STOCHASTIC SEARCH FOR BELL INEQUALITIES

Priscilla Mariani

In partial fulfillment of the requirements for the degree of Bachelor of Science with Honors

Physics Department
National University of Singapore
2013/2014

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Professor Valerio Scarani for his enthusiastic encouragement and valuable feedback on this project. I would also like to offer my special thanks to Dr. Jean-Daniel Bancal for his patient guidance and professional support throughout the project.

ABSTRACT

This project explores the behavior of the newly proposed method of finding the Bell inequalities of any 2-parties scenario. Starting from a point inside the set of probability distributions of a fixed scenario, sampling directions are randomized and the hit facet's equation is solved using the dual linear program.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Bell Experiment

Bell experiment studies the correlation between the results of measurements performed separately at a large distance, as illustrated in the following scheme.

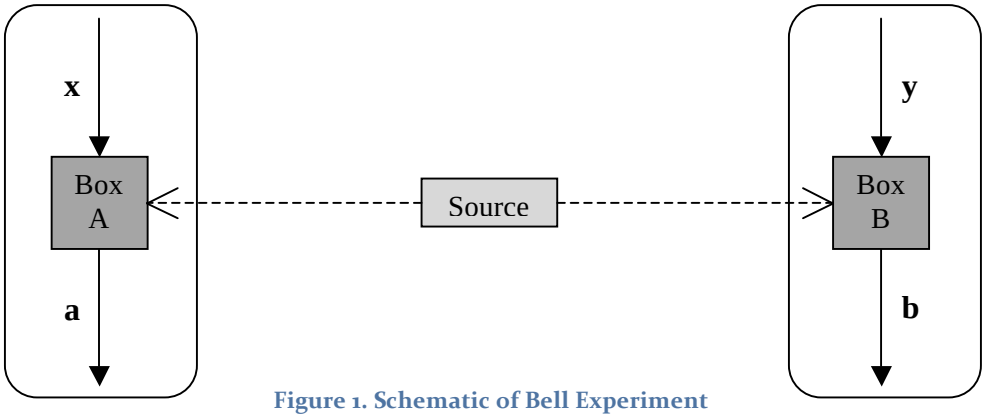

Figure 1. Schematic of Bell Experiment

The source sends the objects of measurement to the two black boxes in which there will be choices of possible measurement settings as inputs (x and y for box A and box B respectively) and results of the measurement (a and b and respectively). If the object of measurement is photon, examples of inputs could be polarization or spin measurement settings, while the outputs are the measured polarizations or spins. For future references, the following notations will be taken.

- Box A's possible measurement settings: $x \in \mathcal{X} = \{1, 2, \dots, M_A\}$
- Box A's possible outcomes: $a \in \mathcal{A} = \{1, 2, \dots, m_A\}$
- Box B's possible measurement settings: $y \in \mathcal{Y} = \{1, 2, \dots, M_B\}$
- Box B's possible outcomes: $b \in \mathcal{B} = \{1, 2, \dots, m_B\}$

After several takes, the joint statistics of the two boxes' results is then studied.

1.2 Local Variables

Correlations between the two distant black boxes can be explained by either of these schemes:

- Signaling: box A's party informing box B's party of his inputs/outputs and/or vice versa
- Pre-established agreement: both parties have set common rules on how to respond to the inputs

This project focuses on the second scheme, pre-established agreement, in which the correlation probability can be written as

$$P_{LV}(a,b|x,y) = \int d\lambda \ \rho(\lambda)\, P(a|x,\lambda)\, P(b|y,\lambda) \tag{1}$$

where λ is the possible mathematical descriptions that could have been invoked in the process of getting the observed statistics and ρ is its weight. "LV" stands for *local variables*, an expression corresponding to the pre-established agreement scheme.

While $P(a|x,\lambda)$ and $P(b|y,\lambda)$ can be any valid probability distributions, they can also be set to be deterministic, i.e.

$$P(a|x,\lambda) = \delta_{a=f(x,\lambda)}\,, \qquad P(b|y,\lambda) = \delta_{b=f(y,\lambda)} \tag{2}$$

which creates a special case of LV called *deterministic local variables* case (it is also equivalent to providing the list of outputs for all possible inputs). With this setting, it can be found that there are ${m_A}^{M_A}{m_B}^{M_B}$ amount of deterministic local points.

It has been proven that a family of probability distributions $\mathcal{P}_{X,Y}$ can be explained with pre-established agreement if and only if it can be explained with deterministic local variables. In other words, expression (1) can also be written as

$$P_{LV}(a,b|x,y) = \sum_{j=1}^{{m_A}^{M_A}} \sum_{k=1}^{{m_B}^{M_B}} \rho_{jk}\ \delta_{a=f_j(x)}\ \delta_{b=g_k(y)} \tag{3}$$

in which $\lambda \equiv (j,k)$ and $\sum_{j,k} \rho_{jk} = 1$.

1.3 Bell Inequalities

It has been observed that, for any fixed set of inputs and outputs $(\mathcal{X}, \mathcal{A}; \mathcal{Y}, \mathcal{B})$, the set $\mathcal{L}$ of all probability distributions obtainable by LV is convex, i.e. for every pair of $\mathcal{P}_1 \in \mathcal{L}$ and $\mathcal{P}_2 \in \mathcal{L}$, the points $q\mathcal{P}_1 + (1-q)\mathcal{P}_2$ for all $0 \leq q \leq 1$ are also elements of $\mathcal{L}$. Since every $\mathcal{P} \in \mathcal{L}$ is also obtainable by deterministic LV, it can be seen that the ${m_A}^{M_A}{m_B}^{M_B}$ deterministic local points serve as the extremal points of this set.

Geometrically, set $\mathcal{L}$ can be seen as a polytope embedded in $\mathbb{R}^D$ and bounded by $(D-1)$-dimensional facets. Below is a simple illustration of it.

