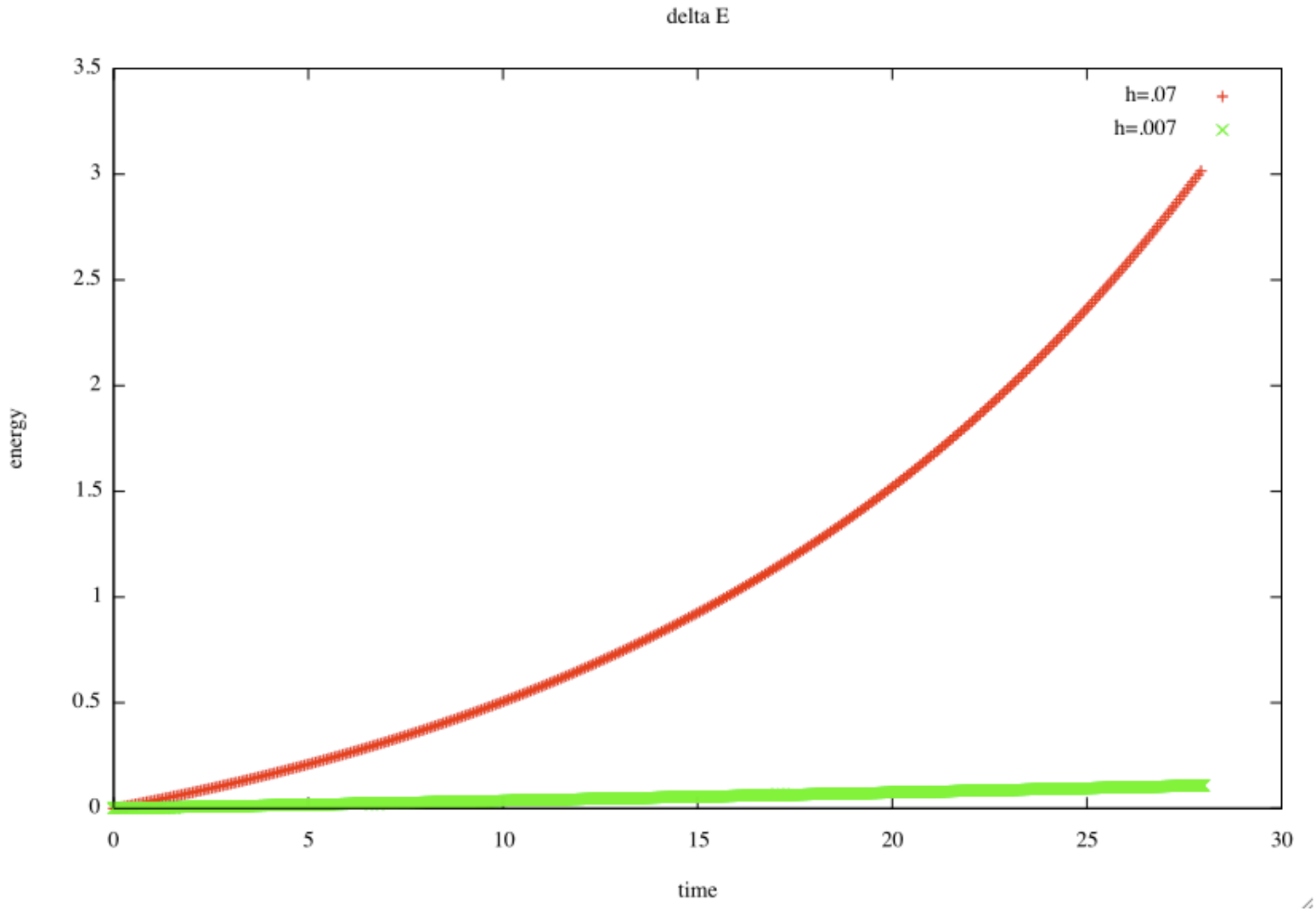
```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <assert.h>


void initialise (double *, double *, double *, int *);
void derivatives(double, double *, double *);
void euler(double *,double *,int,double,double,double *);

void output( FILE *, double, double, double *, double);

int main()
{
    FILE *output_file;

    // declarations of variables

    int number_of_steps;
    double initial_x, initial_v,final_t,E_initial;
    double h;

    double y[2], dydt[2], yout[2];
    double t;

    output_file = fopen("oscillation.dat", "w") ;

    //  read in input data from screen

    initialise (&initial_x, &initial_v, &final_t, &number_of_steps);
    assert(number_of_steps > 0);

    // initialize position and velocity

    y[0]=initial_x;
    y[1]=initial_v;
    E_initial= 0.5*y[0]*y[0] + 0.5*y[1]*y[1];

    t=0.;
    h=final_t/number_of_steps;

    while(t<=final_t){
        derivatives(t,y,dydt);
    euler(y,dydt,2,t,h,yout);
        output(output_file,h,t,y,E_initial);
        y[0]=yout[0];
        y[1]=yout[1];
        t+=h;
    }
    fclose(output_file);

        return 0;
}   // end main program
```

```c
void initialise (double *initial_x,  double *initial_v, double *final_t, int *number_of_steps)

{

    printf("Read in from screen the initial position x, initial velocity v, final time and number of steps\n")
    scanf("%lf %lf %lf %d",initial_x, initial_v, final_t, number_of_steps);
    return;

}  // end of function initialise

//  This function provides the first derivative of y[0] and y[1]; i.e.
// it provides the rhs of the coupled first order equations of motion
//for the harmonic oscillator with omega=1


void derivatives(double t, double *y, double *dydt)

{
    dydt[0] = y[1];
    dydt[1] = - y[0];

}   // end of function derivatives



//  This function computes the first derivative with centered algorithm

void euler(double *y,double *dydt,int n,double t,double h,double *yout)

{ int i;
    for(i=0;i<n;i++){
        yout[i]=y[i] + h*dydt[i];
    }
}   // end of euler integrator


//    function to write out the final results

void output(FILE *output_file, double h, double t, double *y, double E_initial)
{

    fprintf(output_file, "%12.5E \t %12.10E \t %12.10E \t %12.10E \t %12.10E \n",
                h, t,y[0], y[1],0.5*y[0]*y[0]+0.5*y[1]*y[1]-E_initial);
    return;

}  // end of function output
```

