19

Computational and mathematical methods in portfolio insurance - A MATLAB-based approach

Vasilios N. Katsikis
General Department of Mathematics,
Technological Education Institute of Piraeus,
12244 Athens
Greece

1. Introduction

Portfolio insurance is based on the principal of risk transfer i.e., one person's protection is another person's liability. The cost of portfolio insurance is the mechanism to equilibrate its demand with supply. In the theory of finance minimum-cost portfolio insurance has been characterized as a very important investment strategy. In this chapter, we discuss the investment strategy called minimum-cost portfolio insurance as a solution of a cost minimization problem and we propose computational methods that translate the economics problem into the language of computing. This strategy not only enables an investor to avoid losses but also allows him/her to capture the gains at the minimum cost. In general, it is well known that the minimum-cost insured portfolio depends on security prices. The cases where it is price-independent (i.e., it does not depend on arbitrage-free security prices) are very important not only because the insured portfolio can be selected without knowledge of current security prices but also because we can present it in a simple form. Market structures in which minimum-cost portfolio insurance is price-independent relies on the theory of vector lattices (Riesz spaces). In particular, we focus our study in two very important classes of subspaces of a vector lattice, namely vector sublattices and lattice-subspaces. Vector lattices have been used by Brown & Ross (1991) and by Green & Jarrow (1987) in the framework of options markets. Also, Ross (1976) gave a characterization of complete markets by observing that derivative markets are complete if and only if the asset span is a vector sublattice of $\mathbb{R}^k$. Completeness of derivative markets is a sufficient but not necessary condition for the minimum-cost portfolio insurance to be price-independent. Let us denote by X the subspace of payoffs of all portfolios of securities; then in Aliprantis et al. (2000) it is proved that the minimum-cost insured portfolio exists and is price-independent for every portfolio and at every floor if and only if X is a lattice-subspace of $\mathbb{R}^k$. An equivalent necessary and sufficient condition so that X is a lattice-subspace is the existence of a positive basis for X, that is a basis of limited liability payoffs such that every marketed limited liability payoff has a unique representation as a nonnegative linear combination of basis payoffs. The notion of a positive basis for X is a generalization to incomplete markets of a basis of Arrow securities for complete markets. From the previous discussion, it is evident that the mathematical theory of lattice-subspaces has been used in order to provide a characterization of market structures in which the cost minimizing portfolio is price-independent. In general, the theory of lattice-subspaces has been extensively used in the last years in Mathematical Economics, especially in the areas of incomplete markets and portfolio insurance (e.g., Aliprantis et al. (1997; 2000; 2002); Polyrakis (2003)) as well as in completion of security markets (Kountzakis & Polyrakis (2006)). The study of finite-dimensional lattice-subspaces is important since many economic models are finite, such as, for example, the well-known Arrow-Debreu model. Additional applications of lattice-subspaces in economics appear in Aliprantis et al. (1998); Henrotte (1992). In this chapter, the main advantage of the computational techniques that we present is that we are able to solve the minimization problem without making use of any linear programming method. This is possible by using the theory of positive bases in vector lattices; specifically, we are able to provide a practical numerical way to check whether a subspace X is a lattice-subspace or a vector sublattice.

In Polyrakis (1996; 1999), lattice-subspaces and vector sublattices are studied in the space of continuous real valued functions $C(\Omega)$ defined on a compact Hausdorff topological space Ω. In the case where Ω is finite, for example $\Omega = \{1, 2, \ldots, k\}$, then $C(\Omega) = \mathbb{R}^k$ and the results of Polyrakis (1996; 1999) can be applied for the determination of the lattice-subspaces and vector sublattices of $\mathbb{R}^k$. In particular, in Polyrakis (1996) it is provided a solution to the problem of whether a finite collection of linearly independent positive functions in $C(\Omega)$ generates a lattice-subspace. In addition, he proposed an algorithm under which one can check whether the vector subspace[1] $X = [x_1, ..., x_n]$ is a lattice-subspace, where $x_1, ..., x_n$ are linearly independent positive functions in $C(\Omega)$. Another approach to the same problem of whether X forms a lattice-subspace of $\mathbb{R}^k$ is presented in Abramovich et al. (1994).

In Katsikis (2007), based on Abramovich et al. (1994), a computational solution is given to the problem of whether a finite collection of linearly independent, positive vectors of $\mathbb{R}^k$ generates a lattice-subspace. In addition, in Katsikis (2007), applications to the cost minimization problem that ensures the minimum-cost insured portfolio are discussed. The same reference concludes with a Matlab function which is an elegant and accurate tool in order to provide whether or not a given collection of vectors forms a lattice-subspace.

Also, in Katsikis (2008), a different computational method is presented based upon the Polyrakis algorithm (1996), in order to solve the corresponding problem in $C[a, b]$. This computational method implements a general algorithmic process and when slightly modified, this process can also be used in the case of lattice-subspaces of $\mathbb{R}^k$. Following this remark, Katsikis (2009) presents the translation followed by the implementation of this algorithm in $\mathbb{R}^k$ within a Matlab function. This function provides an important tool in order to investigate lattice-subspaces and vector sublattices of $\mathbb{R}^k$ with direct applications to portfolio insurance. Finally, the results of Katsikis (2009) can be applied in completion of security markets and the theory of efficient funds.

The material in this chapter is spread out in 8 sections. Section 2 gives the fundamental properties of lattice-subspaces and vector sublattices of $\mathbb{R}^k$ together with the solution to the problem of whether a finite collection of linearly independent, positive vectors of $\mathbb{R}^k$ generates a lattice-subspace or a vector sublattice. Section 3 studies, in detail, from the computational point of view the mathematical problem stated in Section 2 and presents an efficient computational method in order to solve it. Comparison results with other existing computational methods are also provided. Section 4 studies finite dimensional lattice-subspaces of $C[a, b]$ and presents the solution to the problem stated in Section 2, in the case where the initial space is $C[a, b]$. Section 5 presents computational methods in order to determine finite dimensional lattice-subspaces of $C[a, b]$. Section 6 provides the most important interrelationship between lattice-subspaces and the minimization problem of minimum-cost portfolio insurance. Also,

[1] $[x_1, ..., x_n]$ denotes the n-dimensional vector subspace generated by $x_1, ..., x_n$.

the reader will find in this Section a study that involves the computational techniques presented in Section 3 and Section 5 in order to calculate the minimum-cost insured portfolio both in the case of $\mathbb{R}^k$ and $C[a,b]$. Section 7 provides a computational technique, based on Section 3, in order to solve the problem of completion by options of a two-period security market in which the space of marketed securities is a subspace of $\mathbb{R}^k$. Methods on computing the efficient funds of the market are also presented. Conclusions and research directions are provided in Section 8.
In this chapter, all the numerical tasks have been performed using the Matlab 7.8 (R2009a) environment on an Intel(R) Pentium(R) Dual CPU T2310 @ 1.46 GHz 1.47 GHz 32-bit system with 2 GB of RAM memory running on the Windows Vista Home Premium Operating System.

2. Lattice-subspaces and vector sublattices of $\mathbb{R}^k$

In this section, a brief introduction is provided to the theory of lattice-subspaces and vector sublattices of $\mathbb{R}^k$. In addition, we present the solution to the problem of whether a finite collection of linearly independent, positive vectors of $\mathbb{R}^k$ generates a lattice-subspace or a vector sublattice.

2.1 Preliminaries and notation

We view $\mathbb{R}^k$ as an ordered space, then the *pointwise order* relation in $\mathbb{R}^k$ is defined by

$$x \leq y \textit{ if and only if } x(i) \leq y(i), \textit{ for each } i = 1, ..., k.$$

The positive cone of $\mathbb{R}^k$ is defined by $\mathbb{R}^k_+ = \{x \in \mathbb{R}^k | x(i) \geq 0, \textit{ for each } i\}$ and if we suppose that X is a vector subspace of $\mathbb{R}^k$ then X ordered by the pointwise ordering is an *ordered subspace* of $\mathbb{R}^k$ with positive cone X_+ defined by $X_+ = X \cap \mathbb{R}^k_+$. For a two-point set $S = \{x, y\}$, we denote by $x \vee y$ $(x \wedge y)$ the *supremum* of S i.e., its least upper bound (the *infimum* of S i.e., its greatest lower bound). Thus, $x \vee y$ $(x \wedge y)$ is the componentwise maximum (minimum) of x and y defined by

$$(x \vee y)(i) = \max\{x(i), y(i)\} \; ((x \wedge y)(i) = \min\{x(i), y(i)\}), \; \textit{for all } i = 1, ..., k.$$

An ordered subspace X of $\mathbb{R}^k$ is a *lattice-subspace* of $\mathbb{R}^k$ if it is a vector lattice in the induced ordering, i.e., for any two vectors $x, y \in X$ the supremum and the infimum of $\{x, y\}$ both exist in X. Note that the supremum and the infimum of the set $\{x, y\}$ are, in general, different in the subspace than the supremum and the infimum of this set in the initial space. An ordered subspace Z of $\mathbb{R}^k$ is a *vector sublattice* or a *Riesz subspace* of $\mathbb{R}^k$ if for any $x, y \in Z$ the supremum and the infimum of the set $\{x, y\}$ in $\mathbb{R}^k$ belong to Z. Suppose that X is an ordered subspace of $\mathbb{R}^k$ and $B = \{b_1, b_2, ..., b_m\}$ is a basis for X. Then B is a *positive basis* of X if the positive cone X_+ of X has the form,

$$X_+ = \{x = \sum_{i=1}^{m} \lambda_i b_i | \lambda_i \geq 0, \textit{ for each } i\}.$$

Therefore, if $x = \sum_{i=1}^{m} \lambda_i b_i$ and $y = \sum_{i=1}^{m} \mu_i b_i$ then $x \leq y$ if and only if $\lambda_i \leq \mu_i$ for each $i = 1, 2, ..., m$. The existence of positive bases is not always ensured, but in the case where X is a vector sublattice of $\mathbb{R}^k$ then X has always a positive basis. Moreover, it holds that an ordered subspace of $\mathbb{R}^k$ has a positive basis if and only if it is a lattice-subspace of $\mathbb{R}^k$. If $B = \{b_1, b_2, ..., b_m\}$ is a positive basis for a lattice-subspace (or a vector sublattice) X then the lattice operations in X, namely $x \triangledown y$ for the supremum and $x \vartriangle y$ for the infimum of the set $\{x, y\}$ in X, are given by

$$x \triangledown y = \sum_{i=1}^{m} \max\{\lambda_i, \mu_i\} b_i \textit{ and } x \vartriangle y = \sum_{i=1}^{m} \min\{\lambda_i, \mu_i\} b_i,$$

for each $x = \sum_{i=1}^{m} \lambda_i b_i, y = \sum_{i=1}^{m} \mu_i b_i \in X$. A vector sublattice is always a lattice-subspace, but the converse is not true as shown in the next example.

