1 Laboratory 5: Compressed sensing

EXERCISE 1 *Make a program that solves the following PROBLEM 1:*

1. *We have a polynomial* $p(z) = a_0 + a_1 z + \cdots + a_{N-1} z^{N-1}$ *of degree* $N-1$ *(with* N *very big) but we know that most of its coefficients are 0. That is, we know that only* K *coefficients are non zero and* K *is much smaller than* N.
   Suppose that we know N *and* K *but we don't know the polynomial. Assume that we can evaluate the polynomial at any point. How many evaluations do we need and what is a possible scheme to recover the coefficients?*

In order to do so, we need to hard code a sparse trigonometric polynomial (say $p(x) = 2sin(x) + sin(2x) + 21sin(300x)$*) and then try to guess it by evaluating it on real points only 30 or 40 times taken at random in [0, 400]. We only know that it is a trigonometric polynomial of degree smaller than 400 and with few components.*

As given by the assumptions, we have an N-th degree sparse trigonometric polynomial, with $N \leq 400$, this is

$$\begin{aligned} T(x) &= \frac{1}{2}a_0 + \sum_{n=1}^{N} a_n cos(nx) + \sum_{n=1}^{N} b_n sin(nx) \\ &= \frac{1}{2}a_0 + \sum_{n=1}^{N} (a_n cos(xn) + b_n sin(nx), \end{aligned}$$

with $x \in \mathbb{R}$, $a_n, b_n \in \mathbb{C}$, $0 \leq n \leq N$. By applying the Euler's formula, we have the equivalent expression

$$T(x) = \sum_{n=-N}^{N} c_n e^{inx}$$

with $c_{-k} = \overline{c_k}$ and therefore, we can deduce that $a_k = c_k + c_{-k}$ and $b_k = i(c_k - c_{-k})$. Observe additionally that on this notation, c_k represents the n-th Fourier coefficient of T.

We seek to find the estimates of the non vanishing coefficients of the trigonometric polynomial (in this case, $K = 3 \ll N$), without knowing the value of K a priori. In order to do so, we evaluate the given polynomial at $M = 40$ random points from $[0, 400]$. Thus, we are approximating a trigonometric polynomial of degree $K \leq M$.

```
% Step 1: Create the sparse trigonometric polynomial
x = sym('x');
p = 2*sin(x) + sin(2*x) + 21*sin(300*x);
% Step 2: Generate random evaluation points
numPoints = 40;  % Number of evaluation points
evalPoints = 400 * rand(numPoints, 1);  % Random points in [0, 400]
% Step 3: Evaluate the polynomial at the generated points
evaluations = eval(subs(p, x, evalPoints));
```

As stated bu Tao, Candès and Romberg to reconstruct the sparse polynomial we may consider the standard linear programming solver `linprog` in order to find the solution of the l_1 minimization problem, this is solving the following problem:

$$min_g \sum_{k=0}^{N-1} |g[k]|, \quad \text{such that } G(w) = F(w), \forall w \in \Omega$$

where here the constraints are given by the evaluation of the polynomial at the M random points, and the minimization is exactly the l_1 norm.

Applied to our specific problem, this corresponds to considering the constraint matrix defined by the evaluation points, this is, for $x_k \in [0, 400]$, $k = 1, \dots, M$, we have the pairs $(x_k, y_k) = (x_k, p(x_k))$ obtained by evaluating the initial given trigonometric polynomial. Now, for the approximation trigonometric polynomial $T(x)$ as aforementioned, we consider the constraints defined as:

$$T(x_k) = y_k \iff \sum_{n=-N}^{N} c_n e^{inx_k} = y_k, \ k = 1, \dots, M$$

which, since $M < N$ clearly represents a heavily undetermined linear system, with infinite possible solutions. Therefore, since we are looking for a sparse polynomial, we may focus on minimizing the l_1 norm, in order to find the solution with the maximum vanishing coefficients. Thus, the minimizations problem becomes

$$min_c \sum_{n=-N}^{N} |c_n| \ \text{ subject to } \ \sum_{n=-N}^{N} c_n e^{inx_k}$$

Observe however that when considering the definition of the constraint matrix, we have some more restrictions, given by the relationship of the coefficients, this is $c_{-k} = \overline{c_k}$, adding more restrictions to the linear problem.

Assuming that we are considering a real trigonometric polynomial (since the required MATLAB function just accept real coefficient constraint matrices), we can consider the real part of the polynomial, given by the sinus part. Thus, when considering the constrained matrix only the sinus coefficients are considered. The result obtained by implementing the aforementioned algorithm is the approximated polynomial

$$T(x) = 2sin(2x) + sin(3 * x) + 21sin(301x)$$

which is actually a sparse trigonometric polynomial with $K = 3$ non vanishing coefficients, that lay slightly unmatched with the original polynomial.
The comparison between the initial and the reconstructed polynomials is shown on Figure 1.

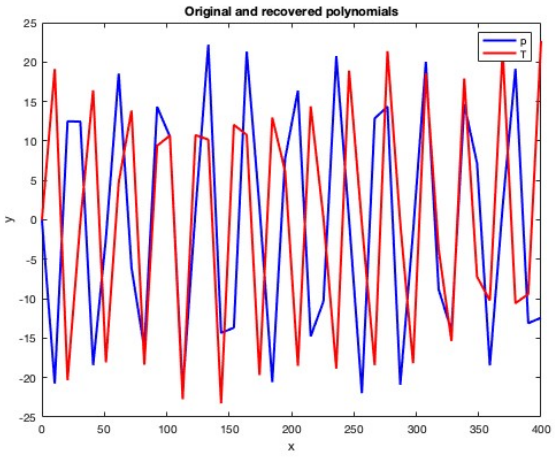

Figure 1: Original and reconstructed trigonometric polynomials.

The corresponding code with the algorithm followed and the optimization linear problem solving is presented on code `LAB5.m`.