# THE LINEAR METHOD IN PROBLEM SOLVING

LUIS FERRONI

Abstract. In these notes we address some techniques coming from basic linear algebra and linear programming in the context of problem solving. The exposition is intended to be expositive rather than comprehensive, and motivated by concrete examples of problems. They were written as part of the lectures given by the author in the Argentinian Training Camp in Competitive Programming held virtually during July 2021.

## 1. The square-subset problem

**Problem 1.1.** Let $S = \{a_1, \ldots, a_n\}$, $1 \leq n \leq 1000$ be a set of positive integers that have a nice property: all of them are divisible only by primes smaller or equal than 497. We want to know how many non-empty subsets of $S$ have the property that the product of all of its elements is a *perfect square*.

For example if $S = \{2, 3, 6, 8\}$, then we can choose $\{2, 8\}$, $\{2, 3, 6\}$ and $\{3, 6, 8\}$. These are the only possibilities. So, there are only 3 among the $2^4 = 16$ subsets of $S$ that have the required property.

Notice that in the general case, iterating over all subsets of $S$ would require $2^n$ flops which would be very slow for $n$ larger than 25. In other words, we have to come up with something better.

## 2. An overview of linear algebra

Its ubiquity in all branches of mathematics makes of linear algebra a fundamental tool both from the theoretic and the applied points of view.

Let us review the basic definitions so we can get us to problem solving very quickly. We will give a brief summary of the basic concepts of the topic that we will address here.

Consider the system of equations

$$\begin{cases} x + y + z &= 4 \\ x + 2y + 3z &= 7 \\ 4x + 5y + 6z &= 19 \end{cases}$$

By using the matrices

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 4 \\ 7 \\ 19 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

it is customary to rewrite the system more compactly as

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

In many situations it is desirable to *solve* the system of equations. In other words, it is required to fully determine which are **all** the values of $\mathbf{x}$ that make the equality $A\mathbf{x} = \mathbf{b}$ a

true statement. The most popular method to accomplish that consists on using *Gaussian elimination* on the so-called *extended matrix*, which is given by

$$A_{\text{ext}} = \left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 1 & 2 & 3 & 7 \\ 4 & 5 & 6 & 19 \end{array}\right].$$

What we obtain after reducing the above matrix is a new matrix with the property that its left part is in *row-echelon-reduced form*. In our particular case, what we get is

$$\text{red}(A_{\text{ext}}) = \left[\begin{array}{ccc|c} 1 & 0 & -1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \end{array}\right].$$

Hence, we make the following observations:

- If in the left part there is a row of zeros, then in the right part within that row there has to be zero. Otherwise, there is *no solution* to the system.
- If we had not extended $A$ to $A_{\text{ext}}$, and we had directly reduced just $A$, we would have obtained
$$\text{red}(A) = \left[\begin{array}{ccc} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{array}\right].$$
- There are some entries of the matrix $\text{red}(A_{\text{ext}})$ that are *special*.
$$\text{red}(A_{\text{ext}}) = \left[\begin{array}{ccc|c} \boxed{1} & 0 & -1 & 1 \\ 0 & \boxed{1} & 2 & 3 \\ 0 & 0 & 0 & 0 \end{array}\right].$$
These elements are 1's that were used during the reduction to *erase* the rest of the elements in its columns. We call them *pivots*.
  - Notice that not all columns have a pivot but, when there is one, the rest of the elements in that column **must** be zero.
  - Notice that there cannot be two pivots in the same row.

Why do we care about pivots? The most important reason is because they tell us how to solve the system of equations. Just take a look at which columns **do not have pivots** (in our case, only the third column, which was associated to the variable $z$). All the columns that do not have pivots are associated to a *free variable* (i.e. we can assign whatever value we want to those variables). The columns that have a pivot are associated to variables that can be recovered from the values assigned to the free variables.

