

C++ code 11.12.6: Function generating table for convergence studies.

```
2  template <typename GVFUNCTOR>
3  void testCVGLEapfrog(GVFUNCTOR &&gradV, const Eigen::VectorXd &y0,
4                      const Eigen::VectorXd &v0, double T) {
5      const unsigned int no_exp = 8; // Number of times the stepsize is halved
6      unsigned int M = 16;           // Smallest number of timesteps
7      // Vectors for storing final approximate states
8      std::vector<Eigen::VectorXd> yT;
9      std::vector<Eigen::VectorXd> vT;
10     // Run leapfrog with various timesteps
11     for (unsigned int j = 0; j < no_exp; ++j, M *= 2) {
12         auto [y, v] = leapFrog(gradV, y0, v0, T, M);
13         yT.push_back(y);
14         vT.push_back(v);
15     }
16     // Finally run leapfrog with 1/10 the timestep to obtain a reference
17     // solution
18     auto [y_ref, v_ref] = leapFrog(gradV, y0, v0, T, 10 * M);
19     // Tabulate the Euclidean vector norms of the final errors
20     M = 16;
21     double y_errp;
22     double v_errp;
23     for (unsigned int l = 0; l < yT.size(); ++l, M *= 2) {
24         const double y_errn = (y_ref - yT[l]).norm();
25         const double v_errn = (v_ref - vT[l]).norm();
26         std::cout << "M= " << M << ": |yM-y(T)| = " << y_errn
27                     << ", |vM-v(T)| = " << v_errn;
28         if (l > 0) {
29             std::cout << ", ratio y = " << y_errp / y_errn
30                     << ", ratio v = " << v_errp / v_errn;
31         }
32         std::cout << std::endl;
33         y_errp = y_errn;
34         v_errp = v_errn;
35     }
36 }
```