Example 0.1. Let $X = [x_1, x_2, x_3]$ be the subspace of $\mathbb{R}^4$ generated by the vectors $x_1 = (6,0,0,1), x_2 = (6,4,0,0), x_3 = (8,4,2,0)$. An easy argument shows that the set $B = \{b_1, b_2, b_3\}$ where

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 2 & 0 \\ 12 & 8 & 0 & 0 \\ 6 & 0 & 0 & 1 \end{bmatrix}$$

forms a positive basis of X therefore X is a lattice-subspace of $\mathbb{R}^4$. On the other hand, let us consider the vectors $y_1 = 2x_1 + x_2 = (18,4,0,2)$ and $y_2 = x_3 - x_2 = (2,0,2,0)$ of X. Then, $y_1 \vee y_2 = (18,4,2,2)$ and since $y_1 = \frac{1}{2}b_2 + 2b_3, y_2 = b_1$, it follows that $y_1 \triangledown y_2 = b_1 + \frac{1}{2}b_2 + 2b_3 = (20,4,2,2)$. Therefore, X is not a vector sublattice of $\mathbb{R}^4$, since the supremum on the subspace X is different than the supremum on the whole space.

For an extensive presentation of lattice-subspaces, vector sublattices and positive bases the reader may refer to Abramovich et al. (1994); Polyrakis (1996; 1999).

2.2 The mathematical problem

Suppose that $\{x_1, x_2, ..., x_n\}$ is a collection of linearly independent, positive vectors of $\mathbb{R}^k$. The problem is, under what conditions the subspace $X = [x_1, x_2, ..., x_n]$ is a lattice-subspace or a vector sublattice of $\mathbb{R}^k$?

Let us denote by β the *basic function* of $x_1, x_2, ..., x_n$, that is, $\beta : \{1, 2, ..., k\} \to \mathbb{R}^k$ such that

$$\beta(i) = \Big(\frac{x_1(i)}{z(i)}, \frac{x_2(i)}{z(i)}, ..., \frac{x_n(i)}{z(i)} \Big),$$

for each $i \in \{1, 2, ..., k\}$ with $z(i) > 0$, where $z = \sum_{i=1}^{n} x_i$. The set

$$R(\beta) = \{\beta(i) | i = 1, 2, ..., k, \textit{ with } z(i) > 0\},$$

is the *range* of the basic function and the *cardinal number*, *card*$R(\beta)$, of $R(\beta)$ is the number of different elements of $R(\beta)$. Let *card*$R(\beta) = m$ then it is clear that $n \leq m \leq k$. Denote by K the convex hull of $R(\beta)$. Since K is the convex hull of a finite subset of $\mathbb{R}^k$ it is a polytope with d vertices and each vertex of K belongs to $R(\beta)$ therefore $n \leq d \leq m$.

Suppose that $R(\beta) = \{P_1, P_2, ..., P_m\}$ such that, under a proper enumeration, the vertices $P_1, P_2, ..., P_n$ are linearly independent and $P_1, P_2, ..., P_d$ are the vertices of K, i.e.,

$$R(\beta) = \{ \overbrace{\underbrace{P_1, P_2, ...P_n}_{\textit{linearly independent}}, P_{n+1}, ...P_d}^{\textit{vertices of } K}, ..., P_m \}.$$

The following theorem, from (Polyrakis, 1999), provides a full answer to the stated problem.

Theorem 0.1. *Suppose that the above assumptions are satisfied. Then,*

(i) *X is a vector sublattice of $\mathbb{R}^k$ if and only if $R(\beta)$ has exactly n points (i.e., $m = n$) and a positive basis $\{b_1, b_2, ..., b_n\}$ for X is defined by the formula*

$$(b_1, b_2, ..., b_n)^T = A^{-1}(x_1, x_2, ..., x_n)^T,$$

where A is the $n \times n$ matrix whose ith column is the vector P_i, for each $i = 1, 2, ..., m$.

(ii) *X is a lattice-subspace of $\mathbb{R}^k$ if and only if the polytope K has n vertices (i.e., $d = n$) and a positive basis $\{b_1, b_2, ..., b_n\}$ for X is defined by the formula*

$$(b_1, b_2, ..., b_n)^T = A^{-1}(x_1, x_2, ..., x_n)^T,$$

where A is the $n \times n$ matrix whose ith column is the vector P_i, for each $i = 1, 2, ..., d$.

2.3 The algorithm

The basic steps of an algorithmic process that will accurately implement the ideas of Theorem 0.1 are the following:

(1) Determine $R(\beta)$.

(2) Compute the number $m = cardR(\beta)$, and the number d of vertices of the polytope K.

(3) If $n = m$ (vector sublattice case) or $n = d$ (lattice-subspace case) then, determine a positive basis for X.

Based on a theorem of Edmonds, Lovász and Pulleybank in Edmond et al. (1982), we close this section with some remarks on the existence of a polynomial-time decision procedure, in order to decide whether the collection of vectors $\{x_1, x_2, ..., x_n\}$ generates a lattice-subspace or a vector sublattice. We shall present this result, in a suitable form for our analysis, as it is presented in Aliprantis et al. (1997).

Theorem 0.2. *There exists a polynomial-time algorithm that for any polytope, P, defined as the convex hull of a given finite set of vectors, determines the affine hull of P. Specifically the algorithm finds affinely independent vertices $u_0, u_1, ..., u_\ell$ of P such that*

$$\mathit{aff}(P) = \mathit{aff}(\{u_0, u_1, ..., u_\ell\}).$$

Recall that, algorithms which have a polynomial or sub-polynomial time complexity (that is, they take time $O(g(n))$ where $g(n)$ is either a polynomial or a function bounded by a polynomial), are practical. Such algorithms with running times of orders $O(\log n), O(n)$, $O(n \log n), O(n^2), O(n^3)$ etc. are called polynomial-time algorithms. There are several arguments to support the thesis that "polynomial" is a synonym to practical and the general conclusion is that a problem can be considered "efficiently solved" when a polynomial-time algorithm has been found for it.
In order to implement algorithm 2.3, we shall use the Quickhull algorithm from Barber et al. (1996) for computing the convex hull of a given set of points. According to Barber et al. (1996) (Theorem 3.2) if d is the dimension, n is the number of input points, r the number of processed points, and f_r the maximum number of facets of r vertices ($f_r = O(r^{\lfloor \frac{d}{2} \rfloor} / \lfloor \frac{d}{2} \rfloor !)$ then the worst-case complexity of Quickhull is $O(n \log r)$ for $d \leq 3$ and $O(n f_r / r)$ for $d \geq 4$.

3. The computational method

3.1 Method presentation

In this section we present the translation followed by the implementation of algorithm 2.3 within a Matlab function named `SUBlatSUB` from Katsikis (2009). This function provides an important tool in order to investigate lattice-subspaces and vector sublattices of $\mathbb{R}^k$ since we are able to perform fast testing for a variety of dimensions and subspaces. Recall that, the numbers n, m, d, k denote the dimension of X, the cardinality of $R(\beta)$, the number of vertices of the convex hull of $R(\beta)$ and the dimension of the initial Euclidean space, respectively.

The function `SUBlatSUB` first checks if the given collection of vectors generates a vector sublattice by examining the validity of condition (i) of Theorem 0.1. In the case of a vector sublattice, i.e., $m = n$, the program responds with the output:

```
vector sublattice
```

followed by a $n \times k$ matrix whose rows are the vectors of the positive basis.

If, instead, the collection does not generate a vector sublattice, that is $m \neq n$, then the function `SUBlatSUB` checks if the given collection generates a lattice-subspace by examining the validity of condition (ii) of Theorem 0.1. In the case of a lattice-subspace, i.e., $d = n$, the program responds with the output:

```
lattice-subspace
```

followed by a $n \times k$ matrix whose rows are the vectors of the positive basis.

If $m \neq n$ and $d \neq n$ then the program responds with the output:

```
not a lattice-subspace
ans=
     []
```

So, in order to decide whether a given collection of linearly independent, positive vectors generates a lattice-subspace or a vector sublattice of $\mathbb{R}^k$, we construct a matrix whose columns are the vectors of the given collection and then we apply the function `SUBlatSUB` on that matrix. It is possible to produce the numbers n, m, d, k, with this order, as a 4×1 matrix with the following code,

```
>> [positivebasis,dimensions]=SUBlatSUB(a)
```

where a is the matrix whose columns are the given vectors.

3.2 Numerical examples

In order to describe the most important features of `SUBlatSUB`, we illustrate some examples featured in Katsikis (2007) for various collections and dimensions. Also, we close this section with comparison results of the `SUBlatSUB` function and the alternative function, namely **K** function, presented in Katsikis (2007).

Example 0.2. Consider the following 7 vectors $x_1, x_2, ..., x_7$ in $\mathbb{R}^{10}$,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

According to the definition of the β function, the rows of the following matrix u are the different elements of $R(\beta)$,

```
u =
    0   0   0   0   0   0    1
    0   0   0   0   0   0.5 0.5
    0   0   0   0   1   0   0
    0   0   0   1   0   0   0
    0   0   1   0   0   0   0
    0   0.5 0   0   0   0   0.5
    0   1   0   0   0   0   0
    0.5 0   0   0   0.5 0   0
```

Thus, $m = 8$ and it is clear that rows $u(1), \ldots, u(5), u(7), u(8)$ of `u` are linearly independent. This means that these vectors belong to the convex hull of $R(\beta)$. Also, it is easy to see that row $u(6)$ of `u` is a convex combination of the other rows. Therefore, $d = 7$ and $X = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$ is a lattice-subspace.
For a numerical solution, we invoke the `SUBlatSUB` function by typing in the command window of the Matlab environment:

```
>> [positive basis,dimensions]=SUBlatSUB(a)
```

The results, then, are as follows:

```
lattice-subspace
positivebasis =
    0     1     0     0     0     0     0     0     0     1
    0     0     0     0     0     0     2     0     0     0
    0     0     0     0     1     1     0     0     1     0
    0     0     0     1     0     0     0     0     0     0
    0     0     1     0     0     0     0     0     0     0
    0     1     0     0     0     0     0     1     0     0
    2     0     0     0     0     0     0     0     0     0
dimensions =
            7
            8
            7
            10
```