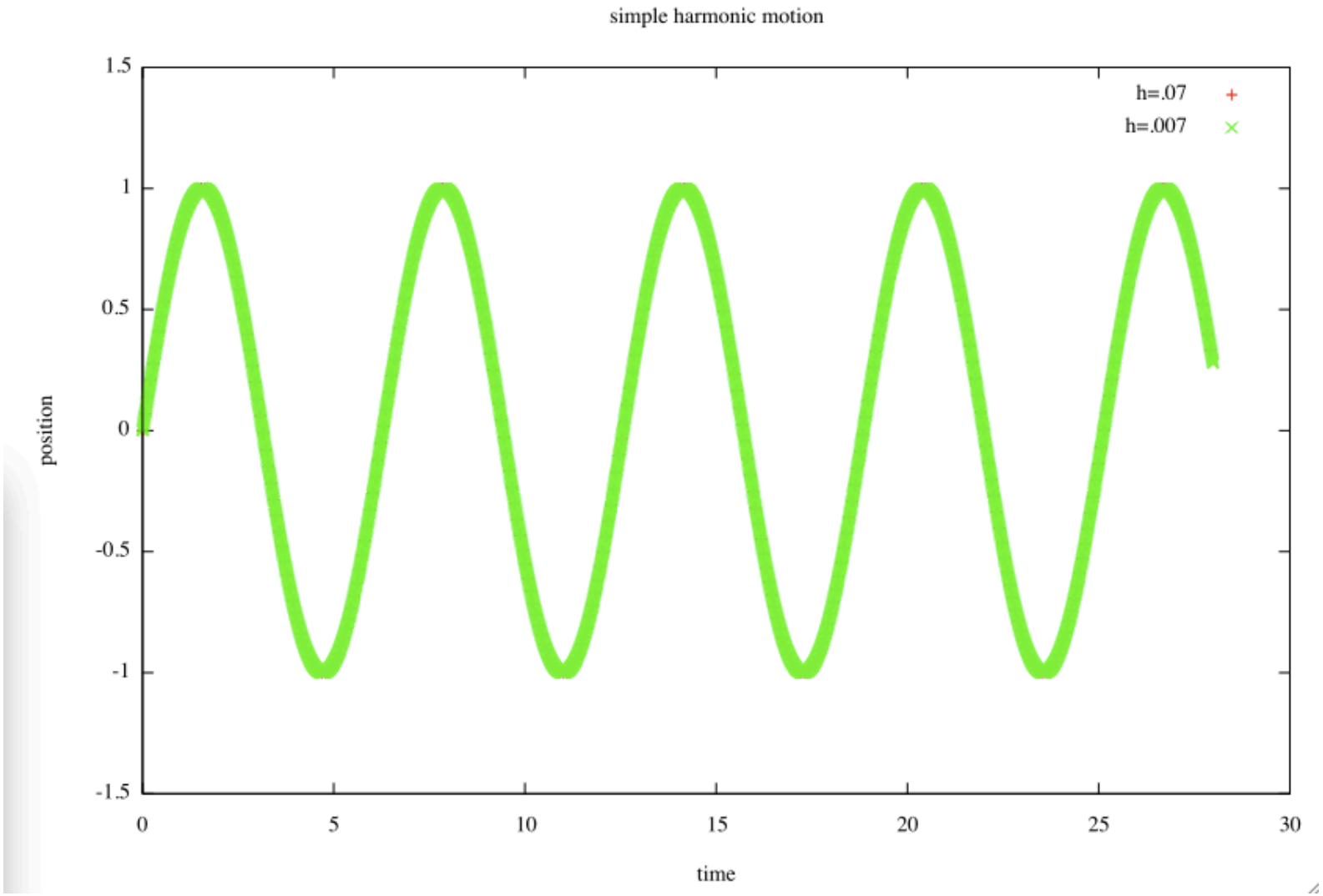

simple harmonic motion
h=.07
position
time
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
6
7


simple harmonic motion
h=.07
position
time
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30