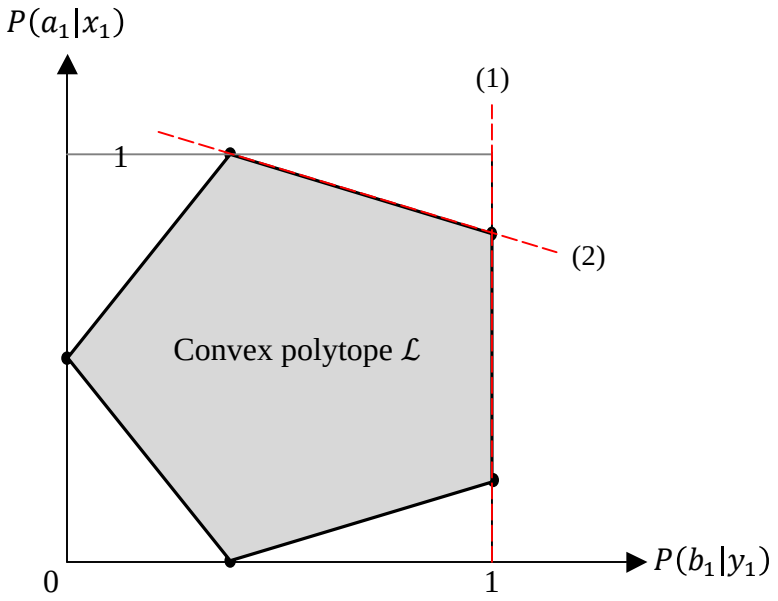

Figure 2. 2-dimensional illustration of polytope *L*

As can be seen, the illustrated convex polytope (shaded region) has 5 extremal points (black dots) and 5 facets (bolded lines). Line 1 is a trivial facet, coming from the restriction $P(b_1|y_1) \leq 1$, while line 2 is an example of the non-trivial facets and is associated with one out of 4 Bell inequalities of this illustrated set.

Polytope $\mathcal{L}$ can be represented in several choices of axis. For example, the Bell inequalities of the case $M_A = M_B = m_A = m_B = 2$ (named the CHSH inequalities) can be represented in a Collins-Gisin format (explained further in the later part of the report) as

$$P_A(1|1) + P_B(1|1) - P_{AB}(1,1|1,1) - P_{AB}(1,1|1,2) - P_{AB}(1,1|2,1) + P_{AB}(1,1|2,2) \leq 1$$

and its 3 other permutations, or in the correlators format as

$$E_{11} + E_{12} + E_{21} - E_{22} \leq 2$$

and its 3 other permutations where $E_{xy} = P(a = b|x, y) - P(a \neq b|x, y)$ are the correlators. The minimum amount of numbers (hence axis and dimension) needed to represent a full set of $\mathcal{L}$ is $M_A(m_A - 1) + M_B(m_B - 1) + M_A M_B(m_A - 1)(m_B - 1)$. This will be demonstrated in the later part of the report.

For any outcomes found within the polytope, there exist pre-established agreements that can be invoked in order to achieve those outcomes. Hence, these outcomes do not have true intrinsic randomness. Meanwhile, points which are outside the polytope (violating the Bell inequalities) do have true intrinsic randomness.

As can be seen from the expressions of the number of extremal points and minimum number of dimension, the complexity of the calculation increases exponentially with every additional input or output. Hence, new methods of finding the facets need to be explored in order to increase the efficiency of the process. The purpose of this project is to explore the behavior of a newly proposed “shooting” method.

Chapter 2

METHODOLOGY

2.1 Method Overview

This method takes advantage of the convexity characteristic of the probability polytope.

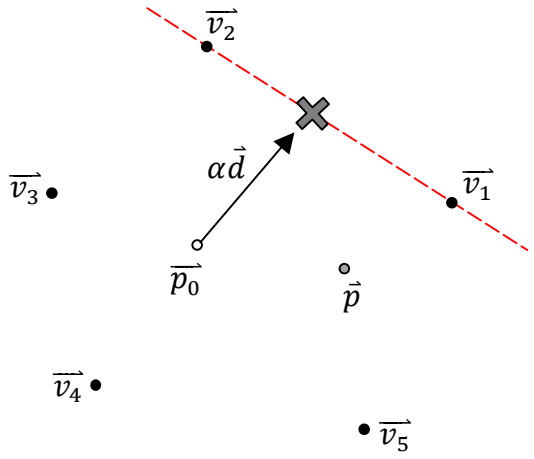

Figure 3. Illustration of the method

First, all the extremal points, $\overrightarrow{v_i}$, need to be found. Due to convexity, every point $\vec{p}$ in the polytope can be written in the form of the extremal points, i.e.

$$\vec{p} = \sum_{i=1}^{N} c_i \overrightarrow{v_i} \ , \qquad \sum_{i} c_i = 1 \ , \ \ c_i \geq 0$$

where N is the total number of extremal points and $\overrightarrow{c_i}$ is the coefficient associated with $\overrightarrow{v_i}$. The point $\overrightarrow{p_0}$ is chosen as a starting point of the "shooting" and a random vector $\vec{d}$ is chosen as the direction. Then, $\vec{d}$ is multiplied with the factor α which is maximized until the limit of

$$\overrightarrow{p_0} + \alpha\vec{d} = \sum_{i=1}^{N} c_i \overrightarrow{v_i} \ , \qquad \sum_i c_i = 1 \ , \ \ c_i \geq 0 \tag{4}$$

is hit, i.e. the point $\overrightarrow{p_0} + \alpha\vec{d}$ can no longer be written as a convex sum of the extremal points. Once a limit is hit, the equation of the hit facet can then be found. Afterwards, $\vec{d}$ is randomized again to continue the "shooting" and find more facets of the polytope.

2.2 Method Implementation

2.2.1 Generating Extremal Points

To generate all the extremal points of a set $\mathcal{L}$, all possible deterministic rules λ need to be exhausted. Below is an example for the case $M_A = M_B = m_A = m_B = 2$.