In the above example we see that the variable $z$ is free, so we know that the variables associated to the columns with pivots (i.e. $x$ and $y$) can be obtained *from* $z$. In fact:

$$x = 1 + z, \quad y = 3 - 2z,$$

so that if (for example) we choose $z = 10$, then a valid solution would be $(x, y, z) = (11, -17, 10)$. If we want a shorter and more general expression, we can just say

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1 \\ -3 \\ 0 \end{bmatrix} + z \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}.$$

Observe that here we see that the set of solutions of our system is essentially a *line* in $\mathbb{R}^3$ (i.e., it has *dimension* 1).

Let us turn into the general case, in which we have a matrix $A$ of size $m \times n$, and $\mathbf{b}$ is a vector of size $m \times 1$, and we consider the system

$$A\mathbf{x} = \mathbf{b},$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$

Let us assume that we had exactly $s$ free variables (i.e. $s$ columns in $\text{red}(A_{\text{ext}})$ that did not have a pivot). Then proceeding as above, we can write all the $n - s$ variables that *did* have a pivot as a function of these $s$ variables. In other words, all the $n$ variables depend only on $s$ of them. We can put all these pieces together into a theorem.

**Theorem 2.1.** *Consider the system of equation $A\mathbf{x} = \mathbf{b}$. The dimension of the space of solutions to this system is $s$, where $s$ is the number of columns in* $\text{red}(A_{ext})$ *that do not have a pivot. Moreover, this number $s$ does not depend on* $\mathbf{b}$.

The last statement comes from the fact that we are only looking at columns of $\text{red}(A)$ which, as we said above, is the left part of $\text{red}(A_{\text{ext}})$.

Now, we deviate for an instant from the world of system of equations. There is a famous number associated to a matrix, whose definition we will repeat here.

**Definition 2.2.** The *rank* of a matrix $A$ is defined as the maximum number of linearly independent rows that one can choose from $A$.

It is straightforward to see that if $A$ has $m$ rows, then $\text{rank}(A) =$ number of rows that are non zero in $\text{red}(A)$. However, if we think for a second, a row of $\text{red}(A)$ is non zero if and only if it has a pivot. Hence,

**Theorem 2.3.** *The rank of $A$ is the number of pivots of* $\text{red}(A)$.

Hence, since $s$ is the number of columns without pivots and $n$ is the total number of columns, the number of columns that have a pivot is $n - s$. Hence, the above theorem states that $\text{rank}(A) = n - s$. Or, more briefly,

**Theorem 2.4** (Rank - Nullity)**.** *Let $A$ be a matrix with $n$ columns. Let $s$ be the dimension of the space of solutions $A\mathbf{x} = \mathbf{b}$ (for whatever vector $\mathbf{b}$, for example $\mathbf{b} = \mathbf{0}$). Then*

$$n = \text{rank}(A) + s.$$

## 3. The solution to the square-subset problem

So far we have just done some abstract nonsense. Let us see how all of the above results yield to a solution of the problem we posed in the first section.

To this end, assume that we are given $S = \{a_1, \ldots, a_n\}$ where the prime divisors of each $a_i$ are at most 497. There are 95 primes less or equal than 497. In particular, for each of the numbers $a_i$ we can consider a vector with 95 integers, given by the exponents of each of the primes in its factorization. For example,

$$45 \mapsto (0, 2, 1, 0, \ldots, 0, 0),$$
$$497 \mapsto (0, 0, 0, 0, \ldots, 0, 1),$$
$$144 \mapsto (4, 2, 0, 0, \ldots, 0, 0), \text{ etc.}$$

