C++ code 7.8.11: Sub-problem (7-8.i): SDIRK solver for gradient flow problem ⇢ GITHUB

```cpp
std::vector<Eigen::VectorXd> SolveGradientFlow(const Eigen::VectorXd &d,
                                               double lambda,
                                               const Eigen::VectorXd &y0,
                                               double T, unsigned int M) {
  // initialize solution vector
  std::vector<Eigen::VectorXd> sol(M + 1, Eigen::VectorXd::Zero(y0.size()));

  // Define the right hand side of the ODE y' = f(y), and the Jacobian of f.
  auto f = [d, lambda](const Eigen::VectorXd &y) {
    Eigen::VectorXd val =
        -2. * std::cos(y.squaredNorm()) * y - 2. * lambda * d.dot(y) * d;
    return val;
  };
  auto df = [d, lambda](const Eigen::VectorXd &y) {
    int dim = y.size();
    Eigen::MatrixXd term1 = 4. * std::sin(y.squaredNorm()) * y * y.transpose();
    Eigen::MatrixXd term2 =
        -2. * std::cos(y.squaredNorm()) * Eigen::MatrixXd::Identity(dim, dim);
    Eigen::MatrixXd term3 = -2. * lambda * d * d.transpose();
    Eigen::MatrixXd dfval = term1 + term2 + term3;
    return dfval;
  };

  // Split the interval [0,T] into M intervals of size h.
  double h = T / M;
  Eigen::VectorXd y = y0;
  sol[0] = y;
  // Evolve up to time T:
  for (int i = 1; i <= M; i++) {
    y = DiscEvolSDIRK(f, df, y, h);
    sol[i] = y;
  }
  return sol;
}
```