```c
#include "basic.h"
#include "linalg.h"
#include "ode.h"

/**
 * Define things for mass-spring system
 */

real mass_spring_c = 1.0;
real mass_spring_m = 0.5;

/**
 * Setup some initial values for the mass-spring system
 */
void init_mass_spring(pvector y0)
{
    assert(y0->dim == 2);

    y0->x[0] = 1.0; // initial positon at right border
    y0->x[1] = 0.0; // initial velocity pointing to the left
}

/**
 * right-hand-side for the mass-spring system
 */
void mass_spring_func(real t, pvector yt, pvector yt1, void *data)
{
    real c_m = *((real *)data); // the constant c/m appearing in the equations

    yt1->x[0] = yt->x[1];
    yt1->x[1] = -c_m * yt->x[0];
}

int main(int argc, char const *argv[])
{

    uint dim;
    real a, b;
    real delta;
    real t;
    uint steps;
    real mass_spring_c_m;
    real gamma;
    pvector yt, yt1;
    FILE *fp;
    char filename[100];

    printf(
        "\n"

        "################################################################################\n
        "
        "#                        Exercise: mass-spring system
        #\n"

        "################################################################################\n
        "
        "\n");

    /*****************************************************************************
     *  Mass-spring system
     *****************************************************************************/

    dim = 2;
    mass_spring_c_m = mass_spring_c / mass_spring_m;
    a = 0.0;
    b = 20.0;

    // Initialize problem
```

```
65      yt = new_vector(dim);
66      yt1 = new_vector(dim);
67
68      printf(
69          "Simulating spring-mass system via explicit Euler method:\n");
70
71      delta = 4.0e-1;
72      while (delta >= 1.0e-4)
73      {
74
75          init_mass_spring(yt);
76
77          // start simulation from 't = a' to 't = b' with stepwidth 'delta'
78          t = a;
79          sprintf(filename, "data/ode_euler_delta%.4f.dat", delta);
80          fp = fopen(filename, "w");
81          assert(fp != NULL);
82          fprintf(fp, "%.4f\t%.5e\t%.5e\n", t, yt->x[0], yt->x[1]);
83          printf("  Starting simulation with delta = %.2e\n", delta);
84          while (t < b)
85          {
86              euler_step(t, yt, (ode_func)mass_spring_func, delta, yt1, &mass_spring_c_m);
87              t += delta;
88              fprintf(fp, "%.4f\t%.5e\t%.5e\n", t, yt->x[0], yt->x[1]);
89          }
90          fclose(fp);
91          printf("  final position: %.3f, final velocity: %.3f\n", yt->x[0], yt->x[1]);
92          printf("\n");
93
94          delta *= 0.25;
95      }
96
97      printf(
98          "Simulating spring-mass system via Runge method:\n");
99
100     delta = 4.0e-1;
101     while (delta >= 1.0e-4)
102     {
103
104         init_mass_spring(yt);
105
106         // start simulation from 't = a' to 't = b' with stepwidth 'delta'
107         t = a;
108         sprintf(filename, "data/ode_runge_delta%.4f.dat", delta);
109         fp = fopen(filename, "w");
110         assert(fp != NULL);
111         fprintf(fp, "%.4f\t%.5e\t%.5e\n", t, yt->x[0], yt->x[1]);
112         printf("  Starting simulation with delta = %.2e\n", delta);
113         while (t < b)
114         {
115             runge_step(t, yt, (ode_func)mass_spring_func, delta, yt1, &mass_spring_c_m);
116             t += delta;
117             fprintf(fp, "%.4f\t%.5e\t%.5e\n", t, yt->x[0], yt->x[1]);
118         }
119         fclose(fp);
120         printf("  final position: %.3f, final velocity: %.3f\n", yt->x[0], yt->x[1]);
121         printf("\n");
122
123         delta *= 0.25;
124     }
125
126     del_vector(yt);
127     del_vector(yt1);
128
129     return 0;
130 }
131
```