Box A settings: $x = (1, 2),\ a = (+, -)$

Box B settings: $y = (1, 2),\ b = (+, -)$

For each box, there are 4 possible sets of local rules:

Table 1. Local determistic rules of Box A and Box B each with 2 inputs and 2 outputs

Box A	Input	
	1	2
Rule 1 Output	+	+
Rule 2 Output	+	-
Rule 3 Output	-	+
Rule 4 Output	-	-

Box B	Input	
	1	2
Rule 1 Output	+	+
Rule 2 Output	+	-
Rule 3 Output	-	+
Rule 4 Output	-	-

Combining all possible pair of local rules of the 2 boxes, Table 2 is obtained. Each row represents an extremal point.

Table 2. Combined outcomes for Box A and Box B

Box 1 + Box 2		(x, y)			
		(1, 1)	(1, 2)	(2, 1)	(2, 2)
(a, b)	Rule 1 & Rule 1	(+, +)	(+, +)	(+, +)	(+, +)
	Rule 1 & Rule 2	(+, +)	(+, -)	(+, +)	(+, -)
	Rule 1 & Rule 3	(+, -)	(+, +)	(+, -)	(+, +)
	Rule 1 & Rule 4	(+, -)	(+, -)	(+, -)	(+, -)
	Rule 2 & Rule 1	(+, +)	(+, +)	(-, +)	(-, +)
	Rule 2 & Rule 2	(+, +)	(+, -)	(-, +)	(-, -)
	Rule 2 & Rule 3	(+, -)	(+, +)	(-, -)	(-, +)
	Rule 2 & Rule 4	(+, -)	(+, -)	(-, -)	(-, -)
	Rule 3 & Rule 1	(-, +)	(-, +)	(+, +)	(+, +)
	Rule 3 & Rule 2	(-, +)	(-, -)	(+, +)	(+, -)
	Rule 3 & Rule 3	(-, -)	(-, +)	(+, -)	(+, +)
	Rule 3 & Rule 4	(-, -)	(-, -)	(+, -)	(+, -)
	Rule 4 & Rule 1	(-, +)	(-, +)	(-, +)	(-, +)
	Rule 4 & Rule 2	(-, +)	(-, -)	(-, +)	(-, -)
	Rule 4 & Rule 3	(-, -)	(-, +)	(-, -)	(-, +)
	Rule 4 & Rule 4	(-, -)	(-, -)	(-, -)	(-, -)

There are 16 combined probabilities (shaded cells in Table 2) that can be chosen to be a set of axis. Ordering the variables by row, coordinates of the extremal points would then be expressed in terms of the values of ($P_{AB}(+,+|1,1), P_{AB}(+,-|1,1), P_{AB}(+,+|1,2)$, $P_{AB}(+,-|1,2), P_{AB}(-,+|1,1), P_{AB}(-,-|1,1), P_{AB}(-,+|1,2), P_{AB}(-,-|1,2)$, $P_{AB}(+,+|2,1), P_{AB}(+,-|2,1), P_{AB}(+,+|2,2), P_{AB}(+,-|2,2), P_{AB}(-,+|2,1)$, $P_{AB}(-,-|2,1), P_{AB}(-,+|2,2), P_{AB}(-,-|2,2)$). For example, coordinates of the first extremal point (first row) would be (1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0), corresponding to the 4 possible combined outcomes of $P_{AB}(+,+|1,1), P_{AB}(+,+|1,2), P_{AB}(+,+|2,1), P_{AB}(+,+|2,2)$. The second row would be (1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0), and so on for all 16 extremal points.

Table 3. Combined probability variables

	$P_B(+\|1)$	$P_B(-\|1)$	$P_B(+\|2)$	$P_B(-\|2)$
$P_A(+\|1)$	$P_{AB}(+,+\|1,1)$	$P_{AB}(+,-\|1,1)$	$P_{AB}(+,+\|1,2)$	$P_{AB}(+,-\|1,2)$
$P_A(-\|1)$	$P_{AB}(-,+\|1,1)$	$P_{AB}(-,-\|1,1)$	$P_{AB}(-,+\|1,2)$	$P_{AB}(-,-\|1,2)$
$P_A(+\|2)$	$P_{AB}(+,+\|2,1)$	$P_{AB}(+,-\|2,1)$	$P_{AB}(+,+\|2,2)$	$P_{AB}(+,-\|2,2)$
$P_A(-\|2)$	$P_{AB}(-,+\|2,1)$	$P_{AB}(-,-\|2,1)$	$P_{AB}(-,+\|2,2)$	$P_{AB}(-,-\|2,2)$

Choosing to express the points in this set of axis is also called the non-signaling probability format, which is indeed the format used to express the extremal points in this project. This is done so in order to avoid any biasness in randomizing the "shooting" direction in the later part of the project.

2.2.2 Constructing Directions

The first step taken after finding all the extremal points is choosing the starting point $\overrightarrow{p_0}$. In this project, $\overrightarrow{p_0}$ is chosen to be the average of all the extremal points so that it is guaranteed to be inside the polytope (note that the polytope is convex). Then, a random direction $\vec{d}$ is taken by generating a vector consisted of $M_A M_B m_A m_B$ random numbers.

Constraints

Merely randomizing elements of vector $\vec{d}$ might put the point $\overrightarrow{p_0} + \alpha\vec{d}$ out of the normalized no-signaling space, i.e. it does not fulfill the following 2 conditions:

- For all quadrants of input (examples are the bolded boxes in Table 3), the total probability must add up to 1. That is, for a $M_A = M_B = m_A = m_B = 2$ case, $P_{AB}(+,+|1,1) + P_{AB}(+,-|1,1) + P_{AB}(-,+|1,1) + P_{AB}(-,-|1,1) = 1$ and so on for the three other quadrants. This is a probability normalization constraint for each pair of inputs of the two boxes. Since $\overrightarrow{p_0}$ already fulfills this condition, the constraint for the elements of $\vec{d}$ is $P_{AB}(+,+|1,1) + P_{AB}(+,-|1,1) + P_{AB}(-,+|1,1) + P_{AB}(-,-|1,1) = 0$ instead.