We conclude with some comments based on the results of the `SUBlatSUB` function. A positive basis is unique in the sense of positive multiples since each element of the basis is an extremal[2] point of the positive cone of the subspace. If we denote by $\{b_1, b_2, ..., b_7\}$ the positive basis that we obtained by using the **K** function (see Katsikis (2007)) and by $\{B_1, B_2, ..., B_7\}$ the positive basis we found with the `SUBlatSUB` function then it holds

$$(B_1, B_2, B_3, B_4, B_5, B_6, B_7) = (b_7, 2b_5, b_4, b_3, b_2, b_6, 2b_1).$$

Example 0.3. Consider the following 7 vectors $x_1, x_2, ..., x_7$ in $\mathbb{R}^{10}$,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 4 & 3 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 & 3 & 1 & 3 & 4 & 4 \\ 3 & 3 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 \end{bmatrix}$$

where following the same procedure, as before, one gets

```
vector sublattice
positivebasis =
   0    0    0    0    0    0    12    0    0    0
   0    0    0    0    3    0    0     0    0    0
   0    0    0    0    0    4    0     4    0    0
   0    0    0    0    0    0    0     0    12   12
   6    6    0    0    0    0    0     0    0    0
   0    0    0    5    0    0    0     0    0    0
   0    0    6    0    0    0    0     0    0    0
dimensions =
                  7
                  7
                  10
```

If we denote by $\{b_1, b_2, ..., b_7\}$ the positive basis that we obtained by using the **K** function and by $\{B_1, B_2, ..., B_7\}$ the positive basis we found with the `SUBlatSUB` function then it holds

$$(B_1, B_2, B_3, B_4, B_5, B_6, B_7) = (12b_6, 3b_4, 4b_5, 12b_7, 6b_1, 5b_3, 6b_2).$$

Example 0.4. Consider the following 5 vectors $x_1, x_2, ..., x_5$ in $\mathbb{R}^{10}$,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 & 1 & 1 & 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 2 & 1 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

For the above set, the program yields

[2] A nonzero element x_0 of X_+ is an *extremal point* of X_+ if, for any $x \in X, 0 \leq x \leq x_0$ implies $x = \lambda x_0$ for a real number λ.

```
lattice-subspace
positivebasis =
  0     0     0     0     0     0     0     8     0     0
  0     7/4   0     7     7/4   7/4   7/4   0     0     0
  6     3/2   6     0     3/2   3/2   3/2   0     0     0
  0     0     0     0     0     0     0     0     0     6
  0     7/4   0     0     7/4   7/4   7/4   0     7     0
dimensions =
            5
            6
            5
            10
```

In this case it holds

$$(B_1, B_2, B_3, B_4, B_5) = (8b_3, 7b_2, 6b_1, 6b_5, 7b_4).$$

For the purpose of monitoring the performance, we present in Table 1 the execution times of the `SUBlatSUB` function and the method presented in Katsikis (2007) (**K** function).

Matlab functions	Example 0.2	Example 0.3	Example 0.4
`SUBlatSUB`	0.052	0.030	0.035
K	0.516	0.828	0.969

Table 1. Time in seconds

3.3 The case of coplanar points

The correct performance of the `SUBlatSUB` function requires the use of the **convhulln** Matlab function which is based on Qhull[3] and Qhull implements the Quickhull algorithm (Barber et al. (1996)) for computing the convex hull of a given set of points. Suppose that a denotes the matrix whose rows are the coefficients of the given points, then the **convhulln** function returns the indices of the points in a that comprise the facets of the convex hull of a. The **convhulln** function is facing problems during the calculation of the convex hull of points that lie in a q-manifold, with $q \leq n-1$, in the n-dimensional space of the given data. So, we cannot use **convhulln** to solve our problem directly.

We illustrate the details in this case through the following example and we also provide an improvement technique to solve the resultant problem in this particular case.

Example 0.5. Consider the following four vectors x_1, x_2, x_3, x_4 in $\mathbb{R}^7$,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 & 0 & 1 & 1 & 4 \\ 0 & 1 & 1 & 1 & 1 & 0 & 2 \\ 2 & 1 & 0 & 1 & 1 & 1 & 2 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Following the second step of algorithm 2.3, the calculation of the convex hull of $R(\beta)$ is required to check whether $X = [x_1, x_2, x_3, x_4]$ is a lattice-subspace or a vector sublattice of $\mathbb{R}^7$.

[3] For information about Qhull see http://www.qhull.org/

In this case, and after the necessary calculations it is clear that $R(\beta) = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ where

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}.$$

Therefore, in order to determine the convex hull of $R(\beta)$ we have used the **convhulln** function. In this case the **convhulln** function yields with the following warning message:

```
??? qhull precision warning: The initial hull is narrow (cosine of
min. angle is 1.0000000000000002). A coplanar point may lead to a
wide facet.
```

If we use the following simple rank test:

```
>> y=[0 1/3 1/3 1/3;1/4 0 1/2 1/4;1/2 0 1/2 0;1/3 1/3 0 1/3;

1/2 1/4 1/4 0;1/4 1/4 1/4 1/4];
>> rank(bsxfun(@minus,y,y(6,:)))
```

then one gets

```
 ans =
     3
```

Thus, by our previous analysis it is clear that the points $P_1, P_2, P_3, P_4, P_5, P_6$ of $\mathbb{R}^4$ lie in a 3-manifold and we cannot use **convhulln** to solve our problem directly. A solution to this problem can be given under the following methodology:

- Translate the points to a hyper-plane that passes through the origin.
- Determine a set of basis vectors for the subspace.
- Transform the points into an equivalent lower dimensional space.
- Form the convex hull triangulation in the lower dimensional space.

Let us describe, in detail, the procedure for this particular example. First, one has to translate the given points to a hyper-plane that passes through the origin by subtracting one of the vectors from the others,

```
>> ytrans = bsxfun(@minus,y,y(6,:))
ytrans =
  -0.2500    0.0833    0.0833    0.0833
        0   -0.2500    0.2500         0
   0.2500   -0.2500    0.2500   -0.2500
   0.0833    0.0833   -0.2500    0.0833
   0.2500         0         0   -0.2500
        0         0         0         0
```

Then, we form an orthonormal basis for the range of `ytrans`.

```
>> rot = orth(ytrans')
rot =
   0.5  -0.64505   0.28967
   0.5  -0.28967  -0.64505
  -0.5   0.64505  -0.28967
   0.5   0.28967   0.64505
```

Recall that, if $\{v_1, v_2, \ldots, v_r\}$ is an orthonormal basis for a finite dimensional subspace W of an inner product space V and u is any vector of V, then the projection of the vector u in W is given by the formula $proj_W u = < u, v_1 > v_1 + \ldots + < u, v_r > v_r$.
Now, we project the points into an equivalent lower dimensional space where `rot` is a basis for this space. Hence,

```
>> yproj = ytrans*rot
yproj =
   0.16667  0.21502   -0.096557
  -0.25      0.23368    0.088845
  -0.5       3.0531e-16 5.5511e-17
   0.16667 -0.21502    0.096557
  -0.25    -0.23368   -0.088845
   0        0          0
```

Note that, the rows of `yproj` matrix are the coordinates of the initial points in terms of the basis in the lower dimensional space.
Finally, we form the convex hull triangulation in the projected subspace `yproj`. That is,

```
>> tri = convhulln(yproj)
tri =
     1  2  3
     2  4  3
     4  2  1
     5  1  3
     4  5  3
     5  4  1
```

Since we are only interested for the number of vertices of the convex hull of $R(\beta)$, we can only determine the number of vertices of the convex hull in the projected subspace. Therefore,

```
>> length(unique(tri(:)))
ans =
     5
```

So, there are 5 vertices in the hull. This procedure is included in the `SUBlatSUB` function therefore, for a direct answer in the previous example, one can apply the `SUBlatSUB` function directly to the given collection by using the code,

```
>> [positive basis,dimensions]=SUBlatSUB(a)
```

the results, then, are as follows:

```
not a lattice-subspace
positivebasis =
                 []
dimensions =
              4
              6
              5
              7
```

where a denotes a matrix that has the vectors $x_i, i = 1, 2, 3, 4$ as columns.

3.4 Comparison results

In this section, we compare the performance of the `SUBlatSUB` function to that of the **K** function. The numerical method, based on the introduction of the `SUBlatSUB` function, enables us to perform fast and accurate estimations of the lattice-subspace or the vector sublattice for a finite collection of positive, linearly independent vectors of $\mathbb{R}^k$ for a variety of dimensions. For this purpose we have used the Matlab function **rand** in order to produce 50 full rank matrices for each rank n, $n = 3, ..., 30$. The cumulative results are presented in Figure 1 (Figure 1 shows the time efficiency curves, i.e., the rank of the 50 tested matrices versus the total computation time (in seconds)) and in Table 2. From the previous results (see Figure 1,Table 2) it is evident

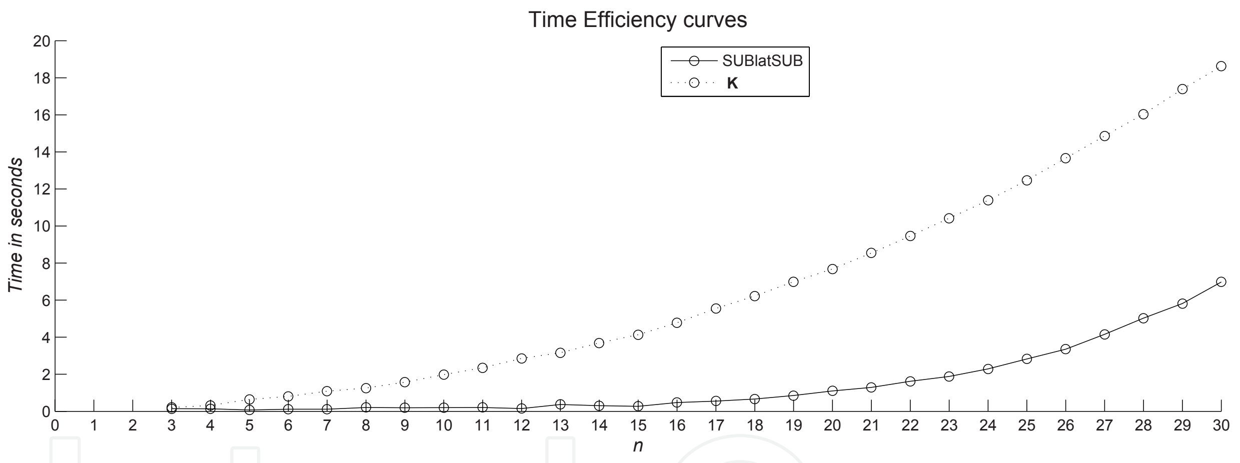

Fig. 1. Time efficiency curves for the **K** function and the SUBlatSUB function

that using the `SUBlatSUB` function the interested user can reach a fast computational solution using a reduced amount of computational resources.

