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

/**
 * Define things for gravitational N-body problem
 */

/**
 * Setup some initial values for the gravitation problem
 */
void init_gravitation(pvector y0)
{
    uint dim = y0->dim;
    // Vector has length 7*n, 3*n for positions, 3*n for velocities, n for masses
    uint n = dim / 7;
    uint i;
    real scale = 1.0 / RAND_MAX;

    srand(42);

    for (i = 0; i < n; i++)
    {
        // set some random positions
        y0->x[i + 0 * n] = (real)rand() * scale;
        y0->x[i + 1 * n] = (real)rand() * scale;
        y0->x[i + 2 * n] = (real)rand() * scale;
        // set velocities to zero
        y0->x[i + 3 * n] = 0.0;
        y0->x[i + 4 * n] = 0.0;
        y0->x[i + 5 * n] = 0.0;
        // set some random masses
        y0->x[i + 6 * n] = 10.0 + 10.0 * ((real)rand() * scale);
    }
}

/**
 * right-hand-side for the gravitation problemm
 */
void gravitation_func(real t, pvector yt, pvector yt1, void *data)
{
    uint dim = yt->dim;
    // Vector has length 7*n, 3*n for positions, 3*n for velocities, n for masses
    uint n = dim / 7;
    real gamma = *((real *)data);
    uint i, j;
    real sum[3];
    real dist[3];
    real norm;
    real m;

    assert(yt1->dim == dim);

    for (i = 0; i < n; i++)
    {
        // x'(t) = v(t)
        yt1->x[i + 0 * n] = yt->x[i + 3 * n];
        yt1->x[i + 1 * n] = yt->x[i + 4 * n];
        yt1->x[i + 2 * n] = yt->x[i + 5 * n];

        // compute forces
        sum[0] = 0.0;
        sum[1] = 0.0;
        sum[2] = 0.0;

        for (j = 0; j < n; j++)
        {
            if (i != j)
            {
                // Get the mass m_j
                m = yt->x[j + 6 * n];

                // Distance vector
```

```
74                  dist[0] = yt->x[j + 0 * n] - yt->x[i + 0 * n];
75                  dist[1] = yt->x[j + 1 * n] - yt->x[i + 1 * n];
76                  dist[2] = yt->x[j + 2 * n] - yt->x[i + 2 * n];
77
78                  norm = dist[0] * dist[0] + dist[1] * dist[1] + dist[2] * dist[2];
79                  norm = 1.0 / sqrt(norm);
80                  norm = m * norm * norm * norm;
81
82                  sum[0] += dist[0] * norm;
83                  sum[1] += dist[1] * norm;
84                  sum[2] += dist[2] * norm;
85              }
86          }
87
88          // set velocities
89          yt1->x[i + 3 * n] = gamma * sum[0];
90          yt1->x[i + 4 * n] = gamma * sum[1];
91          yt1->x[i + 5 * n] = gamma * sum[2];
92
93          // set "new mass" to zero.
94          yt1->x[i + 6 * n] = 0.0;
95      }
96  }
97
98  void print_state_gravitation(pvector y)
99  {
100     uint dim = y->dim;
101     // Vector has length 7*n, 3*n for positions, 3*n for velocities, n for masses
102     uint n = dim / 7;
103     uint i;
104
105     printf("%d bodies\n", n);
106     printf("                  mass      |
        position                  |                  velocity\n");
107     for (i = 0; i < n && i < 10; i++)
108     {
109         printf("mass %2d:\t%.3e\t(%+.5e, %+.5e, %+.5e)\t(%+.5e, %+.5e, %+.5e)\n", i,
            y->x[i + 6 * n],
110                 y->x[i + 0 * n], y->x[i + 1 * n], y->x[i + 2 * n],
111                 y->x[i + 3 * n], y->x[i + 4 * n], y->x[i + 5 * n]);
112     }
113     printf("\n");
114 }
115
116 int main(int argc, char const *argv[])
117 {
118
119     uint dim;
120     real a, b;
121     real delta;
122     real t;
123     uint steps;
124     real gamma;
125     pvector yt, yt1;
126     FILE *fp;
127     char filename[100];
128
129     printf(
130         "\n"
131
            "##############################################################################
            #\n"
132         "#                                Exercise
            02                                #\n"
133
            "##############################################################################
            #\n"
134         "\n");
135
136     /*****************************************************************************
137      *  Many body problem
138      *****************************************************************************/
139
```

```
140         printf(
141             "Simulating gravitational N-Body Problem via explicit Euler method:\n");
142 
143         dim = 20;
144         gamma = 1.0e-5;
145         a = 0.0;
146         b = 2.0;
147 
148         // Initialize problem
149         yt = new_vector(dim * 7);
150         yt1 = new_vector(dim * 7);
151 
152         delta = 1.0e-1;
153         while (delta >= 1.0e-5)
154         {
155 
156             init_gravitation(yt);
157             printf("Initial configuration (delta = %.2e):\n", delta);
158             print_state_gravitation(yt);
159 
160             // start simulation from 't = a' to 't = b' with stepwidth 'delta'
161             t = a;
162             while (t < b)
163             {
164                 euler_step(t, yt, (ode_func)gravitation_func, delta, yt1, &gamma);
165                 t += delta;
166             }
167             printf("Final configuration (delta = %.2e):\n", delta);
168             print_state_gravitation(yt);
169             printf("\n\n");
170 
171             delta *= 0.1;
172         }
173 
174         del_vector(yt);
175         del_vector(yt1);
176 
177         return 0;
178     }
179 
```