- For each row and each column, the combined probabilities coming from every pair of the other box's inputs must subtract to zero. For example, for the first row in Table 3 (which corresponds to Box A's $P_A(+|1)$),
  $P_{AB}(+,+|1,1) + P_{AB}(+,-|1,1) - \{(P_{AB}(+,+|1,2) + P_{AB}(+,-|1,2)\} = 0$
  which is indeed true because, due to no-signaling constraint,
  $P_{AB}(+,+|1,1) + P_{AB}(+,-|1,1) = P_{AB}(+,+|1,2) + P_{AB}(+,-|1,2) = P_A(+|1).$
  Another example is
  $P_{AB}(+,+|1,1) + P_{AB}(-,+|1,1) - \{(P_{AB}(+,+|2,1) + P_{AB}(-,+|2,1)\} = 0$
  for the first column which corresponds to Box B's $P_B(+|1)$.
  Note that this must be applied to every pair of the other box's inputs, which means for the case $M_A = M_B = 2,\ m_A = m_B = 3$ the first row (corresponding to Box A's $P_A(+|1)$ must cover the pairs of Box B's first and second inputs, first and third inputs, as well as second and third inputs, and so on for all rows and columns. Such case would then have a total of 36 no-signaling constraints which also apply to $\vec{d}$.

Hence, for any scenario, there are $m_A m_B$ normalization constraints and $m_A M_A\ {}_{m_B}\mathrm{C}_2 + m_B M_B\ {}_{m_A}\mathrm{C}_2$ no-signaling constraints. However, these constraints are interrelated and they can be reduced to just 1 normalization constraint and $m_A M_A(m_B - 1) + m_B M_B(m_A - 1)$ no-signaling constraints. These truncated constraints for $\vec{d}$ are then translated into a matrix by taking the coefficients of the variables.

For a $M_A = M_B = m_A = m_B = 2$ case, an example constraint matrix would be

$$\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1
\end{pmatrix}$$

where each row corresponds to one constraint and the 16 columns correspond to the 16 axis chosen to express the extremal points in Section 2.2.1.

Null Space and Projector

All direction vectors $\overrightarrow{d'}$ that fulfill all the constraints would satisfy the equation

$$C\overrightarrow{d'} = 0 \tag{5}$$

where *C* is the constraint matrix. These vectors form a subspace of $\mathbb{R}^n$ called the *null space* which is the normalized no-signaling subspace. To find the basis of this null space, the MATLAB function `null()` is used.

```
basis = null(constraints)
```

A projector of this subspace is formulated as

```
projector = basis*basis'
```

which is then applied on every randomized direction $\vec{d}$.

2.2.3 Finding Facet Equation

The linear program applied to find the facet equation for each "shooting" direction is

Maximize α

such that $\alpha d'_k - \sum_i c_i v_{i_k} \leq -p_{0_k}$ (6)

$\sum_i c_i = 1$ (7)

$\alpha \geq 0, \;\; c_i \geq 0$

where $\overrightarrow{d'}$ is the projected direction, $\overrightarrow{v_i}$ is the extremal point, $\overrightarrow{p_0}$ is the starting point of the "shooting", and the index *k* corresponds to the *k*-th component of a vector. The upper limit of α can be expressed as

$$\alpha \leq \sum_k y_k(\alpha d'_k - \sum_i c_i v_{i_k}) + y_n \sum_i c_i \tag{8}$$

where y_k is a variable corresponding to *k*-th equation of expression (6) and y_n to equation (7). By comparing the coefficients of variables α and c_i, new constraints can be obtained. Also, by substituting equations (6) and (7) into equation (8), we get

$$\alpha \leq -\sum_k y_k p_{0_k} + y_n \tag{9}$$

With these, the dual linear program is obtained, i.e.

$$\text{Minimize } -\sum_k y_k p_{0_k} + y_n$$
$$\text{such that } \sum_k y_k d'_k \geq 1$$
$$-\sum_k y_k v_{i_k} + y_n \geq 0$$
$$y_k \text{ is free}$$

By solving this linear program, the equation of the hit facet can be calculated, that is

$$\sum_k y_k P_{AB_k} = y_n \tag{10}$$

with P_{AB_k} being the *k*-th basis of the chosen set of basis as illustrated in Section 2.2.1. For this part, the MATLAB toolbox and solver YALMIP and SeDuMi are used.

2.2.4 Changing Basis

After getting the coefficients of the facet equation, a change of basis needs to be taken. This step is taken due to possible redundancy, i.e. two different sets of coefficients could actually correspond to the same facet. For example, in the set of basis for the case $M_A = M_B = m_A = m_B = 2$ illustrated in the shaded cells of Table 3, the following two sets of coefficients

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

2	1	-1	-1
1	1	-1	-1
0	0	0	0
0	0	0	0

with $y_n = 0$ both refer to trivial positivity facet equation $P_{AB}(+,+|1,1) \geq 0$.

Taking a look at the complete table of probabilities for the case of $M_A = M_B = m_A = m_B = 2$ again, some patterns can be noted:

Table 4, Reduced variables

	$P_B(+\|1)$	$P_B(-\|1)$	$P_B(+\|2)$	$P_B(-\|2)$
$P_A(+\|1)$	$P_{AB}(+,+\|1,1)$	$P_{AB}(+,-\|1,1)$	$P_{AB}(+,+\|1,2)$	$P_{AB}(+,-\|1,2)$
$P_A(-\|1)$	$P_{AB}(-,+\|1,1)$	$P_{AB}(-,-\|1,1)$	$P_{AB}(-,+\|1,2)$	$P_{AB}(-,-\|1,2)$
$P_A(+\|2)$	$P_{AB}(+,+\|2,1)$	$P_{AB}(+,-\|2,1)$	$P_{AB}(+,+\|2,2)$	$P_{AB}(+,-\|2,2)$
$P_A(-\|2)$	$P_{AB}(-,+\|2,1)$	$P_{AB}(-,-\|2,1)$	$P_{AB}(-,+\|2,2)$	$P_{AB}(-,-\|2,2)$

- In the local probabilities cells, for every set of outputs coming from one input, one probability value is dependent on the rests. For example, $P_A(-|1) = 1 - P_A(+|1)$ and $P_B(-|1) = 1 - P_B(+|1)$. Hence, if the value of $P_A(+|1)$ is known, the value of $P_A(-|1)$ can be derived from it, and similarly for all other local inputs.
- For every row/column, due to the no-signaling condition, the sum of the combined probabilities in one quadrant is equal to the local probability corresponding to that row/column. In other words,
  $\sum_b P_{AB}(a, b|x, 1) = \sum_b P_{AB}(a, b|x, 2) = P_A(a|x)$ and
  $\sum_a P_{AB}(a, b|1, y) = \sum_a P_{AB}(a, b|2, y) = P_B(b|y)$. Hence, for every row and every column of each quadrant, one of the combined probabilities is dependent on the others.