It is clear that a number $x$ is a perfect square if and only if all the numbers in its vector of exponents are even. Also, the vector of the product of two numbers is clearly the sum

of both vectors (so, we transform products into sums). In particular, a subset of elements of $S$ has a square product if and only if the coordinate-wise sum of their vectors has even entries. This hints us that we may instead consider the parity of the exponents of each of the numbers. We can put a 1 if the corresponding exponent is odd and a zero if it is even. So that, in reality, we are looking at

$$\begin{aligned} 45 &\mapsto (0,0,1,0,\ldots,0,0) = \mathrm{bi}(45),\\ 497 &\mapsto (0,0,0,0,\ldots,0,1) = \mathrm{bi}(497),\\ 144 &\mapsto (0,0,0,0,\ldots,0,0) = \mathrm{bi}(144),\ \text{etc.} \end{aligned}$$

So that now a number is a square if and only if its vector consists only of zeros. Now, let us consider the following matrix

$$A = \begin{bmatrix} | & | & & | \\ \mathrm{bi}(a_1) & \mathrm{bi}(a_2) & \cdots & \mathrm{bi}(a_n) \\ | & | & & | \end{bmatrix},$$

where in the column $i$ we put $\mathrm{bi}(a_i)$. In other words, we have that $A$ has size $95 \times n$. Observe that if we choose a subset of $S = \{a_1, \ldots, a_n\}$, suppose $\{a_3, a_8, a_{10}\}$, then it holds $\mathrm{bi}(a_3 a_8 a_{10}) = A\mathbf{x}$ where $\mathbf{x}$ has only $0/1$ entries as follows: $x_3 = x_8 = x_{10} = 1$ and the rest of them are zeros.

In other words, by considering the vector $\mathbf{x}$ that has a 1 in the positions of the number that we choose from $S$ and a zero in the positions of the numbers that we *do not* choose, we have translated square-subset problem into:

$$\text{find all vectors of zeros and ones } \mathbf{x} \text{ such that } A\mathbf{x} = \mathbf{0}.$$

Observe that now our situation is very similar to what we have developed in the previous section. We knew that when dealing with $\mathbf{x}$ having *real* coordinates, then the dimension of the space of vectors such that $A\mathbf{x} = \mathbf{b}$ is the number of columns without pivots in $\mathrm{red}(A)$. A proof entirely analogous shows that this still holds if we allow our variables to live only in the *field on two elements* $\mathbb{Z}_2$.

So, turning into our problem of square-subsets, when we reduce our matrix $A$ we have that the space of all vectors $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{0}$ is $s$, where $s$ is the number of columns of $\mathrm{red}(A)$ where there are no pivots.

Having dimension $s$ means that we can write all of our valid vectors $\mathbf{x}$ as linear combinations in $\mathbb{Z}_2$ of a certain family consisting on $s$ vectors. In other words, there are some $\{\mathbf{y}_1, \ldots, \mathbf{y}_s\}$ vectors with $0/1$ entries such that:

$$A\mathbf{x} = \mathbf{0} \iff \mathbf{x} = c_1\mathbf{y}_1 + \ldots + c_s\mathbf{y}_s,$$

since we are living now in a field with two elements, we assume that each $c_i$ is either zero or one. Here we see that there are exactly $2^s$ possible values of $\mathbf{x}$. Since the original problem rules out the possibility of choosing the empty set, we see that the answer to our problem is $2^s - 1$, and that's it.

**Problem 3.1.** Let $S = \{a_1, \ldots, a_n\}$, $1 \leq n \leq 1000$ be a set of positive integers that have a nice property: all of them are divisible only by primes smaller than 497. We want to know how many non-empty subsets of $S$ with an odd number of elements have the property that the product of all of its elements is a *perfect square*.

**Problem 3.2.** Let $S = \{a_1, \ldots, a_n\}$, $1 \leq n \leq 10^5$ and $1 \leq a_i \leq 10^9$. For each subset of $S$ we calculate the XOR of all the elements in $S$. What is the maximum possible value that we can obtain?

## 4. Knapsacks and mathematical formulations

We will address now linear programming. To understand how often a problem can be restated as some linear programming problem, we deal first with a *super famous problem*. We restate a classical version of the *knapsack problem*.

