

C++ code 11.9.6: Sub-problem (11-9.b): The function `computeIsolinePoints()`

```
2 template <typename GradientFunctor>
3 Eigen::Matrix<double, 2, Eigen::Dynamic> computeIsolinePoints(
4     GradientFunctor &&gradF, Eigen::Vector2d y0, double T) {
5     Eigen::Matrix<double, 2, Eigen::Dynamic> States =
6         Eigen::Matrix2Xd::Zero(2, 1);
7     // TODO: (11-9.b) Solve the IVP as given in the task description.
8     // START
9     // Right-hand-side vector field of isoline ODE
10    auto rhs = [gradF](const Eigen::Vector2d &x) {
11        Eigen::Vector2d gradFx = gradF(x);
12        Eigen::Vector2d out(-gradFx(1), gradFx(0)); // rotate
13        out /= gradFx.norm(); // scale
14        return out;
15    };
16    // Adaptive explicit Runge-Kutta method
17    Ode45<Eigen::Vector2d> integrator(rhs);
18    // Set tolerances for timestep control
19    integrator.options().rtol = 0.00001;
20    integrator.options().atol = 1e-9;
21    // Perform explicit timestepping
22    std::vector<std::pair<Eigen::Vector2d, double>> sol = integrator.solve(y0, T);
23    // Convert output into requested format: points on isoline arranged into the
24    // columns of a matrix.
25    const unsigned int N = sol.size() - 1;
26    States = Eigen::MatrixXd::Zero(2, N + 1);
27    for (unsigned int i = 0; i <= N; i++) {
28        States.col(i) = sol[i].first;
29    }
30    // END
31    return States;
32 }
```

Get it on  GitLab (contour.hpp).