4. Finite dimensional lattice-subspaces of $C[a, b]$

In what follows we shall denote by $C[a, b]$ the space of all continuous real functions defined on the interval $[a, b]$. As in the case of $\mathbb{R}^k$, lattice-subspaces of $C[a, b]$ are subspaces which are vector lattices in the induced ordering, i.e., for any two vectors x, y of the subspace the supremum and the infimum of the set $\{x, y\}$ both exist in the subspace. Recall that the supremum and the infimum of the set $\{x, y\}$ are, in general, different in the subspace than the supremum and the infimum of this set in the initial space. In this section we present a brief introduction to the theory of lattice-subspaces in $C[a, b]$. In addition, we describe in detail the construction of a powerful and efficient package for the translation, into the language of computing, of the

Rank	`SUBlatSUB` (Time in seconds)	**K** (Time in seconds)	Rank	`SUBlatSUB` (Time in seconds)	**K** (Time in seconds)
3	0.141	0.205	17	0.553	5.548
4	0.140	0.322	18	0.665	6.230
5	0.073	0.641	19	0.853	6.992
6	0.115	0.805	20	1.100	7.681
7	0.113	1.090	21	1.290	8.545
8	0.207	1.257	22	1.620	9.457
9	0.191	1.572	23	1.882	10.420
10	0.198	1.980	24	2.292	11.382
11	0.209	2.358	25	2.823	12.470
12	0.153	2.858	26	3.353	13.662
13	0.371	3.161	27	4.154	14.854
14	0.308	3.694	28	5.030	16.040
15	0.272	4.135	29	5.815	17.384
16	0.477	4.781	30	6.978	18.617

Table 2. Results for 50 tested full rank matrices for each rank n, $n = 3, ..., 30$.

mathematical problem of whether the vector subspace $X = [x_1, ..., x_n]$ is a lattice-subspace of $C[a,b]$, where $x_1, ..., x_n$ are linearly independent positive functions in $C[a,b]$.

4.1 Preliminaries and notation

Let $C[a,b]_+$ be the positive cone of $C[a,b]$ and assume that X is a subspace of $C[a,b]$. The *induced ordering* on X is the ordering defined by $X_+ = X \cap C[a,b]_+$ (induced cone of X). An *ordered subspace* of $C[a,b]$ is a subspace of $C[a,b]$ under the induced ordering. A *lattice-subspace* of $C[a,b]$ is an ordered subspace X of $C[a,b]$ which is a vector lattice in its own, that is, for each $x, y \in X$ the supremum and the infimum of the set $\{x, y\}$ exists in X. If X is a lattice-subspace of $C[a,b]$ then we will denote by $x \nabla y$ the supremum of the set $\{x, y\}$ in X. Similarly, $x \vartriangle y$ stands for the infimum of the set $\{x, y\}$ in X. If $x \vee y$ denotes the supremum and $x \wedge y$ the infimum in E of the set $\{x, y\}$ and we suppose that $x \vartriangle y, x \wedge y, x \vee y, x \nabla y$ exists, then it follows that

$$x \vartriangle y \leq x \wedge y \leq x \vee y \leq x \nabla y \tag{1}$$

For example, consider $C[0,1]$, the space of all continuous real functions in the interval $[0,1]$ and $X = \{ax + b | a, b \in \mathbb{R}\}$. Then X is a lattice-subspace of $C[0,1]$ and (1) holds for each $x, y \in X$ (Figure 2).

One of the most serious difficulties in the study of lattice-subspaces comes from the fact that the supremum and the infimum depend both on the subspace.

For a general definition of a positive basis, let E be a (partially) ordered Banach space. Then a sequence $\{e_n\}$ of positive vectors of E is a *positive basis* if it is a Schauder basis of E and

$$E_+ = \{x = \sum_{i=1}^{\infty} \lambda_i e_i \in E | \lambda_n \geq 0, \textit{ for all } n \in \mathbb{N}\}.$$

Equivalently, one can say that $\{e_n\}$ is a positive basis of E if

$$x = \sum_{i=1}^{\infty} \lambda_i e_i \geq 0 \Leftrightarrow \lambda_n \geq 0, \textit{ for all } n \in \mathbb{N}.$$

Let Y be a closed subspace of $E = C[a,b]$ with basis $\{b_n\}$ (not necessarily positive). Fix $t \in [a,b]$ and $m \in \mathbb{N}$. Following the terminology introduced in Polyrakis (1996), if $b_m(t) \neq 0$ and

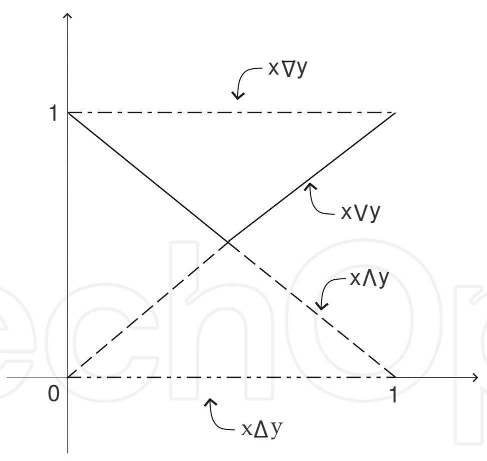

Fig. 2. Relation (1) for x=t, y=1-t,t∈[0,1].

$b_n(t) = 0$ for each $n \neq m$, then we shall say that the point t is an *m-node (or simply a node)* of the basis $\{b_n\}$. If for each n there exists an n-node t_n of the basis $\{b_n\}$, then we shall say that $\{b_n\}$ is *a basis of Y with nodes* and that t_n is *a sequence with nodes* of $\{b_n\}$. If $\dim Y = n$ and for each $m \in \{1, 2, ..., n\}$ there exists an m-node t_m of the basis of Y, then we shall say that $\{b_1, b_2, ..., b_n\}$ is a basis of Y with nodes and that the points $t_1, t_2, ..., t_n$ are nodes of the basis $\{b_1, b_2, ..., b_n\}$. The *support* of a function $x \in C[a, b]$ is the closure of the set $\{t \in [a, b] : x(t) > 0\}$ and shall be denoted by $suppx$.

4.2 The mathematical problem

In this section, we present the method developed in Polyrakis (1996), for the determination of the finite-dimensional lattice-subspaces of $C[a, b]$ and we shall discuss the necessary and sufficient conditions for a collection of linearly independent, positive functions, $x_1, x_2, ..., x_n$ of $C[a, b]$ to generate a lattice-subspace. Recall that the Wronski determinant of the functions x_i, $i = 1, ..., n$ is the $n \times n$ determinant which ith row is constituted of the $(i-1)$th derivatives of the functions x_i. Our first approach to the problem is given through the following Wronskian criterion:

Theorem 0.3. *Consider the closed interval $[a, b]$ of $\mathbb{R}$ and $\dim X > 2$, where $X = [x_1, x_2, ..., x_n]$. Suppose that (c, d) is an open interval of $\mathbb{R}$ which contains $[a, b]$. If the functions x_i have continuous derivatives up to the nth order in (c, d) and the Wronskian of the functions x_i is nonzero for any point of (c, d), then X is not a lattice-subspace of $C[a, b]$.*

As in the case of $\mathbb{R}^k$, let $x_1, ..., x_n$ in $C[a, b]$, we shall denote by z the sum $z = \sum_{i=1}^{n} x_i$ and by β the function $\beta : [a, b] \to \mathbb{R}^n$ such that

$$\beta(t) = \Big(\frac{x_1(t)}{z(t)}, \frac{x_2(t)}{z(t)}, ..., \frac{x_n(t)}{z(t)}\Big)$$

for each $t \in [a, b]$ with $z(t) > 0$. We shall refer to β as the *basic curve* of the functions $x_1, x_2, ..., x_n$. Also, we shall denote by $D(\beta)$ the domain and by $R(\beta)$ the range of the basic curve β of $x_1, x_2, ..., x_n$. If K is a subset of $\mathbb{R}^n$ then we shall denote by $\overline{K}$ the closure of K, by $int(K)$ the interior of K and by ∂K the boundary of K. We shall denote by $co(K)$ the convex hull of K and by $\overline{co}(K)$ the closure of $co(K)$.

The following theorem, is a criterion for lattice-subspaces and provides a full answer to the problem of whether a collection of positive functions $x_1, x_2, ..., x_n$ of $C[a,b]$ generates a lattice-subspace. In addition, if we are in the case of a lattice-subspace then the theorem determines a positive basis for $X = [x_1, x_2, ..., x_n]$.

Theorem 0.4. *The following statements are equivalent,*

(i) *X is a lattice-subspace of* $C[a,b]$.

(ii) *There exist n linearly independent* $P_1, P_2, ..., P_n$ *vectors in* $\mathbb{R}^n$, *belonging to the closure of the range of* β *such that for each* $t \in D(\beta)$ *the vector* $\beta(t)$ *is a convex combination of the vectors* $P_1, P_2, ..., P_n$, *i.e.,* $R(\beta) \subseteq co(\{P_1, P_2, ..., P_n\})$.
If (ii) *is true,* $P_i = \lim_{\nu\to\infty} \beta(\omega_{i\nu})$ *for each i, A is the* $n \times n$ *matrix whose ith column is the vector* P_i *and* $b_1, b_2, ..., b_n$ *are the functions defined by the formula*

$$(b_1(t), b_2(t), ..., b_n(t)) = A^{-1}(x_1(t), x_2(t), ..., x_n(t))^T,$$

then X has the following properties:

(a) *The set* $\{b_1, b_2, ..., b_n\}$ *is a positive basis of X. In addition, if* t_i *is a limit point of the sequence* $\{\omega_{i\nu} : \nu = 1, 2, ...\}$, *then* $t_i \in suppb_i$ *and* $b_k(t_i) = 0$, *for each* $k \neq i$.

(b) *The closed convex hull of* $R(\beta)$ *and the convex polygon with vertices the points* $P_1, P_2, ..., P_n$ *coincide.*

(c) *If* $P_k = \beta(t_k)$, *then* t_k *is a k-node of the basis* $\{b_1, b_2, ..., b_n\}$.

(d) *If* $P_k = \beta(t_k)$ *for some interior point* t_k *of* $[a,b]$ *and* x_i *are* C^2- *functions in a neighborhood of* t_k, *then*

$$\beta'(t_k) = \mathbf{0}.$$

The set $E(\beta)$ is the *extreme subset* of the basic curve β if there exists a subset G of $\overline{R(\beta)}$ consisting of n linearly independent vectors such that $R(\beta) \subseteq co(G)$, then we put $E(\beta) = G$, otherwise we put $E(\beta) = \varnothing$.
From Theorem 0.4 and the preceding definition the following proposition should be immediate.

Proposition 0.1. *The subspace X satisfies the properties*

(i) *X is a lattice-subspace if and only if* $E(\beta) \neq \varnothing$.