**Problem 4.1.** We have a knapsack that has a maximum capacity of $W$. We have $n$ *fractionable* objects. The $i$-th object weights $w_i$ and its total cost is $c_i$. What is the maximum cost that we can carry on our knapsack?

This problem is solvable with a very easy greedy approach. But, take a look at its mathematical formulation. What we want is:

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under the constraint } w_1x_1 + \ldots + w_nx_n \leq W,\\ &\text{and } 0 \leq x_i \leq 1 \text{ for all } i\end{aligned}$$

Where $0 \leq x_i \leq 1$ of course denotes the *fraction* of the object $i$ to put into the knapsack.

Let us consider a variation of the above problem, but now fractioning objects is prohibited.

**Problem 4.2.** We have a knapsack that has a maximum capacity of $W$. We have $n$ *non-fractionable* objects. The $i$-th object weights $w_i$ and its total cost is $c_i$. What is the maximum cost that we can carry on our knapsack?

In many cases (for example if $W$ is not too large) the above problem can be solved with a dynamic-programming approach. Let us look at its abridged mathematical formulation.

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under the constraint } w_1x_1 + \ldots + w_nx_n \leq W,\\ &\text{and } x_i \in \{0, 1\} \text{ for all } i\end{aligned}$$

The first problem is a toy example of a *standard linear-programming problem* and the second is a toy example of *binary linear-programming problem*.

## 5. Linear programming

Let us define now what a standard linear programming problem is.

**Definition 5.1.** Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $\mathbf{b} \in \mathbb{R}^{m\times 1}$, $\mathbf{c} \in \mathbb{R}^n$ two vectors. A standard linear programming problem is a problem of the form

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under the constraint } A\mathbf{x} \leq \mathbf{b} \text{ coordinate by coordinate,}\\ &\text{and } 0 \leq x_i \text{ for all } i\end{aligned}$$

Observe that the condition $x_i \leq 1$ is not required a priori (it can be added in the inequalities given by the matrix $A$).

**Example 5.2.** Consider the problem (which is not standard)

$$\begin{aligned}&\text{minimize } x+3y-2z+3w\\&\text{under the constraints}\\&x-y-z+3w\geq 19,\ \ 2x+3y+4z-w=10\\&\text{and } 0\leq x,y,z,w\leq 11\end{aligned}$$

Observe that minimizing $x+3y-2z+3w$ is the same as maximizing $-x-3y+2z-3w$ (and then putting a minus sign).

Also, observe that $x-y-z+3w\geq 19$ is the same as $-x+y+z-3w\leq -19$. The equality $2x+3y+4z-w=10$ can be splitted into two conditions:

$$2x+3y+4z-w\leq 10$$

$$2x+3y+4z-w\geq 10$$

and, in turn, the second is equivalent to $-2x-3y-4z+w\leq -10$. So far, we have seen that the problem is equivalent to:

$$\begin{aligned}&\text{maximize } -x-3y+2z-3w\\&\text{under the constraints}\\&-x+y+z-3w\leq -19,\ \ 2x+3y+4z-w\leq 10\\&\ \ -2x-3y-4z+w\leq -10\\&\text{and } 0\leq x,y,z,w\leq 11.\end{aligned}$$

We can put the inequalities in a matrix as follows:

$$A=\begin{bmatrix}-1&1&1&-3\\2&3&4&-1\\-2&-3&-4&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix},\quad \mathbf{b}=\begin{bmatrix}-19\\10\\-10\\11\\11\\11\\11\end{bmatrix}.$$

And if $\mathbf{c}=(-1,-3,2,-3)$ we have translated our problem into the standard form.

$$\begin{aligned}&\text{maximize } c_1x+c_2y+c_3z+c_4w\\&\text{under the constraint } A\mathbf{x}\leq\mathbf{b} \text{ coordinate by coordinate,}\\&\text{and } 0\leq x,y,z,w \text{ for all } i\end{aligned}$$

5.1. **Convex Polyhedra.** Consider a matrix $A\in\mathbb{R}^{m\times n}$ and a vector $\mathbf{b}\in\mathbb{R}^{m\times 1}$. Let us consider the following set of points:

$$\mathcal{P}=\left\{\mathbf{x}\in\mathbb{R}^{n\times 1}:A\mathbf{x}\leq\mathbf{b}\text{ and }0\leq\mathbf{x}\right\}.$$

This is what we call the *feasible region* of a linear programming problem in standard form.

