

SOLUTION of (4-1.d):

The function `frequency_content()` evaluates $V(\mathbf{M})$ as defined in (4.1.1) for a given (real) matrix $\mathbf{M} \in \mathbb{R}^{n,m}$.

C++ code 4.1.4: Computation of high-frequency content.

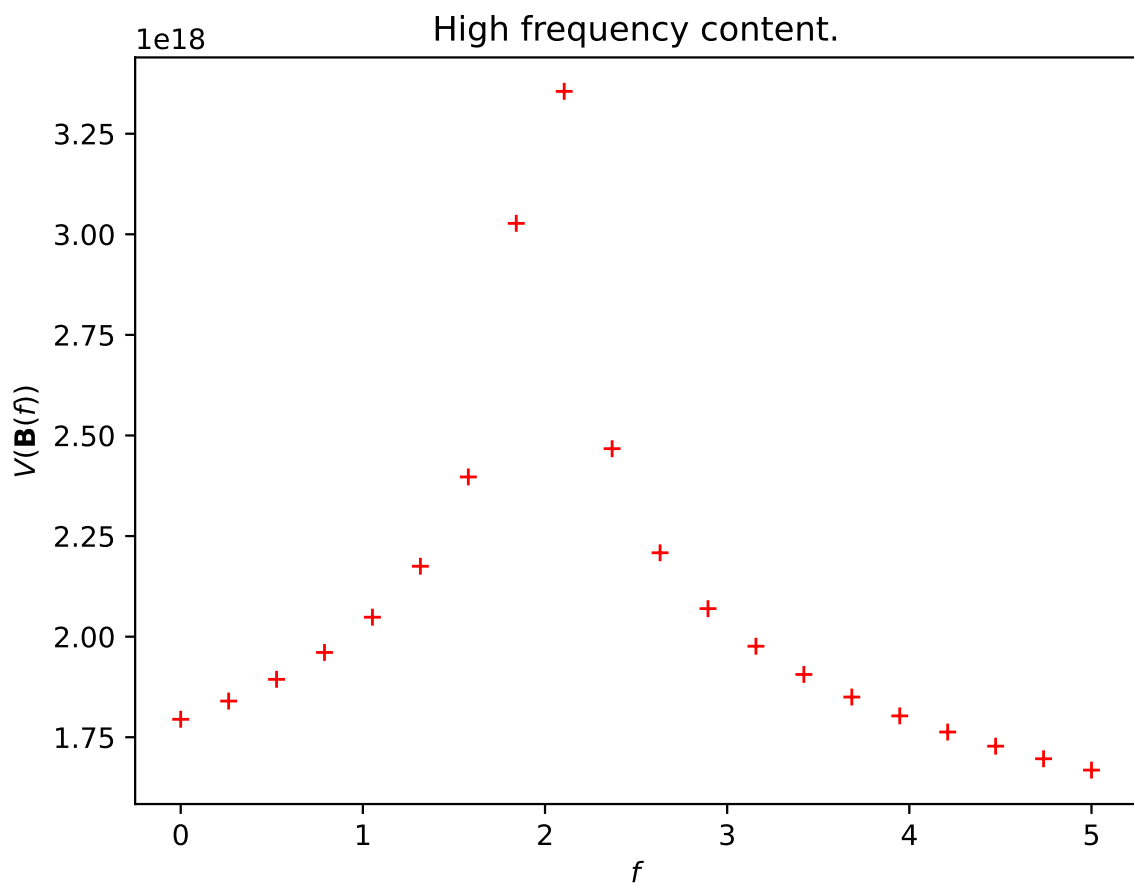
```
2 double high_frequency_content(const Eigen::MatrixXd &M) {
3     const unsigned int n = M.rows(), m = M.cols();
4     double V = 0;
5     // TODO: (4-1.d) compute V(M).
6     // START
7     for (unsigned int i = 0; i < M.rows(); ++i) {
8         for (unsigned int j = 0; j < M.cols(); ++j) {
9             double a = n / 2. - std::abs(i - n / 2.);
10            double b = m / 2. - std::abs(j - m / 2.);
11            V += (a * a + b * b) * M(i, j) * M(i, j);
12        }
13    }
14    // END
15    return V;
16 }
```

Get it on  GitLab (autofocus.hpp).

C++ code 4.1.5: Plot creation for $V(\mathbf{B}(f))$.

```
2 void plotV() {
3     constexpr unsigned int N = 20;
4
5     Eigen::VectorXd x(N), y(N);
6
7     // TODO: (4-1.d) Plot V(B(f)).
8     // START
9     Eigen::FFT fft;
10    for (unsigned int i = 0; i < N; ++i) {
11        double V = high_frequency_content(
12            // Find 2D spectrum of matrix B(t)
13            fft.fwd(
14                // Evaluate set_focus at equidistant points
15                set_focus(5. / (N - 1) * i))
16            .cwiseAbs());
17        x(i) = 5. / (N - 1) * i;
18        y(i) = V;
19    }
20    // END
21
22    plt::figure();
23    plt::title("High frequency content.");
24    plt::plot(x, y, "r+", {{ "label", "V(\mathbf{B}(f))" }});
25    plt::xlabel("f");
26    plt::ylabel("V(\mathbf{B}(f))");
27    plt::savefig("./cx_out/focus_plot.png");
28 }
```

Get it on  GitLab (autofocus.hpp).



There is a pronounced peak of $f \rightarrow V(\mathbf{C}(f))$ for a particular value of f ! For that value the blurred picture has maximal high-frequency content.