(ii) *If* $\beta(t) \in E(\beta)$, *then t is a node of the positive basis of X.*

(iii) *X has a positive basis with nodes if and only if* $E(\beta)$ *is a nonempty subset of* $R(\beta)$.

From Theorem 0.4 it is evident that if $P \in E(\beta)$ and $P \notin R(\beta)$, then we have that $P = \lim_{\nu\to\infty} \beta(t_\nu)$, where t_ν is a sequence of $D(\beta)$ having all limit points in the boundary $\partial D(\beta)$ of $D(\beta)$.
So, the *limit set* $L(\beta)$ of the curve β is defined as follows:

$$L(\beta) = \{P \in \mathbb{R}^n : \exists\{t_\nu\} \subseteq D(\beta) \textit{ with its limit points in } \partial D(\beta), P = \lim_{\nu\to\infty} \beta(t_\nu)\}.$$

Also, if $a, b \in D(\beta)$ then we shall denote by $\beta(\partial[a,b])$ the set

$$\beta(\partial[a,b]) = \{\beta(a), \beta(b)\}.$$

If t is an interior point of $[a, b]$ and $\beta(t) \in E(\beta)$, then t is a root of the equation

$$\beta'(t) = \mathbf{0}, \tag{2}$$

and we shall denote by $I(\beta)$ the images of the roots of equation (2), i.e.,

$$I(\beta) = \{\beta(t) : t \in Int([a, b]) \cap D(\beta) \text{ and } t \text{ is root of the equation } (2)\}.$$

Any subset of $L(\beta) \cup I(\beta) \cup \beta(\partial[a, b])$ consisting of n linearly independent vectors will be called a *possible extreme subset* of β.

Proposition 0.2. *If the functions $x_1, x_2, ..., x_n$ are C^2-functions in the set $Int([a, b]) \cap D(\beta)$, then $E(\beta) \subseteq L(\beta) \cup I(\beta) \cup \beta(\partial[a, b])$.*

The set $\beta(\partial[a, b])$ is known because $\partial[a, b] = \{a, b\}$ and if $D(\beta) = [a, b]$ then $L(\beta) = \varnothing$. In addition, if we assume that the domain of β has the form

$$D(\beta) = [a, t_1) \cup (t_1, t_2) \cup ... \cup (t_{n-1}, t_n) \cup (t_n, b]$$

and the limits

$$P_i = \lim_{t \to t_i} \beta(t)$$

exist for each i, then

$$L(\beta) = \{P_1, P_2, ..., P_n\}.$$

In view of Proposition 0.2 one has to determine the set $L(\beta) \cup I(\beta) \cup \beta(\partial[a, b])$ and then must investigate when one of the possible extreme subsets of β is indeed an extreme subset of β. The details are included in the next algorithm.

4.3 The algorithm

Based upon Theorem 0.3, Theorem 0.4 and the discussion in Subsection 4.2, next, we illustrate the steps of an algorithm in order to decide whether the collection $\{x_1, x_2, ..., x_n\}$ generates a lattice-subspace.

(1) Does the Wronskian of the functions $x_1, x_2, ..., x_n$ have at least one root in the interval $[a, b]$?

(2) Determine the sets $L(\beta), I(\beta), \beta(\partial[a, b])$ and the possible extreme subsets of β.

(3) Is one of the possible extreme subsets an extreme subset of β ?

(4) If step (3) holds, determine a positive basis of X.

5. The computational method

5.1 Method presentation

In this section, we present a procedure that will accurately implement the ideas of algorithm 4.3 while the main concern is to further calculate the positive basis (if one exists) in order to provide an exact description of the lattice-subspace. So, the first step of our approach consists of describing the functionality of the functions **wr**, **V**, **L**, **I**, **sisets** and **xitest** from Katsikis (2008).

According to Theorem 0.3, function **wr** checks if the Wronskian of the given collection $\{x_1, x_2, ..., x_n\}$ of $C[a, b]$ has at least one root in the interval $[a, b]$. In addition, the **wr** function provides the roots (if there exist any) of the Wronskian. So, in this case the program responds with the output:

```
The Wronskian has at least one root in [a,b]
```

and it yields the roots of the Wronskian. If, instead, the Wronskian does not have any roots in the interval $[a, b]$, then the program provides only the roots outside the interval $[a, b]$ (if there exist any).
Suppose that the given collection passes the Wronskian test, then we try to determine the possible extreme subsets of the basic curve β starting with the computation of the set $\beta(\partial[a, b])$. So, in our next step we call the function **V**. Function **V** first checks whether there are any real roots of the function $z(t) = \sum_{i=1}^{n} x_i$. If that is the case it displays the message

```
Possible non empty limit set
```

so that we can continue in order to determine the limit set of the curve β. Function **V** responds with a matrix whose columns are the elements of the set $\beta(\partial[a, b])$.
In the case of a non empty limit set, we use the function **L** in order to determine the limit set of the curve β. The output of the function **L** is a matrix whose columns are the elements of the set $L(\beta)$.
In order to determine the set

$$I(\beta) = \{\beta(t) : t \in Int([a, b]) \cap D(\beta) \textit{ and t is root of the equation } (2)\},$$

we use the function **I**. The function **I** provides a matrix whose columns are the elements of the set $I(\beta)$.
Suppose that $\{P_1, ..., P_n\}$ is a possible extreme subset of β and

$$\beta(t) = \xi_1(t)P_1 + ... + \xi_n(t)P_n.$$

In order to prove that $\{P_1, ..., P_n\}$ is an extreme subset of β we must show that $\xi_i(t) \geq 0$, for each i and each $t \in [a, b]$. So, for the next step in our approach, we need to construct all the possible extreme subsets of β and check whether there exists an extreme subset of β. To this end, we make use of the function **sisets** in order to generate all the possible extreme subsets of the curve β. Note that, **sisets** calls automatically the function **xitest** in order to determine the domain where each one of the $\xi_i(t)$ are negative. Let us denote by Δ_i the domain of negativity that corresponds to the function $\xi_i(t)$, for $i = 1, ..., n$ then, if for at least one of the $\xi_i(t)$, Δ_i has non empty intersection with the interval $[a, b]$, the set $\{P_1, ..., P_n\}$ is not an extreme subset of β. In the case where an extreme subset exists we determine the positive basis by using the formula

$$(b_1(t), b_2(t), ..., b_n(t)) = A^{-1}(x_1(t), x_2(t), ..., x_n(t))^T,$$

from Theorem 0.4.

5.2 Numerical examples

For the purpose of monitoring the performance, in the following we present some examples in $C[a, b]$ for various collections of functions together with the time responses we obtained when running these examples (Table 3).

Example 0.6. Let $x_1(t) = t^2 - 2t + 2$, $x_2(t) = -t^3 + 2t^2 - t + 2$ and $x_3(t) = t^3 - 3t^2 + 3t$ and X be the subspace of $C[0, 2]$ generated by the functions x_1, x_2, x_3.
Our first step consists of loading the data of the problem by using the following commands:

```
>> syms t
>> f=t^2-2*t+2;
>> g=-t^3+2*t^2-t+2;
>> h=t^3-3*t^2+3*t;
>> K=[f g h];
```

For the above set we start our analysis by using the Wroskian criterion (Theorem 0.3) through the **wr** function as follows:

```
>> wr(K,0,2);
```

The results, then, are as follows:

```
The Wronskian has at least one root in [a,b]
ans =
     1
```

Since, the Wroskian has at least one root in the interval [0,2], then we determine the set $\beta(\partial[a,b])$ by using the **V** function as the following code suggests:

```
>> betathetaomega=V(K,0,2);
```

As a result we get

```
limit set is empty
betathetaomega=
               1/2  1/2
               1/2  0
               0    1/2
```

Let us denote $P_1(\frac{1}{2},\frac{1}{2},0), P_2(\frac{1}{2},0,\frac{1}{2})$. Note that, if there is a possibility of a non empty limit set then the **V** function displays the message

```
Possible non empty limit set
```

Our next step is to determine the set $I(\beta)$ by using the **I** function which it provides a matrix whose columns are the elements of the set $I(\beta)$. Therefore,

```
>> I(K,0,2);
```

In this case the program yields

```
iotabeta =
          1/4
          1/2
          1/4
```

Let us denote $P_3(\frac{1}{4},\frac{1}{2},\frac{1}{4})$. Then, the set

$$\{P_1(\frac{1}{2},\frac{1}{2},0), P_2(\frac{1}{2},0,\frac{1}{2}), P_3(\frac{1}{4},\frac{1}{2},\frac{1}{4})\}$$

is the only possible extreme subset of β.

Suppose that

$$\beta(t) = \xi_1(t)P_1 + \xi_2(t)P_2 + \xi_3(t)P_3.$$

In order to prove that $\{P_1, P_2, P_3\}$ is an extreme subset of β we must show that $\xi_i(t) \geq 0$, for each i and each $t \in [0,2]$ and that $\sum_{i=1}^{3} \xi_i(t) = 1$. Since, in the present example, there is only one possible extreme subset of the curve β we can use the **xitest** function directly as the following code suggests:

```
>> b=[1/2 1/2 1/4;1/2 0 1/2;0 1/2 1/4];
>> xitest(K,b);
```

The results, then, are as follows:

```
sumofxi =
         1
ans =
     RealRange(Open(2), infinity)
ans =
     RealRange(-infinity,Open(0)),RealRange(Open(2),infinity)
ans =
     RealRange(-infinity, Open(0))
```

Thus, we have that $\sum_{i=1}^{3} \xi_i(t) = 1$. Also, $\Delta_1 = (2, +\infty), \Delta_2 = (-\infty, 0) \cup (2, +\infty)$ and $\Delta_3 = (-\infty, 0)$, therefore $\Delta_i \cap [0,2] = \varnothing$ for each i. As a result we have that $\{P_1, P_2, P_3\}$ is an extreme subset of β. Note that in the above code we denote by b the matrix whose columns are the elements of the set $\{P_1, P_2, P_3\}$. According to Theorem 0.4(ii), a positive basis $\{b_1, b_2, b_3\}$ of $X = [x_1, x_2, x_3]$ is given by the following code:

```
>> positivebasis=factor(inv(b)*K)
```

The results, then, are as follows:

```
positivebasis =
                [                 2]
                [-2(t - 2)(t - 1) ]
                [   -4t(t - 2)    ]
                [             2   ]
                [   2t(t - 1)     ]
```

hence $b_1(t) = -2(t-2)(t-1)^2$, $b_2(t) = -4t(t-2)$, $b_3(t) = 2t(t-1)^2$.