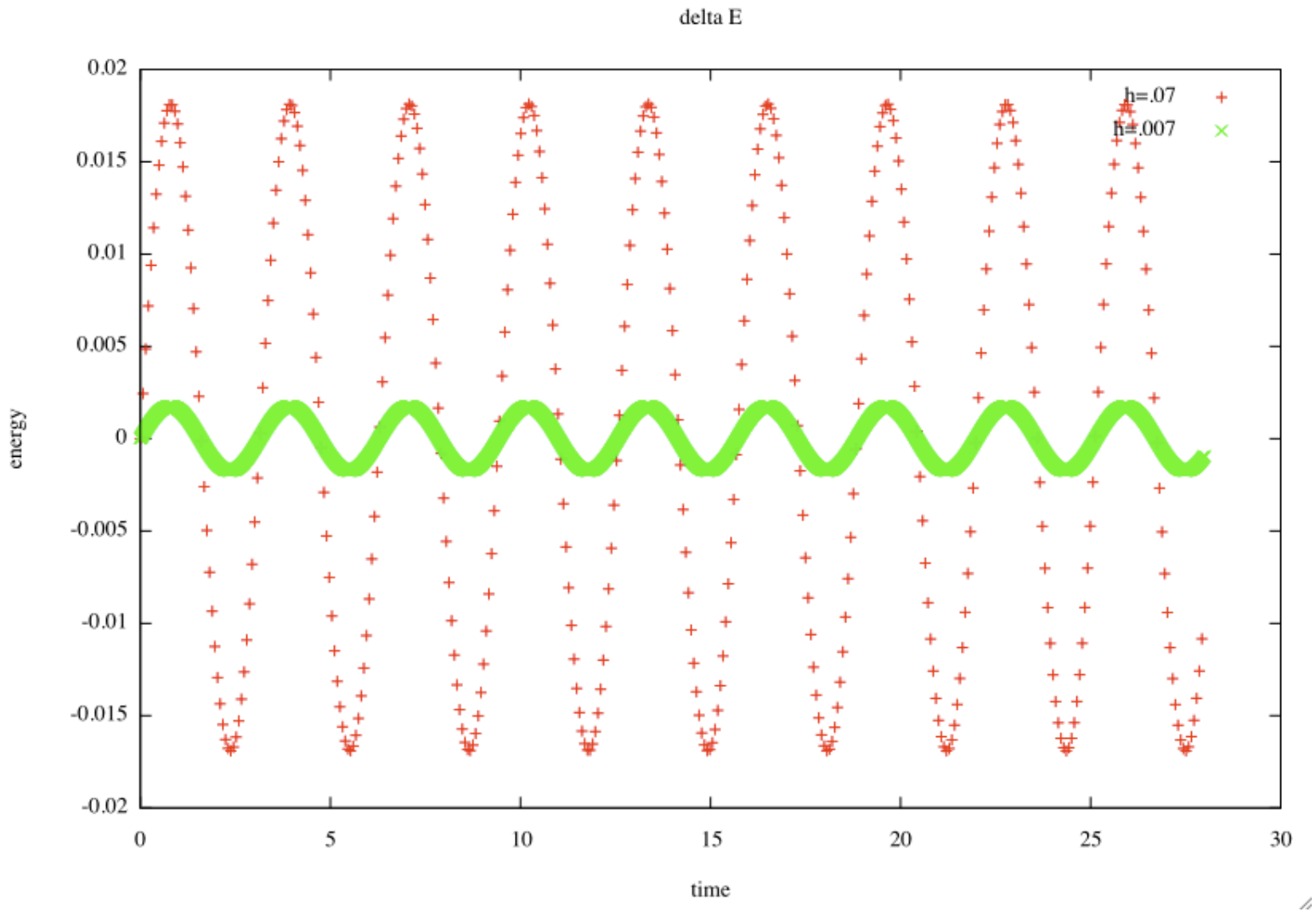

simple harmonic motion
h=.07
h=.007
position
time
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30