Observe that it might be the case that $\mathcal{P}=\varnothing$. For example, if we put the inequalities:

$$x+y\leq 1$$

$$-2x-2y\leq -2$$

they clearly are incompatible so that in this case one woulde get $\mathcal{P}=\varnothing$.

Also, it can happen that the set $\mathcal{P}$ is *unbounded*. For example, when we have an inequality:

$$x - y \leq 10,$$

then the set $\mathcal{P}$ is unbounded (even after adding the conditions $0 \leq x, y$).

When our set $\mathcal{P}$ is bounded, we say that it is a *convex polytope* (a generalization of the notion of polygon but for dimensions higher than 2).

**Remark 5.3.** If we have a linear programming problem in standard form

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under the constraint } A\mathbf{x} \leq \mathbf{b} \text{ coordinate by coordinate,}\\ &\text{and } 0 \leq x_i \text{ for all } i\end{aligned}$$

by looking at its feasible region

$$\mathcal{P} = \left\{\mathbf{x} \in \mathbb{R}^{n\times 1} : A\mathbf{x} \leq \mathbf{b} \text{ and } 0 \leq \mathbf{x}\right\},$$

what we have is just:

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under } \mathbf{x} \in \mathcal{P}.\end{aligned}$$

Why do we care about $\mathcal{P}$? Because the solution to a linear program always happens to be at a *vertex* of $\mathcal{P}$. To see why this makes sense, consider a polygon $P$ in $\mathbb{R}^2$ and try to find the point in $P$ which has the largest coordinate $y$. You will see that the largest coordinate $y$ is always achieved at a vertex of the polygon (possibly in some other non-vertices too, but we do not care). This heuristic of *maximizing $y$ in a polygon* works with some modifications to solve

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under } \mathbf{x} \in \mathcal{P}.\end{aligned}$$

But since here our $\mathcal{P}$ can be unbounded, it might be the case that the maximum value for the function $c_1x_1 + \ldots + c_nx_n$ can simply be infinity.

**Fact 5.4.** There is an algorith called *simplex* that solves the problem

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under the constraint } A\mathbf{x} \leq \mathbf{b} \text{ coordinate by coordinate,}\\ &\text{and } 0 \leq x_i \text{ for all } i\end{aligned}$$

as follows:

1. First it computes (some of)[1] the vertices of the region $\mathcal{P}$.
2. Then it iterates over the vertices of $\mathcal{P}$ and try to see if it has a neighbor that increases the function. If it does not have such neighbors, it is a vertex where the maximum is achieved (care needed if the region is unbounded).

The complexity depends on which heuristics are used to iterate in the set of vertices (random strategies usually work very well). The input of the algorithm are $A, \mathbf{b}$ and $\mathbf{c}$. The output are the maximum value of $c_1x_1 + \ldots + c_nx_n$ in the feasible region, and some point $\mathbf{x}$ for which this maximum is achieved. If the function is unbounded or the region is empty, the algorithm reports that.

[1]This may depend on the implementation of the algorithm

## 6. Binary programming and alcoved polytopes

Now we will turn our attention into linear programming where the variables **can only be zero or one**. We saw that for knapsacks the greedy approach worked when the variables could be real numbers, but a dynamic programming approach was needed to deal with the problem in which fractioning was not allowed.