Example 0.7. Let $x_1(t) = t^2(t-1)^2$, $x_2(t) = t^4(t-1)^2$ and $x_3(t) = t^4(t-1)^4$ and X be the subspace of $C[-1,2]$ generated by the functions x_1, x_2, x_3. Working as before, we load the data of the problem by using the following commands:

```
>> syms t
>> f=t^2*(t-1)^2;
>> g=t^4*(t-1)^2;
>> h=t^4*(t-1)^4;
>> K=[f g h];
```

and for the above set we start our analysis by using the Wroskian criterion as follows:

```
>> wr(K,-1,2);
```

The results, then, are as follows:

```
The Wronskian has at least one root in [a,b]
ans =
     0
     3/4
     1
```

Thus, $0, 3/4$ and 1 are the roots of the Wroskian in the interval $[-1, 2]$. In this case, we determine the set $\beta(\partial[-1, 2])$ with the following code:

```
>> betathetaomega=V(K,-1,2);
```

As a result we get

```
>> Possible non empty limit set
betathetaomega =
                1/6  1/9
                1/6  4/9
                2/3  4/9
```

Let us denote $P_1(\frac{1}{6}, \frac{1}{6}, \frac{2}{3}), P_2(\frac{1}{9}, \frac{4}{9}, \frac{4}{9})$, then, from the above results, the possibility of a non empty limit set is further investigated. We can determine the limit set $L(\beta)$ of the curve β with the **L** function as the following code suggests:

```
>> limitset=L(K,-1,2);
```

In this case the program yields

```
limitset =
          1  1/2
          0  1/2
          0    0
```

Let us denote $P_3(1, 0, 0), P_4(\frac{1}{2}, \frac{1}{2}, 0)$. Our next step is to determine the set $I(\beta)$. So,

```
>> iotabeta=I(K,-1,2);
```

The results, then, are as follows:

```
iotabeta is empty
iotabeta =
          []
```

Thus, we have

$$I(\beta) \cup L(\beta) \cup \beta(\partial\Omega) = \{P_1(\frac{1}{6}, \frac{1}{6}, \frac{2}{3}), P_2(\frac{1}{9}, \frac{4}{9}, \frac{4}{9}), P_3(1, 0, 0), P_4(\frac{1}{2}, \frac{1}{2}, 0)\}.$$

In order to determine all the possible extreme subsets of the curve β we shall use the **sisets** function under the following code:

```
>> V=[1/6 1/9 1 1/2;1/6 4/9 0 1/2;2/3 4/9 0 0];
>> sisets(K,V);
```

where by V we denote the matrix whose columns are the elements of the set $I(\beta) \cup L(\beta) \cup \beta(\partial\Omega)$. Note that the **sisets** function not only calculates all the possible extreme subsets of the curve β but it also calls, automatically, the function **xitest** in order to determine the domain where each one of the $\xi_i(t)$ are negative. The results, then, are as follows:

```
S =
   1/6   1/9   1
   1/6   4/9   0
   2/3   4/9   0
ans =
     RealRange(Open(0.),Open(2.))
ans =
     RealRange(-infinity,Open(-1.)), RealRange(Open(3.),infinity)
ans =
     RealRange(-infinity,Open(-1.)), RealRange(Open(2.),infinity)
S =
   1/6   1/9   1/2
   1/6   4/9   1/2
   2/3   4/9   0
ans =
     RealRange(Open(1.),Open(2.))
ans =
     RealRange(Open(-1.),Open(1.))
ans =
     RealRange(-infinity,Open(-1.)), RealRange(Open(2.),infinity)
S =
   1/6   1   1/2
   1/6   0   1/2
   2/3   0   0
ans =
     NULL
ans =
     RealRange(-infinity,Open(-1.)), RealRange(Open(1.),infinity)
ans =
     RealRange(-infinity,Open(-1.)), RealRange(Open(3.),infinity)
S =
   1/9   1   1/2
   4/9   0   1/2
   4/9   0   0
ans =
     NULL
ans =
     RealRange(Open(1.),Open(2.))
ans =
     RealRange(-infinity,Open(0.)), RealRange(Open(2.),infinity)
```

In the previous results, S denotes each one of the possible extreme subsets of the curve β followed by the intervals of negativity of the corresponding $\xi_i(t)$. The conclusion here is that $X = [x_1, x_2, x_3]$ is not a lattice subspace of $C[-1, 2]$ since, from the above results, no set from the above candidates is an extreme subset.

We close this section with the time responses we obtained when running Example 0.6 and Example 0.7. Note that despite the relatively small collections of functions featured in these two examples the programs needed to perform a large number of elaborate checks in order to produce an answer. It easily follows that the **wr**, **V**, **L**, **I sisets** and **xitest** functions provide a practical numerical way to check whether the subspace X is a lattice-subspace, and thus allows us to simplify an extremely challenging task if it were to be done manually.

Matlab Function	Total time (in seconds)	Matlab Function	Total time (in seconds)
wr	0.035	**wr**	0.047
V	0.084	**V**	0.213
I	0.053	**I**	0.166
xitest	0.036	**sisets**	0.404
		L	0.108

Table 3. Time responses for Example 0.6 Time responses for Example 0.7

6. Applications in portfolio insurance

The theory of vector sublattices and lattice-subspaces has been extensively used in the last years in Mathematical Economics, especially in the areas of incomplete markets and portfolio insurance. In this section, we shall discuss this interconnection and we shall present computational methods in order to calculate the minimum-cost insured portfolio both in $\mathbb{R}^k$ and $C[a, b]$.

6.1 Portfolio insurance in $\mathbb{R}^k$

Let us assume that in the beginning of a time period there are N securities traded in a market. Let $\mathcal{S} = \{1, ..., S\}$ denote a finite set of states and $x_n \in \mathbb{R}^S_+$ be the payoff vector of security n in S states. The payoffs $x_1, x_2, ..., x_N$ are assumed linearly independent so that there are no redundant securities. By y_s we denote the N-dimensional vector of payoffs of all securities in state s. If $\theta = (\theta_1, \theta_2, ..., \theta_N) \in \mathbb{R}^N$ is a non-zero portfolio then its payoff is the vector

$$P(\theta) = \sum_{n=1}^{N} \theta_n x_n$$

and the set of payoffs of all portfolios is the linear span of the payoffs vectors $x_1, x_2, ..., x_N$ in $\mathbb{R}^S$ which we shall denote it by X, i.e.,

$$X = [x_1, x_2, ...x_N].$$

Let us also assume that $p = (p_1, p_2, ..., p_N) \in \mathbb{R}^N$ is a vector of security prices and $\mathbf{k} = \{k, k, ..., k\}, k \in \mathbb{R}$ denotes a vector with S coordinates. Then, the insured payoff on a portfolio $\theta = (\theta_1, \theta_2, ..., \theta_N)$ at a *floor* k is the contingent claim $P(\theta) \vee \mathbf{k}$ where

$$(P(\theta) \vee \mathbf{k})(s) = \max\{P(\theta)(s), \mathbf{k}(s)\} = \max\{P(\theta)(s), k\}, \textit{ for } s = 1, ..., S.$$

The solution of the following cost minimization problem is referred to as the *minimum-cost insured portfolio*,

$$\min_{\eta \in \mathbb{R}^N} p \cdot \eta$$

subject to

$$P(\eta) \geq P(\theta) \vee \mathbf{k}.$$

It is evident from the previous analysis that in order to calculate the minimum-cost insured portfolio we have to solve a linear programming problem. In Aliprantis et al. (2000), it is proved that if the supremum $P(\theta) \vee \mathbf{k}$ exists relative to X, i.e., if $P(\theta) \vee_X \mathbf{k}$ exists, then a portfolio that generates this payoff is the minimum-cost insured portfolio. The details are presented in the next theorem,

Theorem 0.5. *The minimum-cost portfolio exists and it is price-independent for every portfolio $\theta = (\theta_1, \theta_2, ..., \theta_N)$ and at every floor $\mathbf{k}$ if and only if X is a lattice-subspace of $\mathbb{R}^S$, then the minimum-cost insured portfolio θ^k is given by the formula*

$$P(\theta^k) = P(\theta) \vee_X \mathbf{k}.$$

Therefore, it is evident that in the special case where the subspace X is a lattice subspace we can find the minimum-cost insured portfolio by expressing the payoff vector and the floor vector in terms of the positive basis. Under these conditions, we can calculate the minimum-cost insured portfolio without making use of a linear programming method.
To this end, one can follow the following methodology:

- Calculate, by using the `SUBlatSUB` function, a positive basis $\{b_1, b_2, ..., b_N\}$ for the subspace X.
- If we denote by x and $\mathbf{k}$ the payoff and the floor vector, respectively, then express x, $\mathbf{k}$ in terms of the positive basis i.e., determine $\lambda_i, \mu_i,\ i = 1, 2, ..., N$ such that $x = \sum_{i=1}^N \lambda_i b_i$ and $\mathbf{k} = \sum_{i=1}^N \mu_i b_i$.
- Determine the supremum of the expressed payoff and floor vector i.e, determine the following supremum
$$x \vee_X \mathbf{k} = \sum_{i=1}^N \max\{\lambda_i, \mu_i\} b_i.$$
- If we denote by $\theta^k = (\theta_1, \theta_2, ..., \theta_N)$ the minimum-cost insured portfolio then θ^k is the solution of the following linear system
$$\sum_{n=1}^N \theta_n x_n = \sum_{i=1}^N \max\{\lambda_i, \mu_i\} b_i.$$

The above algorithmic procedure can be implemented through the following Matlab function, namely `mcpinsurance`:

```
function [theta_k]=mcpinsurance(a,floorvector,portfolio)
%a denotes a matrix whose columns are the given
%vectors x_1,x_2,...,x_N
%floorvector denotes the vector (k,k,...,k)
%portfolio denotes the theta vector
```

```
%Note that mcpinsurance requires the presence of the
%SUBlatSUB function
payoffvector=sum(a*diag(portfolio),2);
positivebasis=SUBlatSUB(a)';
r=(positivebasis\payoffvector);
k=(positivebasis\floorvector');
w=max(r,k)';
sup=w*positivebasis';
theta_k=a\sup';
```

We illustrate the above with the following example:

Example 0.8. We consider seven securities with payoffs in ten states given by

$$\begin{aligned}
&x_1 = (2,2,4,3,0,0,0,0,1,1),\ x_2 = (0,0,1,1,2,3,1,3,4,4),\\
&x_3 = (3,3,0,0,0,0,4,0,0,0),\ x_4 = (1,1,0,1,0,1,0,1,0,0),\\
&x_5 = (0,0,1,0,1,0,1,0,1,1),\ x_6 = (0,0,0,0,0,0,6,0,0,0)\\
&x_7 = (0,0,0,0,0,0,0,0,6,6).
\end{aligned}$$

Their linear span $X = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$ is a seven-dimensional subspace of $\mathbb{R}^{10}$. Consider the portfolio $\theta = (0,3,0,0,0,0,0)$ of three shares of security 2 at floor $\mathbf{1} = (1,1,1,1,1,1,1,1,1,1)$. It follows that one can calculate the minimum-cost insured portfolio, i.e., the portfolio that generates the payoff $x_2 \vee_X \mathbf{1}$, by using the following code,

```
>> theta_k=mcpinsurance(a,floorvector,portfolio)
```

Here the result is $\theta^k = (0, 3, 0.3333, 0, 0, -0.2222, 0)$.

