```
1
     #include "ode.h"
2
     void euler step (real t, pvector yt, ode func f, real delta, pvector yt1, void *data)
3
4
         uint dim = yt->dim;
5
6
         uint i;
8
         assert(dim == yt1->dim);
9
10
         // evaluate function at point 't', current vector 'yt' into 'yt1'
11
         // compute the right hand side function and store the value in the vector yt1
         ---> Phi
12
         f(t, yt, yt1, data);
13
14
         // update new vector 'yt1'
15
         for (i = 0; i < dim; i++)</pre>
16
17
             yt->x[i] = yt->x[i] + delta * yt1->x[i];
18
         }
19
     }
20
21
     void runge step(real t, pvector yt, ode func f, real delta, pvector yt1, void *data)
22
23
         uint dim = yt->dim;
24
25
         pvector ym;
26
         uint i;
27
28
         assert(dim == yt1->dim);
29
30
         ym = new vector(dim);
31
32
         // evaluate function at point 't', current vector 'yt' into 'yt1'
33
         f(t, yt, yt1, data);
34
35
         // update new vector 'ym'
36
         for (i = 0; i < dim; i++)
37
         {
             ym->x[i] = yt->x[i] + 0.5 * delta * yt1->x[i];
38
39
         1
40
         // evaluate function at point 't+delta/2', current vector 'ym' into 'yt1'
41
42
         f(t, ym, yt1, data);
43
44
         // update new vector 'yt'
45
         for (i = 0; i < dim; i++)</pre>
46
47
             yt->x[i] = yt->x[i] + delta * yt1->x[i];
48
49
50
         del_vector(ym);
51
     }
52
53
     void leapfrog_step(real t, pvector yt, ode_func f, real delta, pvector yt1, void
     *data)
54
55
         uint dim = yt->dim;
56
         real c_mass_spring = *((real *)data);
57
58
         (void) f;
59
         (void) yt1;
60
61
         assert(dim == yt1->dim);
62
         assert(dim == 2);
63
64
         yt-x[0] = yt-x[0] + 2.0 * delta * yt-x[1];
6.5
         yt-x[1] = yt-x[1] - 2.0 * delta * c mass spring * yt-x[0];
66
     }
67
68
     void crank_nicolson_step(real t, pvector yt, ode_func f, real delta, pvector yt1,
     void *data)
69
     {
         uint dim = yt->dim;
```

```
71
         real c mass spring = *((real *)data);
72
73
         uint i;
74
         real onepdelta, onemdelta;
75
76
         (void) f;
77
78
         assert(dim == yt1->dim);
79
         assert(dim == 2);
80
81
         onepdelta = 1.0 + 0.25 * delta * delta * c_mass_spring;
         onemdelta = 1.0 - 0.25 * delta * delta * c_mass_spring;
82
83
84
         yt1-x[0] = onemdelta * yt-x[0] + delta * yt-x[1];
85
         yt1-x[1] = onemdelta * yt-x[1] - delta * c_mass_spring * yt-x[0];
86
        yt-x[0] = yt1-x[0] / onepdelta;

yt-x[1] = yt1-x[1] / onepdelta;
87
88
89
     }
90
```