NumCSE

Autumn Semester 2017

Prof. Rima Alaifari

Exercise sheet 5
Filtering

P. Bansal

Problem 5.1: FFT and least squares

In this problem, we look at an application of fast Fourier transform in the context of a least squares problem.

A planet orbits the sun in a closed, planar curve and its distance from the sun at polar angles $\varphi_j = 2\pi \cdot \frac{j}{n}$ is measured to be d_j, for $j = 0, \ldots, n-1$.

Define $\mathcal{P}_m^C$, $m \in \mathbb{N}$, as the space of real trigonometric polynomials of the form:

$$p(t) = \sum_{k=0}^{m} c_k \cos(2\pi k t), \quad c_k \in \mathbb{R}. \tag{5.1}$$

We choose a $p \in \mathcal{P}_m^C$ to approximate the planet's trajectory, i.e. p maps the polar angle to the distance from the sun. Therefore, the objective is to find an optimal $p^* \in \mathcal{P}_m^C$ such that:

$$p^* = \underset{p \in \mathcal{P}_m^C}{\operatorname{argmin}} \sum_{j=0}^{n-1} \left| p\left(\frac{\varphi_j}{2\pi}\right) - d_j \right|^2.$$

Assumption: $2m < n$ or more generally $m << n$.

(a) Reformulate the objective as a standard linear least squares problem (LSQ)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{m+1}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

with suitable matrix $\mathbf{A}$ and vectors $\mathbf{b}, \mathbf{x}$.

SOLUTION:

We have

$$p\left(\frac{\varphi_j}{2\pi}\right) = p\left(\frac{j}{n}\right) = \sum_{k=0}^{m} c_k \cos\left(2\pi k \frac{j}{n}\right) = \sum_{k=0}^{m} c_k \cos\left(\frac{2\pi}{n} kj\right).$$

Hence,

$$\sum_{j=0}^{n-1} \left| p\left(\frac{\varphi_j}{2\pi}\right) - d_j \right|^2 = \left\| \begin{bmatrix} p\left(\frac{\varphi_0}{2\pi}\right) - d_0 \\ p\left(\frac{\varphi_1}{2\pi}\right) - d_1 \\ \vdots \\ p\left(\frac{\varphi_{n-1}}{2\pi}\right) - d_{n-1} \end{bmatrix} \right\|_2^2 := \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2,$$

where

$$\mathbf{A} = \begin{bmatrix} \cos(0) & \cos(0) & \cos(0) & \ldots & \cos(0) \\ \cos(0) & \cos(\frac{2\pi}{n}) & \cos(\frac{4\pi}{n}) & \ldots & \cos(\frac{2\pi m}{n}) \\ \cos(0) & \cos(\frac{4\pi}{n}) & \cos(\frac{8\pi}{n}) & \ldots & \cos(\frac{4\pi m}{n}) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \cos(0) & \cos(\frac{2\pi(n-1)}{n}) & \cos(\frac{4\pi(n-1)}{n}) & \ldots & \cos(\frac{2\pi(n-1)m}{n}) \end{bmatrix}_{n \times (m+1)}$$

$$\mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \ldots \\ c_m \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} d_0 \\ d_1 \\ \ldots \\ d_{n-1} \end{bmatrix}.$$

(b) State the normal equations for this LSQ problem in explicit form. Use the following:

$$\cos(2\pi x) = \frac{1}{2}(e^{2\pi i x} + e^{-2\pi i x}), \quad i^2 = -1 \tag{5.2a}$$

$$\sum_{\ell=0}^{n-1} q^\ell = \frac{1-q^n}{1-q} \quad \text{(sum of a geometric series)} \tag{5.2b}$$

SOLUTION:

The normal equations are

$$\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b}.$$

The (j,k)-th entry of $\mathbf{A}^\top \mathbf{A}$, which is $(m+1) \times (m+1)$ matrix, is

$$\begin{aligned}
(\mathbf{A}^\top \mathbf{A})_{j,k} &= \sum_{\ell=0}^{n-1} \cos\left(\frac{2\pi j \ell}{n}\right) \cos\left(\frac{2\pi k \ell}{n}\right) \\
&= \frac{1}{4} \sum_{\ell=0}^{n-1} \left(e^{\frac{2\pi j \ell i}{n}} + e^{-\frac{2\pi j \ell i}{n}}\right)\left(e^{\frac{2\pi k \ell i}{n}} + e^{-\frac{2\pi k \ell i}{n}}\right) \\
&= \frac{1}{4} \sum_{\ell=0}^{n-1} \left(e^{\frac{2\pi (j+k) \ell i}{n}} + e^{\frac{2\pi (j-k) \ell i}{n}} + e^{\frac{2\pi (k-j) \ell i}{n}} + e^{-\frac{2\pi (j+k) \ell i}{n}}\right) \\
&= \begin{cases} n, & j = k = 0, \\ n/2, & j = k \neq 0, \\ 0, & j \neq k. \end{cases}
\end{aligned}$$

Thus,

$$\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} n & & & \\ & n/2 & & \\ & & \ddots & \\ & & & n/2 \end{bmatrix}.$$

(c) Implement a C++function:

```
VectorXd triPolyFit(const VectorXd & d, unsigned int m);
```

that computes the coefficients $c_k \in \mathbb{R}$ of p^*, for $k = 0, 1, \ldots, m$, in the form (5.1) for arbitrary inputs $d_j \in \mathbb{R}$, for $j = 0, 1, \ldots, n-1$, passed in a vector `d`. The function must not have an asymptotic complexity worse than $O(n \log n)$.