With these properties observed, the variables shaded in red can be dropped and the remaining 8 variables $(P_A(+|1), P_A(+|2), P_B(+|2), P_B(+|2), P_{AB}(+,+|1,1), P_{AB}(+,+|1,2), P_{AB}(+,+|2,1), P_{AB}(+,+|2,2))$ can be used to represent the full polytope of the case $M_A = M_B = m_A = m_B = 2$. Applying the same concept to other scenarios, the expression $M_A(m_A - 1) + M_B(m_B - 1) + M_A M_B(m_A - 1)(m_B - 1)$ as the minimum number of variables needed to represent the full set of $\mathcal{L}$ is obtained. This set of basis is called the Collins-Gisin format, which is the format chosen to convert the previously found facet coefficients into.

To convert the basis, the dropped variables in the facet equation must be substituted by the new variables, while the rests remain. The substitutions happen following the same normalization and no-signaling constraints mentioned in Section 2.2.2. For example, in the case of Table 4, $P_{AB}(+,-|1,1)$ would be replaced by $\left(P_A(+|1) - P_{AB}(+,+|1.1)\right)$, $P_{AB}(-,+|1,1)$ by $\left(P_B(+|1) - P_{AB}(+,+|1,1)\right)$, and $P_{AB}(-,-|1,1)$ by $\left(1 - P_{AB}(+,+|1,1) - P_{AB}(+,-|1,1) - P_{AB}(-,+|1,1)\right)$ in which $P_{AB}(+,-|1,1)$ and $P_{AB}(-,+|1,1)$ would be re-substituted further, and similarly for all the other quadrants.

With these substitutions, it can be seen that the coefficient of the new variable $P_A(+|1)$ would be

$\left(coef\left(P_{AB}(+,-|1,1)\right) - coef\left(P_{AB}(-,-|1,1)\right)\right) +$

$\left(coef\left(P_{AB}(+,-|1,2)\right) - coef\left(P_{AB}(-,-|1,2)\right)\right)$ where $coef(X)$ stands for the coefficient of variable X, and similarly for $P_A(+|2)$. The coefficient of $P_B(+|1)$ would be

$\left(coef\left(P_{AB}(-,+|1,1)\right) - coef\left(P_{AB}(-,-|1,1)\right)\right) +$

$\left(coef\left(P_{AB}(-,+|2,1)\right) - coef\left(P_{AB}(-,-|2,1)\right)\right)$, and similarly for $P_B(+|2)$, while the new coefficient of $P_{AB}(+,+|1,1)$ would be

$coef\left(P_{AB}(+,+|1,1)\right) - coef\left(P_{AB}(+,-|1,1)\right) - coef\left(P_{AB}(-,+|1,1)\right) +$

$coef\left(P_{AB}(-,-|1,1)\right)$, and similarly for other $P_{AB}(a,b|x,y)$.

The same method is applied to all other scenarios.

2.2.5 Sorting Results

The *n* times of "shooting" would result in *n* sets of coefficients. To filter these sets of coefficients, four steps are taken.

1. Each set is normalized by dividing it with the largest element.
2. The elements are rounded to a certain decimal.
3. The rounded elements are made rational by using the MATLAB function `rat()`.
4. Unique sets are selected using the MATLAB function `unique()` and the

amount of sets compiled to each unique set is noted.

After the unique sets of facet coefficients are obtained, they are further transformed into their non-signaling probability canonical forms using `Faacets` software in order to compile together inequalities which are equivalent under relabeling of inputs and/or outputs. The final results are then compared with existing data.

Chapter 3

RESULTS

The canonical coefficients found are compared with known inequalities i.e. CHSH (from the case $M_A = M_B = m_A = m_B = 2$), I_{3322} (from the case $M_A = M_B = 3$ and $m_A = m_B = 2$) and CGLMP (from the case $M_A = M_B = 2$ and $m_A = m_B = 3$).

Case $M_A = M_B = m_A = m_B = 2$

The canonical form of CHSH in the full no-signaling probability format is

$$\begin{pmatrix} -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \leq 2$$

This inequality is found 466 times out of 10,000 times of sampling. The positivity inequality (with a canonical form of $(1 - 1)$) is found 9532 times, while invalid sets of coefficients are found 2 times. For comparison with the other 2 cases, this is equivalent to finding these inequalities 46.6 times, 953.2 times and 0.2 times respectively out of a 1000 times of sampling.

Case $M_A = M_B = 3,\ m_A = m_B = 2$

The Bell inequalities of this scenario include CHSH and I_{3322}. The canonical form of the latter is

$$\begin{pmatrix} -5 & 3 & -5 & 3 & -4 & 2 \\ 3 & -1 & 3 & -1 & 4 & -2 \\ -5 & 3 & -5 & 3 & 2 & -4 \\ 3 & -1 & 3 & -1 & -2 & 4 \\ -4 & 4 & 2 & -2 & 0 & 0 \\ 2 & -2 & -4 & 4 & 0 & 0 \end{pmatrix} \leq 12$$

Out of 10 runs each with 1000 times of sampling, these inequalities are found with these frequencies.

