

C++ code 12.8.11: Sub-problem (12-8.i): SDIRK solver for gradient flow problem

```
2  std::vector<Eigen::VectorXd> solveGradientFlow(const Eigen::VectorXd &d,
3                                                  double lambda,
4                                                  const Eigen::VectorXd &y,
5                                                  double T, unsigned int M) {
6
7      std::vector<Eigen::VectorXd> Y(M + 1);
8      // TODO: (12-8.i) Solve the Gradient flow ODE with the vector potential given
9      // in 12.8.8
10     // START
11     // Define the right hand side of the ODE  $y' = f(y)$ , and the
12     // Jacobian of  $f$ .
13     auto f = [d, lambda](const Eigen::VectorXd &yt) {
14         Eigen::VectorXd val =
15             -2. * std::cos(yt.squaredNorm()) * yt - 2. * lambda * d.dot(yt) * d;
16         return val;
17     };
18     auto J = [d, lambda](const Eigen::VectorXd &yt) {
19         const unsigned int dim = yt.size();
20         Eigen::MatrixXd term1 =
21             4. * std::sin(yt.squaredNorm()) * yt * yt.transpose();
22         Eigen::MatrixXd term2 =
23             -2. * std::cos(yt.squaredNorm()) * Eigen::MatrixXd::Identity(dim, dim);
24         Eigen::MatrixXd term3 = -2. * lambda * d * d.transpose();
25         Eigen::MatrixXd Jval = term1 + term2 + term3;
26         return Jval;
27     };
28
29     // Split the interval  $[0, T]$  into  $M$  intervals of size  $h$ .
30     const double h = T / M;
31     Eigen::VectorXd yt = y;
32     Y.at(0) = y;
33     // Evolve up to time  $T$ :
34     for (unsigned int i = 1; i <= M; i++) {
35         yt = discEvoSDIRK(f, J, yt, h);
36         Y.at(i) = yt;
37     }
38     // END
39     return Y;
40 }
```

Get it on  GitLab ([gradientflow.hpp](#)).