Use the EIGEN class `FFT<double>`. To this end, include header "`<unsupported/Eigen/FFT>`.
Recall:

$$\sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} jk} d_j = \overline{\sum_{j=0}^{n-1} e^{-\frac{2\pi i}{n} jk} d_j},$$

where the over-line denotes the complex conjugation.

SOLUTION:

The j-th entry of the right hand-side vector of the normal equations $\mathbf{A}^\top \mathbf{b}$ is:

$$\begin{aligned}(\mathbf{A}^\top \mathbf{b})_k &= \sum_{j=0}^{n-1} \cos\left(\frac{2\pi jk}{n}\right) d_j \\ &= \frac{1}{2}\sum_{j=0}^{n-1} \left(e^{-\frac{2\pi i}{n}jk} + e^{\frac{2\pi i}{n}jk}\right) d_j \\ &= \frac{1}{2}\sum_{j=0}^{n-1} e^{-\frac{2\pi i}{n}jk} d_j + \frac{1}{2}\sum_{j=0}^{n-1} e^{\frac{2\pi i}{n}jk} d_j \\ &= \frac{1}{2}\left((\mathbf{F}_n \mathbf{b})_k + \overline{(\mathbf{F}_n \mathbf{b})_k}\right) \\ &= (\mathrm{Re}(\mathbf{F}_n \mathbf{b}))_k\end{aligned}$$

C++11-code 5.1: Implementation of `triPolyFit`.

```
 */
VectorXd triPolyFit(const VectorXd & d,
                    unsigned int m) {
    unsigned int n = d.size();

    // We will use a real to complex, discrete Fourier transform.
    FFT<double> fft;
    // Computing Fourier coefficients
    VectorXcd fft_d = fft.fwd(d);
    // Solving normal equation by inverting diagonal matrix
    VectorXd coeffs = fft_d.head(m).real() / n;
    coeffs.tail(m-1) *= 2;

    return coeffs;
```

(d) Test your implementation by fitting a trigonometric polynomial of degree 3 with the data values `d` provided in `main()` of `fftlsq.cpp`.

Expected coefficients: $[0.984988, -0.00113849, -0.00141515]$

Problem 5.2: Autofocus with FFT

In this problem, we try to understand how we can find the "best focused" image amongst a collection of out-of-focus photos using 2D frequency analysis.

A grey-scale image consisting of $n \times m$ pixels is given as a matrix $\mathbf{P} \in \mathbb{R}^{n,m}$, where an element of the matrix corresponds to the gray-value of the pixel as a number between 0 and v_{max}. We regard this as the **perfect image**.

If a camera is poorly focused due to an improper arrangement of the lenses, it will record a **blurred image** $\mathbf{B} \in \mathbb{R}^{n,m}$. The blurring operation can be modeled as a **2D (periodic) discrete convolution**. The **perfect image** can be recovered by **deconvolution**, provided that the **point spread function (PSF)** is known. In this problem, we assume no prior knowledge of the PSF. The only data available is from the "black-box" C++ function (defined in `setFocus.hpp`):

```
MatrixXd set_focus(double f);
```

which returns a potentially blurred image $\mathbf{B}(f)$ for an input value of the focus parameter f. The focus parameter can be understood as the distance between the lenses of a camera, which can be changed through turning on and off a stepper motor.

Objective of autofocusing: Determine an optimal focus parameter f which minimizes the blur in the image $\mathbf{B}(f)$.

Core idea: *The less an image is marred by blur, the larger its "high frequency content"*.

Quantiative notion: For an image $\mathbf{C} \in \mathbb{R}^{n,m}$, "high frequency content" can be quantified by the second moments of its 2D DFT, $\hat{\mathbf{C}} \in \mathbb{C}^{n,m}$, as

$$V(\mathbf{C}) = \sum_{k_1=0}^{n-1} \sum_{k_2=0}^{m-1} \left(\left(\frac{n}{2} - \left|k_1 - \frac{n}{2}\right|\right)^2 + \left(\frac{m}{2} - \left|k_2 - \frac{m}{2}\right|\right)^2 \right) |\hat{\mathbf{C}}_{k_1,k_2}|^2 .$$

Problem statement: Find $f \in \mathbb{R}$ such that $V(\mathbf{B}(f))$ becomes maximal.

(a) Compare the blurred images and their respective frequency content for $f = 0, 1, 2, 3$ using

```
void save_image(double f);     void plot_freq(double f);
```

[See `utilities.hpp`]

(b) Implement a C++ function which returns the value $V(\mathbf{C})$ for a given matrix $\mathbf{C}$:

```
double high_frequency_content(const MatrixXd & C);
```

SOLUTION:

C++11-code 5.2: Computation of high-frequency content.

```
1 double high_frequency_content(const MatrixXd & M) {
2     int n = M.rows() ,m = M.cols();
3     double V = 0;
4     for(unsigned int i = 0; i < M.rows(); ++i) {
5         for(unsigned int j = 0; j < M.cols(); ++j) {
6             double a = n/2. - std::abs(i - n/2.);
7             double b = m/2. - std::abs(j - m/2.);
```

```
8                 V += (a*a + b*b) * M(i,j) * M(i,j);
9             }
10        }
11        return V;
12    }
```

(c) Study the variation of $V(\mathbf{B}(f))$ w.r.t. $f \in [0,5]$ using the function:

```
void plotV(unsigned int N);
```

Based on the observed structure of the function $V(\mathbf{B}(f))$, what is the optimal focus?

[See `autofocus.cpp`]

Important remark: Practically, to determine the optimal focus parameter we need to use optimization methods. For this problem, you do not need to know how we compute the 2D DFT. However, the important point is to understand the usefulness of the frequency data provided by DFT for image analysis. It might also be interesting for you to look at the frequency content of images with higher quality.