

C++ code 8.18.4: Implementation of `eigenwton()`

```
2  std::pair<double, Eigen::VectorXd> eigenwton(const Eigen::MatrixXd& A,
3                                              double rtol = 1.0E-6,
4                                              double atol = 1.0E-8,
5                                              unsigned int maxit = 20) {
6      const unsigned int n = A.cols();
7      assert(A.rows() == n);
8      if ((A - A.transpose()).norm() > 1.0E-16 * A.norm()) {
9          throw std::runtime_error("Matrix A must be symmetric");
10     }
11     // Vector storing the iterates, (z)1:n ↔ x, (z)n+1 ↔ λ
12     Eigen::VectorXd z = Eigen::VectorXd::Zero(n + 1);
13     // Determine initial guess from maximal modulus diagonal element
14     Eigen::Index max_idx;
15     z[n] = A.diagonal().maxCoeff(&max_idx);
16     z[max_idx] = std::sqrt(2);
17     // Variables for use within the main Newton loop
18     Eigen::VectorXd Fz(n + 1);           // For storing F(z)(k)
19     Eigen::VectorXd s(n + 1);           // Newton correction
20     Eigen::MatrixXd DF(n + 1, n + 1);   // Jacobi matrix
21     double s_norm;                      // Norm of Newton correction
22     unsigned int it_cnt = 0;             // Iteration counter
23     // Main Newton iteration loop
24     do {
25         // Assemble the Jacobi matrix DF(z)(k) from its blocks
26         DF.block(0, 0, n, n) = A - z[n] * Eigen::MatrixXd::Identity(n, n);
27         DF.block(0, n, n, 1) = -z.head(n);
28         DF.block(n, 0, 1, n) = -z.head(n).transpose();
29         DF(n, n) = 0.0;
30         // Evaluate F(z)(k)
31         Fz.head(n) = A * z.head(n) - z[n] * z.head(n);
32         Fz[n] = 1 - 0.5 * z.head(n).squaredNorm();
33         // Compute Newton correction s := DF(z)(k)-1F(z)(k) by Gaussian
34         elimination
35         s = DF.lu().solve(Fz);
36         s_norm = s.norm();
37         // Next iterate z(k+1)
38         z -= s;
39     } while ((s_norm > rtol * z.norm()) && (s_norm > atol) && (++it_cnt < maxit));
40     return {z[n], z.head(n)};
41 }
```