**Definition 6.1.** Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $\mathbf{b} \in \mathbb{R}^{m\times 1}$, $\mathbf{c} \in \mathbb{R}^n$ two vectors. A standard **binary** linear programming problem is a problem of the form

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under the constraint } A\mathbf{x} \leq \mathbf{b} \text{ coordinate by coordinate,}\\ &\text{and } x_i \in \{0,1\} \text{ for all } i\end{aligned}$$

When one adds the condition that the variables are discrete, the situation changes drastically and the simplex algorithm does not work. This is because we actually no longer have a convex set $\mathcal{P}$, and hence we cannot be sure that the maximum would be achieved at a *vertex* (in fact, we no longer have the notion of vertex, as our feasible region is just some discrete grid of points).

But, let us consider the following standard linear programming problem.

$$\begin{aligned}&\text{maximize } c_1x_1 + \ldots + c_nx_n,\\ &\text{under the constraint } A\mathbf{x} \leq \mathbf{b} \text{ coordinate by coordinate,}\\ &\text{and } 0 \leq x_i \leq 1 \text{ for all } i\end{aligned}$$

where (of course), we have to think of the inequalities $x_i \leq 1$ as part of the matrix $A$. In this new problem we are allowed to use the simplex algorithm. If the output of the algorithm is a point $\mathbf{x}$ that has only coordinates 0 or 1, then the problem is fully solved, because then this point is the answer to the binary problem that we stated above. However, it might be the case that we obtain a vector $\mathbf{x}$ with a lot of fractional entries, and even more dramatically, we cannot be sure that the maximum for this point would coincide with the maximum for the original binary problem. However, in some situations the region $\mathcal{P}$ has all of its vertices with $0/1$ coordinates, so that we can be sure that the simplex algorithm *will for sure* give us the right answer.

**Definition 6.2.** An *alcoved polytope* is a polytope obtained by using inequalities of the following form:

$$\alpha_{ij} \leq x_i + x_{i+1} + \ldots + x_j \leq \beta_{ij},$$

for $\alpha_{ij}, \beta_{ij}$ integer numbers and $1 \leq i \leq j \leq n$.

**Theorem 6.3** (Lam - Postnikov)**.** *The vertices of an alcoved polytope have only integer coordinates.*

**Example 6.4.** Consider the following

$$\begin{aligned}&\text{maximize } 10x_1 + 3x_2 - 2x_3 + 7x_4,\\ &\text{under } \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} \leq \begin{bmatrix}7\\ 2019\\ 30\end{bmatrix}\\ &\text{and } 0 \leq x_1, x_2, x_3, x_4 \leq 1 \text{ for all } i\end{aligned}$$

We claim that the convex set defined by all the inequalities is an alcoved polytope. Indeed, all the inequalities in the matrix appear to be in a *range* of variables, and involve only integer numbers. Hence, all the vertices of our polytope have integer coordinates. Since in particular the last inequality tells us that each coordinate lies in the interval between zero and one, they have to be *either* zero or one.

**Remark 6.5.** It may happen that to obtain a description of the problem by using inequalities as in the definition of alcoved polytope, we have to *permute accordingly* the names of the variables.

**Corollary 6.6.** *If the inequalities determined by a matrix $A$ define an alcoved polytope, then the binary linear programming problem can be solved using the simplex algorithm.*

**Example 6.7.** Problem D "Delight for a Cat" of NEERC 16 has an abridged description as follows:

$$\max \sum_{i=1}^{n} (s_i - e_i) x_i ,$$

under the constraints of the form:

$$x_i \in \{0, 1\} \text{ for all } i ,$$

and:

$$m_s \leq \sum_{i=j}^{j+k-1} x_i \leq k - m_e ,$$

for all $j = 1, \ldots, n - k + 1$.

Since these inequalities define an alcoved polytope, we can instead of $x_i \in \{0, 1\}$ assume $0 \leq x_i \leq 1$ and use the simplex algorithm which will solve this problem.

Università di Bologna, Dipartimento di Matematica, Piazza di Porta San Donato, 5, 40126 Bologna BO - Italia

*E-mail address*: luis.ferronirivetti2@unibo.it,