It is clear that the `mcpinsurance` function is an important tool in order to calculate the minimum-cost insured portfolio.

6.2 Portfolio insurance in $C[a,b]$

In this section, we shall discuss the investment strategy called minimum-cost portfolio insurance in the case where the initial space is $C[a,b]$, the space of all continuous real functions defined on the interval $[a,b]$. In our model we use a method of comparing portfolios called portfolio dominance ordering from Aliprantis et al. (1998). This method compares portfolios by means of the ordering of their payoffs and under this consideration we are able to use the order structure of the payoff space together with the theory of lattice-subspaces. As in the case of $\mathbb{R}^k$, we calculate the minimum-cost insured portfolio under the general assumption that the asset span X is a lattice-subspace of the initial space $C[a,b]$. Also, in what follows we shall use the notation from Katsikis (2008).

The model of security markets we study here is extended over two periods, the period 0 and the period 1. We assume n securities labeled by the natural numbers $1, 2, ..., n$, acquired the period 0 and that these n securities are described by their payoffs at date 1. The payoff of the *ith* security is in general a positive element x_i of an ordered space E which is called *payoff space*. In addition, we assume that the payoffs $x_1, x_2, ..., x_n$ are linearly independent so that there are no redundant securities and that the securities have limited liability which ensures the positivity of $x_1, x_2, ..., x_n$.

In our model we consider E to be the space of real valued continuous functions $C[a,b]$ defined in an interval $[a,b]$. A portfolio is a vector $\theta = (\theta_1, \theta_2, ..., \theta_n)$ of $\mathbb{R}^n$ where θ_i is the number of shares of the *ith* security. The space $\mathbb{R}^n$ is then known as *portfolio space*. If $\theta = (\theta_1, \theta_2, ..., \theta_n) \in \mathbb{R}^n$ is a non-zero portfolio then its payoff is the vector

$$P(\theta) = \sum_{i=1}^{n} \theta_i x_i \in C[a,b].$$

The operator P is called the *payoff operator*. The pointwise ordering in $C[a,b]$, induces the partial ordering $\geq_P$ in the portfolio space $\mathbb{R}^n$ and is defined as follows: For each $\theta, \phi \in \mathbb{R}^n$ we have

$$\theta \geq_P \phi, \text{ if and only if } P(\theta) \geq P(\phi).$$

This ordering is known as the *portfolio dominance ordering*. The set of payoffs of all portfolios, or the range space of the payoff operator, is the linear span of the payoffs vectors $x_1, x_2, ..., x_n$ in $C[a,b]$ which we shall denote it by X, i.e.,

$$X = [x_1, x_2, ..., x_n].$$

The subspace X of $C[a,b]$ is called the *asset span* of securities or the *space of marketed securities*. Let us assume that $p = (p_1, p_2, ..., p_n) \in \mathbb{R}^n$ is a vector of security prices and θ, ϕ are two portfolios. Then, the insured payoff on the portfolio $\theta = (\theta_1, \theta_2, ..., \theta_n)$ at the *floor* ϕ and in the price p is the contingent claim $P(\theta) \vee P(\phi)$.
As in the case of $\mathbb{R}^k$, the solution of the following cost minimization problem is referred to as the *minimum-cost insured portfolio*, or a *minimum-cost insurance of the portfolio θ at the floor ϕ and in the price p*,

$$\min_{\eta \in \mathbb{R}^n} p \cdot \eta$$

subject to

$$P(\eta) \geq P(\theta) \vee P(\phi).$$

In Aliprantis et al. (2000) it is proved that if the supremum $P(\theta) \vee P(\phi)$ exists relative to X, i.e., if $P(\theta) \vee_X P(\phi)$ exists and X contains the order unit (risk-free payoff $\mathbf{1}$) then a portfolio that generates this payoff is the minimum-cost insured portfolio. The details are presented in the next theorem,

Theorem 0.6. *The minimum-cost insured portfolio exists and it is price-independent for every portfolio $\theta = (\theta_1, \theta_2, ..., \theta_n)$ and at every floor ϕ if and only if the asset span X, which contains the risk-free payoff, is a lattice-subspace of $C[a,b]$. In this case, the minimum-cost insured portfolio θ^{ϕ} is given by the formula*

$$P(\theta^{\phi}) = P(\theta) \vee_X P(\phi).$$

Therefore, if X is a lattice-subspace and $\{b_1, b_2, ..., b_n\}$ is a positive basis of the asset span X, then the minimum-cost insured portfolio θ^{ϕ} can be calculated with the following methodology:

- Expand $P(\theta)$ and $P(\phi)$ in terms of the positive basis $\{b_1, b_2, ..., b_n\}$.
- Suppose that $P(\theta) = \sum_{i=1}^{n} \lambda_i b_i$, $P(\phi) = \sum_{i=1}^{n} \mu_i b_i$. Then

$$P(\theta^{\phi}) = \sum_{i=1}^{n} (\lambda_i \vee \mu_i) b_i.$$

We shall illustrate the above with a simple example.

Example 0.9. Let $C[0,2]$ be the payoff space. Suppose that our model has three securities with payoff vectors $x_1(t) = t^2 - 2t + 2$, $x_2(t) = -t^3 + 2t^2 - t + 2$ and $x_3(t) = t^3 - 3t^2 + 3t$. The asset span X is the subspace of $C[0,2]$ generated by the functions x_1, x_2, x_3.
We will investigate whether X is a lattice-subspace and in the case of a lattice-subspace we shall determine a positive basis for X. Also, we shall calculate the minimum-cost insured portfolio θ^{ϕ} of the portfolio $\theta = (1,3,0)$ at the floor $\phi = (2,1,1)$.
From Example 0.6, we have that X is a lattice-subspace of $C[0,2]$ and the positive basis is defined by the functions

$$b_1(t) = -2(t-2)(t-1)^2, b_2(t) = -4t(t-2), b_3(t) = 2t(t-1)^2.$$

Let us denote by S, the matrix whose columns are the numeric coefficients of the symbolic polynomials of the positive basis, then $S = \begin{bmatrix} -2 & 0 & 2 \\ 8 & -4 & -4 \\ -10 & 8 & 2 \\ 4 & 0 & 0 \end{bmatrix}$
It follows that one can calculate the minimum-cost insured portfolio, i.e., the portfolio that generates the payoff $P(\theta) \vee_X P(\phi)$, with the following code:

```
>> syms t
>> x1=t^2-2*t+2;
>> x2=-t^3+2*t^2-t+2;
>> x3=t^3-3*t^2+3*t;
>> a=[x1;x2;x3];
>> theta=[1 3 0];
>> phi=[2 1 1];
>> Ptheta=sym2poly(theta*a);
>> Pphi=[0 sym2poly(phi*a)];
>> Rthetanew=S\Rtheta';
>> Pphinew=S\Pphi';
>> theta_phi=max(Pthetanew,Pphinew)
```

As a result we get $\theta^{\phi} = (2, 1.75, 1.5)$ which is the minimum-cost insured portfolio at every arbitrage price.

The procedure we followed in Example 0.9 allowed us to solve the minimization problem, without making use of any linear programming method and can be used in conjunction with the **wr**, **V**, **L**, **I**, **sisets** and **xitest** functions, in order to calculate minimum-cost insured portfolios.

7. Applications in the theory of efficient funds

In this section, we shall apply the `SUBlatSUB` function in order to determine the completion of security markets and the efficient funds of the market.
Let us assume that in the beginning of a time period there are n securities traded in a market. Let $\mathcal{S} = \{1, ..., m\}$ denote a finite set of states and $x_j \in \mathbb{R}^m_+$ be the payoff vector of security j in m states. The payoffs $x_1, x_2, ..., x_n$ are assumed linearly independent so that there are no redundant securities. If $\theta = (\theta_1, \theta_2, ..., \theta_n) \in \mathbb{R}^m$ is a non-zero portfolio then its payoff is the

vector $P(\theta) = \sum_{i=1}^{n} \theta_i x_i$. The set of payoffs of all portfolios is referred as the space of *marketed securities* and it is the linear span of the payoffs vectors $x_1, x_2, ..., x_n$ in $\mathbb{R}^m$ which we shall denote it by X, i.e., $X = [x_1, x_2, ...x_n]$.
For any $x, u \in \mathbb{R}^m$ and any real number a the vector

$$c_u(x, a) = (x - au)^+$$

is the *call option* and

$$p_u(x, a) = (au - x)^+$$

is the *put option* of x with respect to the *strike vector* u and *exercise price* a.
In what follows we shall use the theoretical background introduced in Kountzakis & Polyrakis (2006). Let U be a fixed subspace of $\mathbb{R}^m$ which is called *strike subspace* and the elements of U are the *strike vectors*. Then, the *completion by options* of the subspace X with respect to U is the space $F_U(X)$ which is defined inductively as follows:

- X_1 is the subspace of $\mathbb{R}^m$ generated by $\mathcal{O}_1$, where $\mathcal{O}_1 = \{c_u(x, a) | x \in X, u \in U, a \in \mathbb{R}\}$, denotes the set of call options written on the elements of X,
- X_n is the subspace of $\mathbb{R}^m$ generated by $\mathcal{O}_n$, where $\mathcal{O}_n = \{c_u(x, a) | x \in X_{n-1}, u \in U, a \in \mathbb{R}\}$, denotes the set of call options written on the elements of X_{n-1},
- $F_U(X) = \cup_{n=1}^{\infty} X_n$.

The completion by options $F_U(X)$ of X with respect to U is the vector sublattice of $\mathbb{R}^m$ generated by the subspace $Y = X \cup U$. The details are presented in the next theorem,

Theorem 0.7. *In the above notation, we have*

(i) $Y \subseteq X_1$,

(ii) $F_U(X)$ *is the sublattice* $S(Y)$ *of* $\mathbb{R}^m$ *generated by* Y, *and*

(iii) *if* $U \subseteq X$, *then* $F_U(X)$ *is the sublattice of* $\mathbb{R}^m$ *generated by* X.

Any set $\{y_1, y_2, \ldots, y_r\}$ of linearly independent positive vectors of $\mathbb{R}^m$ such that $F_U(X)$ is the sublattice of $\mathbb{R}^m$ generated by $\{y_1, y_2, \ldots, y_r\}$ is a *basic set* of the market.

