

## C++ code 6.7.4: Sub-problem (6-7.b): Computation of cubic Hermite interpolant

```
2 template <typename Function>
3 Eigen::VectorXd fppchip(Function &&f, const Eigen::VectorXd &t,
4                          const Eigen::VectorXd &x) {
5     Eigen::VectorXd ret{x.size()};
6     // TODO: (6-7.b) Compute ret.
7     // START
8     Eigen::VectorXd y = t.unaryExpr(f);
9     unsigned int i_star = 0;
10    double tmp, h_tmp, t1, t2, y1, y2, a1, a2, a3;
11    for (unsigned int j = 0; j < x.size(); j) {
12        // Stores the current interval index and some temporary variable
13        // Find the interval for x_j while assuming that x values are sorted
14        // and within  $[t_{\min}, t_{\max}]$ 
15        if (t(i_star) <= x(j) && x(j) <= t(i_star + 1)) {
16            t1 = t(i_star);
17            t2 = t(i_star + 1);
18            h_tmp = t2 - t1; // h_tmp = h for equidistant grid
19            // values of interpolant
20            y1 = y(i_star);
21            y2 = y(i_star + 1);
22
23            a1 = y2 - y1;
24            a2 = a1;
25            a3 = -a1 - a2;
26
27            // Compute s(x(j))
28            tmp = (x(j) - t1) / h_tmp;
29            ret(j) = y1 + (a1 + (a2 + a3 * tmp) * (tmp - 1.)) * tmp;
30
31            ++j;
32        } else {
33            // Otherwise, go to next interval
34            ++i_star;
35            // Terminate if outside any interval
36            if (i_star >= x.size() - 1) break;
37            continue;
38        }
39    }
40
41    // END
42    return ret;
43 }
```

Get it on 🐘 [GitLab](#) (pchi.hpp).