Advanced Numerical Algorithms (MU4IN920)

Tutorial-Practical 3 - Fast Fourier Transform

1. Tutorial

Exercise 1 (Practice with the fast Fourier transform)**.**

1. What is the sum of the n-th roots of unity?
2. What is their product if n is even and if n is odd?
3. What is the FFT of $(1, 0, 0, 0)$? What is the appropriate value of ω in this case? And of which vector is $(1, 0, 0, 0)$ the FFT?
4. Find the unique polynomial of degree 4 that takes on values $p(1) = 2$, $p(2) = 1$, $p(3) = 0$, $p(4) = 4$ and $p(5) = 0$.

Exercise 2 (Practice with polynomial multiplication by FFT)**.** Suppose that you want to multiply the two polynomials $x + 1$ and $x^2 + 1$ using FFT. Choose an appropriate power of 2, find the FFT of the 2 sequences, multiply the result componentwise, and compute the inverse FFT to get the final result.

Exercise 3 (FFT using modular arithmetic)**.** As defined, the discrete Fourier transform requires computation with complex numbers, which can result in a loss of precision due to round-off errors. For some problems, the answer is known to contain only integers, and a variant of the FFT based on modular arithmetic can guarantee that the answer is calculated exactly. An example of such a problem is that of multiplying two polynomials with integer coefûcients.
Let n be a power of 2 and k the smallest integer such that $p = kn + 1$ is prime. Let g be a generator of $(\mathbb{Z}/p\mathbb{Z})^*$ and let $\omega = g^k \pmod p$.

1. Show that ω is a primitive n-th root of unity in $\mathbb{Z}/p\mathbb{Z}$.
2. Explain why the FFT and the inverse FFT are well defined when ω is a primitive root n-th of the unit in $\mathbb{Z}/p\mathbb{Z}$.
3. Explain why the FFT and its inverse can be computed in $\mathcal{O}(n \log n)$ if we suppose that the cost of the operations on the words of length $\mathcal{O}(\log n)$ is constant. We will assume that p and ω are given.
4. Compute the FFT modulo $p = 17$ of the vector $(0, 5, 3, 7, 7, 2, 1, 6)$.

2. Practical

Exercise 4 (recursive and iterative FFT)**.**

1. Implement the recursive version of the FFT.
2. Implement the iterative version of the FFT.
3. Compare in terms of efficiency your two implementations.
4. Then compare your implementations with the MATLAB command `fft`.

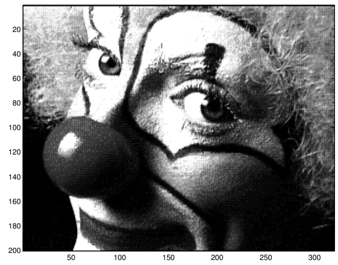

Figure 1: Image size 320 × 200 pixels

Exercise 5 (Image compression via FFT)**.** Figure 1 represents an image of 320 × 200 pixels corresponding to a matrix X of size 320 × 200.
The display of the image in MATLAB is done as follows:

```
load clown.mat;
colormap('gray');
image(X);
```

We recall that we note

$$M_n(\omega) = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ & & \vdots & & \\ 1 & \omega^j & \omega^{2j} & \cdots & \omega^{(n-1)j} \\ & & \vdots & & \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix} \text{ with } \omega = e^{2i\pi/n}.$$

How to compress an image with the FFT ? After all, an image is a 2-dimensional array and so far we have only talked about only talked about the FFT of a one dimensional array. Let X be a matrix of size $m \times n$ (or an image). The FFT in 2 dimensions consists in to compute $ffX = M_m X M_n$. We can implement this calculation in the following way:

- for each column of X, we compute its FFT. Let us note fX the matrix obtained. In other words, the column i of fX is the FFT of the column i of X.
- for each row of fX, we compute its FFT. Let us note ffX the matrix obtained. In other words, the row i of ffX is the FFT of the line i of fX.

The matrix ffX is the 2-dimensional FFT of X. We use ffX for compression in the following way. We perform the FFT of the image but we store only the non-zero elements. In fact, to compress, we do not only remove the zeros but also all the elements whose absolute value is smaller than a threshold threshold ε.

1. Using the FFT, propose an algorithm for image compression. Test your algorithm on Figure 1.
2. Calculate the compression rate allowing to measure the quality of compression of the images.

Exercise 6 (Multiplication of polynomials using FFR)**.** Implement the algorithm seen in class to calculate the product of two polynomials using FFT. Plot the time as a function of the degree of the polynomials. Compare with the classical method of polynomial multplication.