Take	**Positivity**	**CHSH**	**I_{3322}**	**Invalid**
1	814	177	4	5
2	816	181	2	1
3	799	196	5	0
4	820	176	1	3
5	817	175	6	2
6	801	188	8	3
7	806	189	2	3
8	832	160	8	0
9	789	204	4	3
10	799	193	5	3

This results in an average of (809±4) times of finding positivity inequalities, (184±4) times of finding CHSH inequalities, (4.5±0.8) times of finding I_{3322} inequalities and (2.3±0.5) times of finding invalid inequalities per 1000 times of sampling.

A straight 10,000 times of sampling is also taken, resulting in 8084 times of finding positivity inequalities, 1834 of finding CHSH inequalities, 69 times of finding I_{3322} and 13 times of finding invalid inequalities. These results are closely comparable with the data of per 1000 times of sampling.

Case $M_A = M_B = 2,\ m_A = m_B = 3$

The Bell inequalities of this scenario includes CHSH and CGLMP which has a canonical form of

$$\begin{pmatrix} -1 & 0 & 1 & -1 & 0 & 1 \\ 0 & 1 & -1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 1 \end{pmatrix} \leq 2$$

Out of 10 runs each with 1000 times of sampling, they are found with the following frequencies.

Take	**Positivity**	**CHSH**	**CGLMP**	**Invalid**
1	990	0	0	10
2	985	3	0	12
3	984	0	0	16
4	986	1	0	13
5	985	1	0	14
6	989	1	0	10
7	991	1	0	8
8	989	0	0	11
9	982	2	0	16
10	989	2	0	9

This results in an average of (987±1) times of finding positivity inequalities, (1.1±0.3) times of finding CHSH inequalities, 0 times of finding CGLMP inequalities, and (11.9±0.9) times of finding invalid inequalities out of each 1000 times of sampling.

Like the previous case, a straight 10,000 times of sampling is also taken, resulting in 9912 times of finding positivity inequalities, 10 times of finding CHSH inequalities, 0 times of finding CGLMP inequalities and 78 times of finding invalid inequalities. These results are also closely comparable with the data of per 1000 times of sampling.

The source of error of the invalid sets of coefficients most likely lies in the accuracy limit of the SeDuMi solver. Instead of the theoretical idea that the "shooting" direction must be exactly pointing at a vertex between facets in order to obtain a combined inequalities, SeDuMi's accuracy limit creates a range of in which more than one set of facet coefficients could merge together. This would result in an invalid set of coefficients.

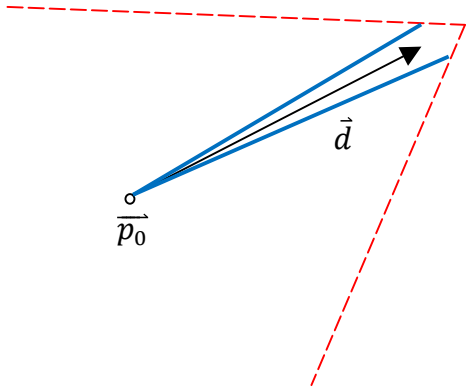

Figure 4. Illustration of SeDuMi's accuracy limit

Other limitations of the program include rounding-up error during the first filtering of the sets of coefficients as well as `Faacets` software's limitation in expressing the coefficients (they are bound to be rational numbers with a common denominator of maximum 10,000).

An additional approach that can be taken to improve the results of this method is creating a better way of setting the sampling direction, e.g. to shoot more at regions that have higher probability of finding the intended facets.

Chapter 4

CONCLUSION

This method starts from a point within the local set's polytope, generates stochastic sampling directions and finds the equations of the hit facets using the dual linear programs. It aims to be a faster and more efficient alternative method of finding Bell inequalities compared to other existing methods, e.g. PORTA (Fourier-Motzkin elimination).

In this project, the method has managed to find the CHSH inequalities and I_{3322} inequalities, but the CGLMP inequalities have not been found in the sampling that has been run.

For the case of $M_A = M_B = m_A = m_B = 2$, CHSH inequalities have been found 466 times out of 10,000 times of samplings.

For the case of $M_A = M_B = 3$ and $m_A = m_B = 2$, the CHSH inequalities have been found (184±4) times out of 1000 times of sampling, while the I_{3322} inequalities have been found (4.5±0.8) times out of 1000 times of sampling.

For the case of $M_A = M_B = 2$ and $m_A = m_B = 3$, the CHSH inequalities have been found (1.1±0.3) times out of 1000 times of sampling, while the CGLMP inequalities have not been found.

One step that can be taken to enhance this method is to improve the choosing of the "shooting" directions.

Moving Forward

These facets mark the boundaries of the local set of correlations. There is not true randomness within these facets because, for any case of correlations still fulfilling these inequalities, a pre-established agreement could be made. However, these facets can be used to find the boundaries of the quantum set of correlations, within which (shaded region in Figure 5) true randomness can happen.

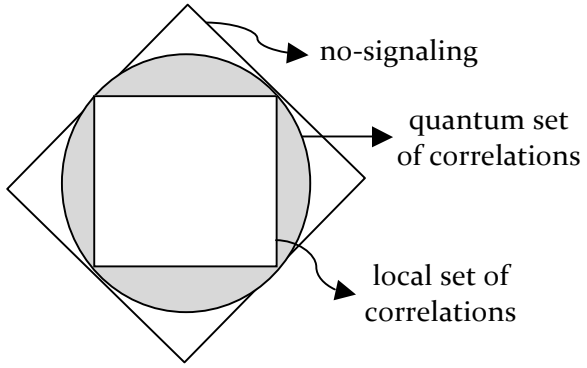

Figure 5. Different sets of correlations

One of the most important applications of this true randomness (existing in the shaded regions) is in Quantum Key Distribution which will create a safer communication method.

BIBLIOGRAPHY

Browne D, Hoban M. Bell inequalities made simple® [Internet]. University College London; 2011 [cited 2014 Apr 7]. Available from http://www.physics.usyd.edu.au/quantum/Coogee2011/Presentations/Browne.pdf.

Faacets [Internet]. [cited 2014 Apr 7]. Available from http://faacets.com/.