Theorem 0.8. *Any maximal subset* $\{y_1, y_2, \ldots, y_r\}$ *of linearly independent vectors of* $\mathcal{A}$ *is a basic set of the market, where* $\mathcal{A} = \{x_1^+, x_1^-, \ldots, x_n^+, x_n^-\}$, *if* $U \subseteq X$ *and*
$\mathcal{A} = \{x_1^+, x_1^-, \ldots, x_n^+, x_n^-, u_1^+, u_1^-, \ldots, u_d^+, u_d^-\}$, *if* $U \subsetneq X$.

The space of marketed securities X is *complete by options* with respect to U if $X = F_U(X)$.

Theorem 0.9. *The space* X *of marketed securities is complete by options with respect to* U *if and only if* $U \subseteq X$ *and* $card R(\beta) = n$.

Theorem 0.10. *The dimension of* $F_U(X)$ *is equal to the cardinal number of* $R(\beta)$. *Therefore,* $F_U(X) = \mathbb{R}^m$ *if and only if* $card R(\beta) = m$.

It is clear, from the previous discussion that we can use the `SUBlatSUB` function to the problem of calculating the completion of security markets. Let us describe, in detail, the procedure with an example:

Example 0.10. Suppose that in a security market, the payoff space is $\mathbb{R}^{12}$ and the primitive securities are:

$$x_1 = (1, 2, 2, -1, 1, -2, -1, -3, 0, 0, 0, 0),$$
$$x_2 = (0, 2, 0, 0, 1, 2, 0, 3, -1, -1, -1, -2),$$
$$x_3 = (1, 2, 2, 0, 1, 0, 0, 0, -1, -1, -1, -2).$$

and that the strike subspace is the vector subspace U generated by the vector $u = (1, 2, 2, 1, 1, 2, 1, 3, -1, -1, -1, -2)$. Then, a maximal subset of linearly independent vectors of $\{x_1^+, x_1^-, x_2^+, x_2^-, x_3^+, x_3^-, u_1^+, u_1^-\}$ can be calculated by using the following code:

```
>> XX = [max(X,zeros(size(X)));max(-X,zeros(size(X)))];
>> S = rref(XX');
>> [I,J] = find(S);
>> Linearindep = accumarray(I,J,[rank(XX),1],@min)';
>> W = XX(Linearindep,:)
```

where X denotes a matrix whose rows are the vectors x_1, x_2, x_3, u. The results, then, are as follows:

```
W =
  1    2    2    0    1    0    0    0    0    0    0    0
  0    2    0    0    1    2    0    3    0    0    0    0
  20   36   40   3    18   16   3    24   2    2    2    4
  0    0    0    1    0    2    1    3    0    0    0    0
  0    0    0    0    0    0    0    0    1    1    1    2
```

We can determine the completion by options of X i.e., the space $F_U(X)$, with the SUBlatSUB function by using the following code:

```
>> [VectorSublattice,Positivebasis]=SUBlatSUB(W')
```

The results then are as follows

```
vector sublattice
positivebasis =
  0    0    0    0    0    0    0    0    3    3    3    6
  0    0    0    4    0    0    4    0    0    0    0    0
  0    0    0    0    0    20   0    30   0    0    0    0
  21   0    42   0    0    0    0    0    0    0    0    0
  0    40   0    0    20   0    0    0    0    0    0    0
```

Since we know a positive basis for $F_U(X)$, then we know the completion by options of X.

In the following we assume that U is the one-dimensional subspace of $\mathbb{R}^m$ generated by a vector $u \neq 0$ of $\mathbb{R}^m$. The notion of efficient funds have been studied in many economic articles (cf. John (1981); Ross (1976)). In Kountzakis & Polyrakis (2006) the authors introduced a definition that generalizes the notion of efficient funds. In particular, a vector $e \in F_u(X)$ is an $F_u(X)$ *-efficient fund* if $F_u(X)$ is the linear subspace of $\mathbb{R}^m$ which is generated by the set of nontrivial call options and the set of nontrivial put options of e.
It is clear that in order to calculate the efficient funds of the market we must determine , by using the SUBlatSUB function, a positive basis for $F_u(X)$. Then, in order to decide when an element e is an efficient fund of the market one has to apply the following theorem.

Theorem 0.11. *Suppose that $\{b_1, b_2, \ldots, b_\mu\}$ is a positive basis of $F_u(X)$, $u = \sum_{i=1}^{\mu} \lambda_i b_i$, and $\lambda_i > 0$ for each i. Then the vector $e = \sum_{i=1}^{\mu} k_i b_i$ of $F_u(X)$ is an $F_u(X)$-efficient fund if and only if $\frac{k_i}{\lambda_i} \neq \frac{k_j}{\lambda_j}$ for each $i \neq j$.*

We shall describe the computation procedure with an example:

Example 0.11. Suppose that in a security market, the payoff space is $\mathbb{R}^{12}$ and the primitive securities are as in Example 0.10 and $U = [u]$, where $u = (20, 36, 40, 3, 18, 16, 3, 24, 2, 2, 2, 4)$. Then the vector $e = (84, 16, 168, 4, 8, 20, 4, 30, 15, 15, 15, 30)$ is an $F_u(X)$-efficient fund of the market. Indeed, working as in Example 0.10 we have that a positive basis $\{b_1, b_2, b_3, b_4, b_5\}$ for $F_u(X)$ has the following form:

```
Positivebasis =
   0     0     0     0     0     0     0     0     3     3     3     6
   0     0     0     4     0     0     4     0     0     0     0     0
   0     0     0     0     0    20     0    30     0     0     0     0
  21     0     2     0     0     0     0     0     0     0     0     0
   0    40     0     0    20     0     0     0     0     0     0     0
```

Then, it easy to see that,

$$e = \sum_{i=1}^{\mu} k_i b_i = 5b_1 + b_2 + b_3 + 4b_4 + \frac{2}{5} b_5$$

and

$$u = \sum_{i=1}^{\mu} \lambda_i b_i = \frac{2}{3} b_1 + \frac{3}{4} b_2 + \frac{4}{5} b_3 + \frac{20}{21} b_4 + \frac{9}{10} b_5.$$

Therefore, $\frac{k_i}{\lambda_i} \neq \frac{k_j}{\lambda_j}$ for each $i \neq j$, and by Theorem 0.11 we have that e is an $F_u(X)$-efficient fund of the market.

8. Conclusions

This chapter describes new computational methods in order to determine vector sublattices and lattice-subspaces of $\mathbb{R}^k$ and $C[a, b]$. In order to reach our goal the study of a vector-valued function β is further involved by introducing new Matlab functions, namely `SUBlatSUB` , **wr**, **V**, **L**, **I**, **sisets** and **xitest**. The results of this work (cf. `mcpinsurance` function) can give us important tools in order to study the interesting problems of finding the minimum-cost insured portfolio and calculating the completion by options of a two-period security market as well as the efficient funds of the market.

The material of this chapter provides the opportunity for several research directions. In particular, we are convinced that, from the mathematical point of view, the proposed algorithms and methods can be further analyzed independently, in terms of formal numerical analysis. Also, the construction of a computational method that can solve the problem of whether a collection of linearly independent, positive functions, $x_1, x_2, ..., x_n$ of $C(\Omega)$ generates a lattice-subspace, where Ω denotes a compact Hausdorff topological space remains open. Finally, applications of the theory of lattice-subspaces and positive bases must be further investigated.

9. References

Abramovich, Y.A.; Aliprantis, C.D. & Polyrakis, I.A. (1994). Lattice-Subspaces and positive projections, *Proccedings of the Royal Irish Academy*, 94A, pp.237-253.

Aliprantis, C.D.; Brown, D.J. & Werner, J. (1997). Incomplete derivative markets and portfolio insurance, *Cowles Foundation Discussion Paper*, 1126R, pp.1-13.

Aliprantis, C.D.; Brown, D.J. & Polyrakis, I.A. (1998). Portfolio dominance and optimality in infinite security markets, *Journal of Mathematical Economics*, 30, pp.347-366.

Aliprantis, C.D.; Brown, D.J. & Werner, J. (2000). Minimum-cost portfolio insurance, *Journal of Economic Dynamics & Control*, 24, pp.1703-1719.

Aliprantis, C.D.; Polyrakis, I.A. & Tourky, R. (2002). The cheapest hedge, *Journal of Mathematical Economics*, 37, pp.269-295.

Barber, C.B; Dobkin, D.P. & Huhdanpaa, H.T. (1996). The Quickhull Algorithm for Convex *ACM Transactions on Mathematical Software*, 22, No. 4, pp.469-483.

Brown, D.J. & Ross, S.A. (1991). Spanning, valuation and options, *Economic Theory*, 1, pp.3-12.

Edmonds, J.; Lovász, L. & Pulleybank, W.R. (1982). Brick decompositions and the matching bank of graphs, *Combinatorica*, 2, pp.247-274.

Green, R. & Jarrow, R.A. (1987). Spanning and completness in markets with contigent claims, *Journal of Economic Theory*, 41, pp.202-210.

Henrotte, P. (1992). Existence and optimality of equilibria in markets with tradable derivative securities, *Technical report No.48 Stanford Institute for Theoretical Economics* , pp.1-39.

John, K. (1981). Efficient funds in a financial market with options: a new irrelevance proposition,*The Journal of Finance*, 36, pp.685-695.

Katsikis, V.N. (2007). Computational methods in portfolio insurance, *Applied Mathematics and Computation*, 189, pp.9-22.

Katsikis, V.N. (2008). Computational methods in lattice-subspaces of $C[a,b]$ with applications in portfolio insurance, *Applied Mathematics and Computation*, 200, pp.204-219.

Katsikis, V.N. (2009). A Matlab-based rapid method for computing lattice-subspaces and vector sublattices of $\mathbb{R}^n$: Applications in portfolio insurance, *Applied Mathematics and Computation*, 215, pp.961-972.

Kountzakis, C. & Polyrakis, I.A. (2006). The completion of security markets, *Decisions in Economics and Finance*, 29, pp.1-21.

Polyrakis, I.A. (1996). Finite-dimensional lattice-subspaces of $C(\Omega)$ and curves of $\mathbb{R}^n$, *Transactions of the American Mathematical Society*, 348, pp.2793-2810.

Polyrakis, I.A. (1999). Minimal lattice-subspaces, *Transactions of the American Mathematical Society*, 351, pp.4183-4203. *del Seminario Matematico e Fisico dell' Universita di Modena*, L, pp.327-348.

Polyrakis, I.A. (2003). Linear Optimization in $C(\Omega)$ and Portfolio Insurance, *Optimization*, 52,221-239.

Ross, S.A. (1976). Options and efficiency, *Quaterly Journal of Economics*, 90, pp.75-89.