delta E
h=.07
h=.007
energy
time
0
0.5
1
1.5
2
2.5
3
3.5
0
5
10
15
20
25
30

Harmonic Oscillator by Euler-Cromer Algorithm

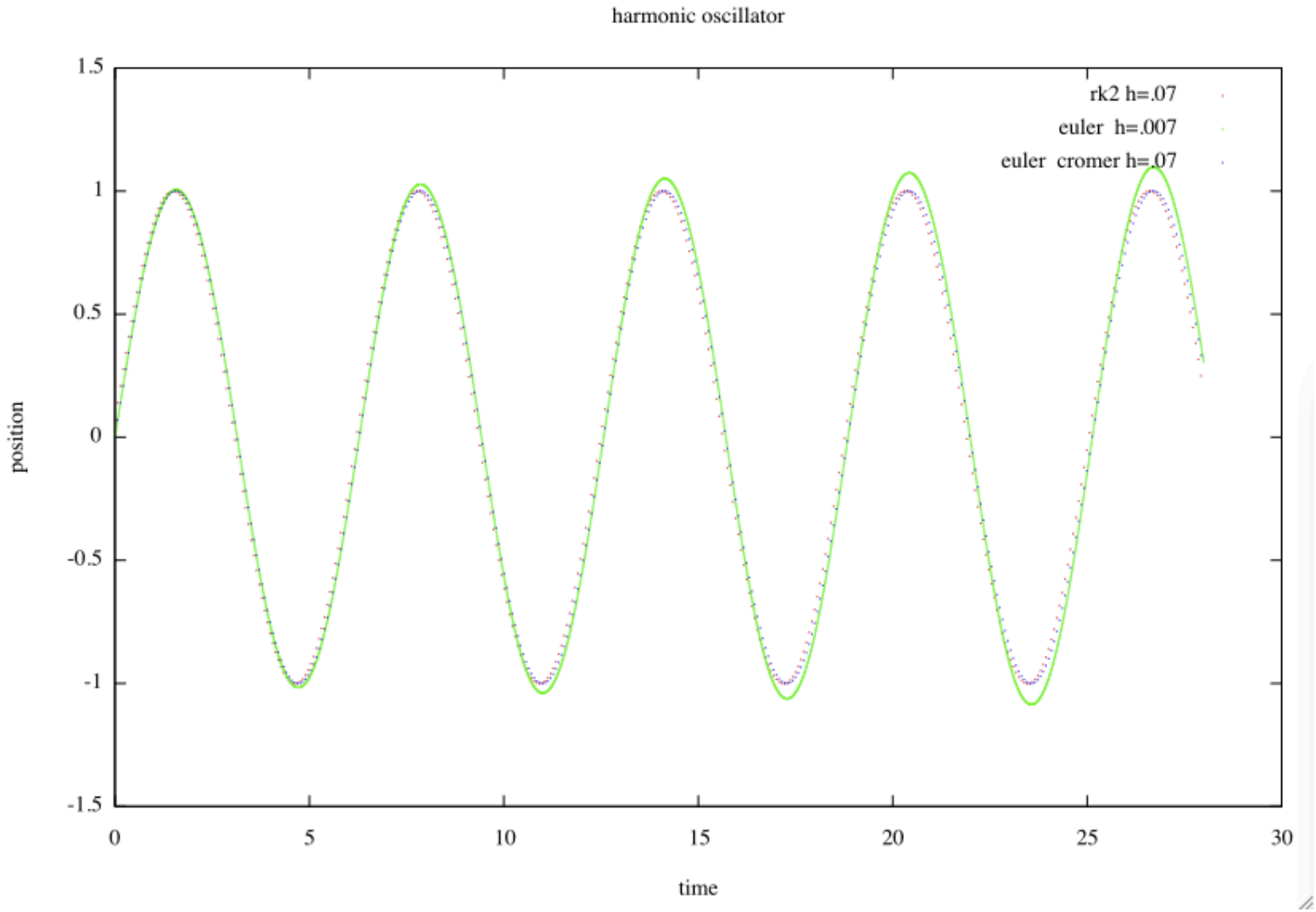


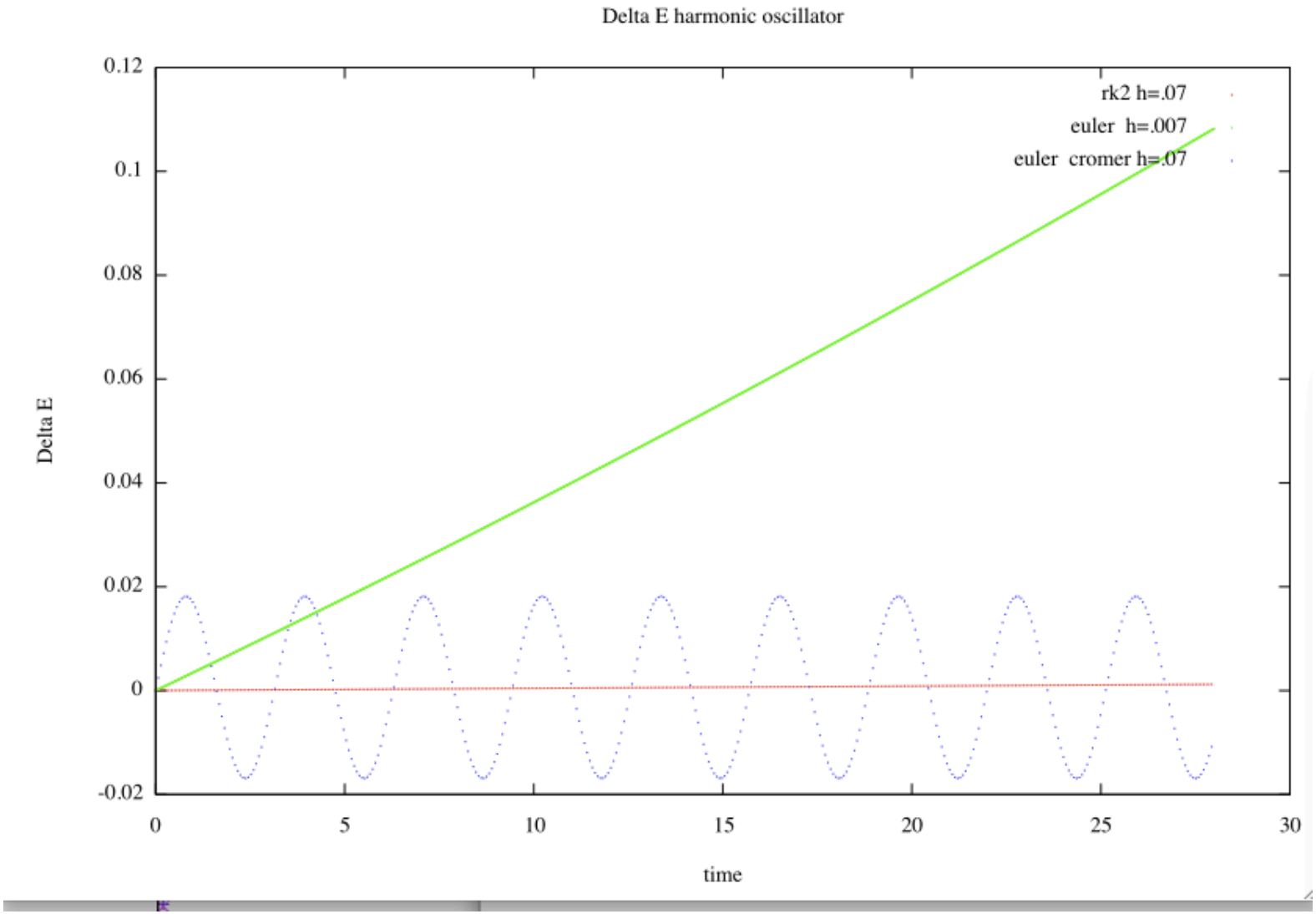
Euler Cromer algorithm result

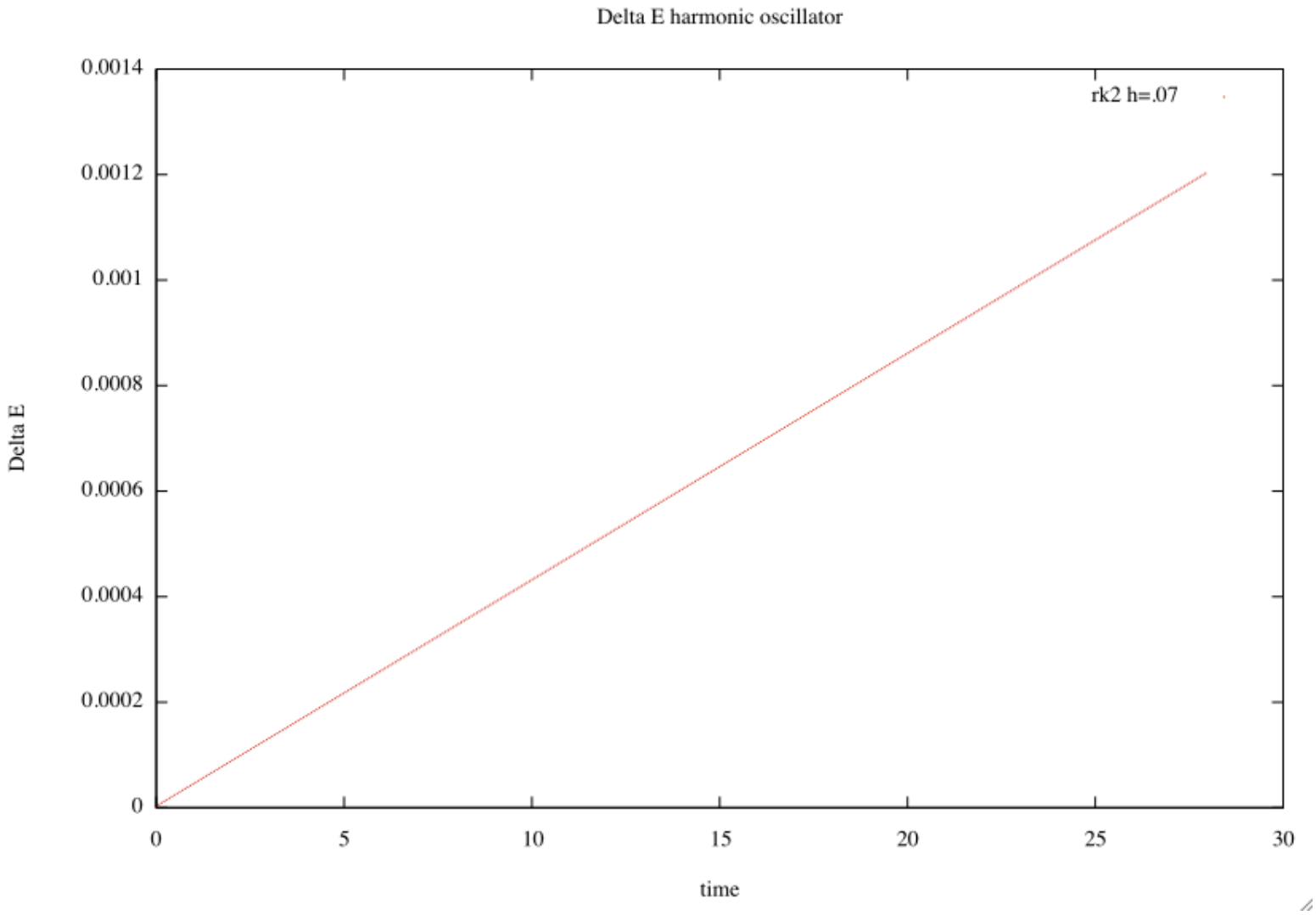

delta E
h=.07
h=.007
energy
time
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
0
5
10
15
20
25
30

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <assert.h>

void initialise (double *, double *, double *, int *);
void derivatives(double, double *, double *);
void runge_kutta_2(double *, double *, int, double , double ,
                   double *, void (*)(double, double *, double *));
void output( FILE *, double, double, double *, double);

int main()
{
    FILE *output_file;