Hazewinkel M. Kernel of a matrix [Internet]. Encyclopedia of mathematics; 2011 [cited 2014 Apr 7]. Available from http://www.encyclopediaofmath.org/index.php?title=Kernel_of_a_matrix&oldid=12040

Linear programming duality [Internet]. Max planck institut informatik; 2011 [cited 2014 Apr 7]. Available from http://www.mpi-inf.mpg.de/departments/d1/teaching/ss11/OPT/lec8.pdf.

Scarani V. Bell inequalities as an operational notion. In: Acta physica slovaca: reviews & tutorials. Bratislava (Slovakia): Institute of Physics, Slovak Academy of Sciences; 2012. p. 353-368.

APPENDIX

Appendix

MATLAB FILES

Filename: extremalpoints_2boxes.m

```
function extremalpoints_2boxes()
%To generate the extremal points of any 2-parties scenario
input1 = {'1', '2', '3'};
input2 = {'1', '2', '3'};
output1 = {'+', '-'};
output2 = {'+', '-'};

box1rules = get_box_n_rules(input1, output1);
box2rules = get_box_n_rules(input2, output2);

local_outputlist = multiply_rows(box1rules, box2rules);
%outputlist = multiply_tensors(box1rules, box2rules)

box1poss_name = multiply_tensors(input1', output1');
%input1' and output1' need to be written this way in order to group according
to input
for i = 1:length(box1poss_name)
        box1poss_name_temp = box1poss_name{i, 1};
        box1poss_name{i, 1} = [box1poss_name_temp(2) box1poss_name_temp(1)];
end
%to flip the order of input and output in box1poss_name

box2poss_name = multiply_tensors(input2, output2);
%input2 and output2 need to be written this way in order to group according to
input
for i = 1:length(box2poss_name)
        box2poss_name_temp = box2poss_name{1, i};
        box2poss_name{1, i} = [box2poss_name_temp(2) box2poss_name_temp(1)];
end

allposs_name = multiply_tensors(box1poss_name, box2poss_name);
for i = 1:length(box1poss_name)
    for j = 1:length(box2poss_name)
        allposs_name_temp = allposs_name{i, j};
        allposs_name{i, j} = [allposs_name_temp(1) allposs_name_temp(3)...
            allposs_name_temp(2) allposs_name_temp(4)];
    end
end

allposs_name_temp = allposs_name';
allposs_name_temp = allposs_name_temp(:);
allposs_result_name = allposs_name_temp';

size_local_outputlist = size(local_outputlist);
allposs_result = zeros(size_local_outputlist(1), length(allposs_result_name));

for i = 1:size_local_outputlist(1)
    for j = 1:length(allposs_result_name)
        input1index = getindex(allposs_result_name{j}, 3, input1);
        input2index = getindex(allposs_result_name{j}, 4, input2);
        if and(allposs_result_name{j}(1) == local_outputlist{i,
1}{input1index}, ...
                alposs_result_name{j}(2) == local_outputlist{i, 2}{input2index})
            allposs_result(i, j) = 1;
        else
```

```
            allposs_result(i, j) = 0;
        end
    end

end

save('3322_result.mat', 'allposs_result_name', 'allposs_result',
'allposs_name', ...
    'box1poss_name', 'box2poss_name', 'input1', 'input2', 'output1', 'output2')

allposs_result_name
allposs_result

end

function tensor_product = multiply_tensors(tensor1, tensor2)
%concatenation tensor product of tensor1 and tensor2
%tensor1 and tensor2 are cells containing strings
size1 = size(tensor1);
size2 = size(tensor2);
tensor_product = cell(size1(1)*size2(1), size1(2)*size2(2));
for i = 1:size1(1)*size2(1)
    tensor1row = tensor1(ceil(i/size2(1)), :);
    tensor2row = tensor2(mod(i-1, size2(1))+1, :);
    for j = 1:size1(2)*size2(2)
        tensor_product{i, j} = [tensor1row{ceil(j/length(tensor2row))} ...
            tensor2row{mod(j-1, length(tensor2row))+1}];
    end
end
end

function row_product = multiply_rows(tensor1, tensor2)
%concatenation tensor product of tensor1's rows and tensor2's rows
%tensor1 and tensor2 are cells containing rows (and columns) of strings
size1 = size(tensor1);
size2 = size(tensor2);
row_product = cell(size1(1)*size2(1), 2);
for i = 1:size1(1)*size2(1)
    tensor1row = tensor1(ceil(i/size2(1)), :);
    tensor2row = tensor2(mod(i-1, size2(1))+1, :);
    row_product{i, 1} = tensor1row;
    row_product{i, 2} = tensor2row;
end
end

function box_n_rules = get_box_n_rules(input, output)
%box_n_rules = (local output of input 1, local output of input 2, ...)
box_n_rules = cell(length(output)^length(input), length(input));
for i = 1:length(output)^length(input)
    c = str2num(dec2base(i-1, length(output)));
    for j = 1:length(input)
        box_n_rules{i, j} =  output{getdigitn(c, j) + 1};
    end
end
end

function digitn = getdigitn(num, n)
%to get the n-th digit of an integer num
digitn = mod(floor(num/10^(n-1)), 10);
end

function index = getindex(source, source_index, matching_source)
source = source(source_index);
```

```
for n = 1: length(matching_source)
if source == matching_source{n}
   index = n;
end
end

end
```

