

C++ code 8.19.14: Efficient implementation of `groundState_fpit()`

```
2 Eigen::VectorXd groundState_fpit(unsigned int n, double rtol = 1.0E-8,
3                                 double atol = 1.0E-10,
4                                 unsigned int maxit = 20) {
5     const double h = 1.0 / (1.0 + n);
6     // Vector for storing the iterates
7     Eigen::VectorXd u(n);
8     // Initialize the matrix A
9     Eigen::MatrixXd A(n, n);
10    // Abuse the vector u as temporary storage to avoid an excessive
11    // number of calls to the exponential function.
12    for (unsigned int j = 0; j < n; ++j) {
13        u[j] = 2.0 / h * std::exp(h * j);
14    }
15    for (unsigned int l = 0; l < n; ++l) {
16        A(l, l) = 2.0;
17        for (unsigned int j = 0; j < l; ++j) {
18            A(l, l) += u[l - j];
19        }
20        for (unsigned int j = l + 1; j < n; ++j) {
21            A(l, l) += u[j - l];
22        }
23        for (unsigned int j = 0; j < l; ++j) {
24            A(l, j) = A(j, l) = u[l - j];
25        }
26    }
27    // Precompute LU factorization of the matrix A, see
28    // [Lecture → Code 2.5.0.12]
29    auto A_ludec = A.lu();
30    // Initial guess zero
31    u.setZero();
32    // Main loop for fixed-point iteration
33    unsigned int it_cnt = 0; // iteration count
34    double corr_norm;       // Norm of correction
35    Eigen::VectorXd b(n);   // Stores the vector b(u)
36    do {
37        // Initialize vector b(u)
38        for (unsigned int j = 0; j < n; ++j) {
39            b[j] = -h * std::exp(u[j]);
40        }
41        // Backward and forward substitution
42        Eigen::VectorXd u_new = A_ludec.solve(b); //
43        corr_norm = (u - u_new).norm();
44        // Next iterate
45        u = u_new;
46    } while ((corr_norm > rtol * u.norm()) && (corr_norm > atol) &&
47            (++it_cnt < maxit));
48    return u;
49 }
```