    // declarations of variables

    int number_of_steps;
    double initial_x, initial_v,final_t,E_initial;
    double h;

    double y[2], dydt[2], yout[2];
    double t;

    output_file = fopen("oscillation.dat", "w") ;

    //  read in input data from screen

    initialise (&initial_x, &initial_v, &final_t, &number_of_steps);
    assert(number_of_steps > 0);

    // initialize position and velocity

    y[0]=initial_x;
    y[1]=initial_v;
    E_initial= 0.5*y[0]*y[0] + 0.5*y[1]*y[1];

    t=0.;
    h=final_t/number_of_steps;
    output(output_file,h,t,y,E_initial);

    while(t<=final_t){
        derivatives(t,y,dydt);
        runge_kutta_2(y,dydt,2,t,h,yout,derivatives);
        y[0]=yout[0];
        y[1]=yout[1];
        output(output_file,h,t,y,E_initial);
        t+=h;
    }
    fclose(output_file);

    return 0;
}   // end main program
```

```
//  This function provides the first derivative of y[0] and y[1]; i.e.
// it provides the rhs of the coupled first order equations of motion
//for the harmonic oscillator with omega=1


void derivatives(double t, double *y, double *dydt)

{
    dydt[0] = y[1];
    dydt[1] = - y[0];

}   // end of function derivatives



void runge_kutta_2(double *y, double *dydt, int n, double x, double h,
                   double *yout, void (*derivatives)(double, double *, double *))
{
    int i;
    double      xh;
    double  *dyt, *yt;
    //   allocate space for local vectors
    dyt =  (double *) malloc(n*sizeof(double));
    yt =  (double *) malloc(n*sizeof(double));
    xh = x+h/2.0;
    for (i = 0; i < n; i++) {
        yt[i] = y[i]+h/2.0*dydt[i];
    }

    // computation of y_t+1/2h = yt + h/2 dy/dt using k1 = h dy/dt

    (*derivatives)(xh,yt,dyt);

    // find k2 which is h* dyt

    for (i = 0; i < n; i++) {
        yout[i] = y[i]+h*dyt[i];
    }
    //increment yout using yout = y + h k2

    free(dyt);
    free(yt);
}       //  end of function Runge-kutta 2
```


harmonic oscillator
rk2 h=.07
euler h=.007
euler cromer h=.07
position
time
1.5
1
0.5
0
-0.5
-1
-1.5
0
5
10
15
20
25
30


Delta E harmonic oscillator
rk2 h=.07
euler h=.007
euler cromer h=.07
Delta E
time
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
0
5
10
15
20
25
30


Delta E harmonic oscillator
rk2 h=.07
Delta E
time
0.0014
0.0012
0.001
0.0008
0.0006
0.0004
0.0002
0
0
5
10
15
20
25
30