Filename : facets_2boxes.m

```
function facets_2boxes()

load('3322_result.mat')

expts_size = size(allposs_result);
P_0 = mean(allposs_result, 1);
allposs_result;

% To compute an orthonormal basis of the normalized no-signalling space:

constraints = zeros((length(box1poss_name)*(length(input2)-...
    1)+length(box2poss_name)*...
    (length(input1)-1)+1), expts_size(2));
for i = 1
    for j = 1:length(output1)
        for k = 1:length(output2)
            constraints(i, (j-1)*length(box2poss_name)+k) = 1;
        end
    end
end

for i = 1:length(box1poss_name)
    for j = 1:length(input2)-1
        for k = 1:length(output2)
            constraints(1+(i-1)*(length(input2)-1)+j, ...
                (i-1)*length(box2poss_name)+k) = 1;
            constraints(1+(i-1)*(length(input2)-1)+j, ...
                (i-1)*length(box2poss_name)+j*length(output2)+k) = -1;
        end
    end
end

for i = 1:length(box2poss_name)
    for j = 1:length(input1)-1
        for k = 1:length(output1)
            constraints(1+length(box1poss_name)*(length(input2)-1)+ ...
                (i-1)*(length(input1)-1)+j, (k-1)*length(box2poss_name)+i) = 1;
            constraints(1+length(box1poss_name)*(length(input2)-1)+ ...
                (i-1)*(length(input1)-1)+j,
j*length(output1)*length(box2poss_name)+ ...
                (k-1)*length(box2poss_name)+i) = -1;
        end
    end
end
constraints
basis=null(constraints);
projector = basis*basis';

n = 1000;
%the number of sampling

facet_coef_old = zeros(n, expts_size(2)+1);
```

```
for i=1:n
    iyalmip = i
    d = rand(1, expts_size(2))-0.5;

    % Projceting the direction onto the normalized no-signalling space:
    d = (projector*d')';

    LambdaANDqi = sdpvar(expts_size(1)+1, 1);
    A = [[d' -allposs_result']; [0 ones(1, expts_size(1))]];
    b = [-P_0 1]';
    F = set(A*LambdaANDqi == b) + set(LambdaANDqi>=0);
    solvesdp(F, -LambdaANDqi(1), sdpsettings('verbose',0));
    facet_coef_old(i, :) = dual(F(1))';
end
%Facet eq.: facet_coef(i,1:expts_size(2)) * allposs_result_name' =
%   facet_coef(i, expts_size(2)+1)

var_old = [allposs_result_name 'constant'];

allposs_usedname = allposs_name;

local_remove_box1 =
length(output1):length(output1):(length(input1)*length(output1));
box1poss_usedname = box1poss_name;
box1poss_usedname(local_remove_box1) = [];
allposs_usedname(local_remove_box1, :) = [];

local_remove_box2 =
length(output2):length(output2):length(input2)*length(output2);
box2poss_usedname = box2poss_name;
box2poss_usedname(local_remove_box2) = [];
allposs_usedname(:, local_remove_box2) = [];

box1poss_usedname_hor = box1poss_usedname';
allposs_usedname_temp = allposs_usedname';
allposs_usedname_hor = allposs_usedname_temp(:)';

var_new = horzcat(box1poss_usedname_hor, box2poss_usedname,
allposs_usedname_hor, 'constant');

for i = 1:n
    isubs = i
    facet_coef_old_i = facet_coef_old(i, 1:length(allposs_result_name));
    coef_old_temp = reshape(facet_coef_old_i, length(box2poss_name),
length(box1poss_name));
    coef_old = coef_old_temp';

    coef_new_temp = zeros(1, length(var_new));
    for j = 1:length(input1)
        for k = 1:(length(output1)-1
            temp = 0;
            for l = 1:length(input2)
                temp = temp + coef_old(((j-1)*length(output1)+k),
l*length(output2)) - ...
                    coef_old(j*length(output1), l*length(output2));
            end
            coef_new_temp((j-1)*(length(output1)-1)+k) = temp;
        end
    end

    for j = 1:length(input2)
        for k = 1:(length(output2)-1)
            temp = 0;
```

```
            for l = 1:length(input1)
                temp = temp + coef_old(l*length(output1), ((j-
1)*length(output2)+k)) - ...
                    coef_old(l*length(output1), j*length(output2));
            end
            coef_new_temp(length(box1poss_usedname)+(j-1)*(length(output2)-1)+k)
= temp;
        end
    end

    for j = 1:length(input1)
        for k = 1:length(input2)
            for l = 1:(length(output1)-1)
                for m = 1:(length(output2)-1)
                    row = (j-1)*length(output1)+l;
                    col = (k-1)*length(output2)+m;
                    temp = coef_old(row, col) - coef_old(row, k*length(output2))
- ...
                        coef_old(j*length(output1), col) +
coef_old(j*length(output1), k*length(output2));
coef_new_temp(length(box1poss_usedname)+length(box2poss_usedname)+...
                        (j-1)*(length(output1)-1)*length(box2poss_usedname)+(l-
1)*length(box2poss_usedname)+...
                        (k-1)*(length(output2)-1)+m) = temp;
                end
            end
        end
    end


    temp = facet_coef_old(i, (length(allposs_result_name)+1));
    for j = 1:length(input1)
        for k = 1:length(input2)
            temp = temp - coef_old(j*length(output1), k*length(output2));
        end
    end
    coef_new_temp(length(var_new)) = temp;
    coef_new(i, :) = coef_new_temp;
    coef_new_normalized(i, :) = coef_new(i, :)/max(abs(coef_new(i, :)));
end

coef_new_normalized = cut(coef_new_normalized, 1e-2);
[N,D] = rat(coef_new_normalized, 9*10^-5);
coef_new_rat = N./D;
[coef_new_filtered,i,j] = unique(coef_new_rat, 'rows');
var_new
coef_new_filtered

for i = 1:size(coef_new_filtered, 1)
    coef_amount(i) = sum(j==i);
end
coef_amount

save('3322_facets_1000.mat', 'n', 'var_old', 'var_new', 'facet_coef_old',
'coef_new_rat', 'coef_new_filtered', 'coef_amount')
end

function out = cut(matrix,tol)
out = round(matrix/tol)*tol;
end
```

Filename: identify_family.m

```
%%function result = identifyFamily(coeffs, input1, input2, output1, output2)
% Identifies the Bell inequality family of an inequality in Collins-Gisin
% format, for the 2 parties, 2 inputs, 2 outputs scenario.
%
% For faacets library version 0.14

clear
load('3322_result.mat')
load('3322_facets_1000_10.mat')

input1_l = length(input1);
input2_l = length(input2);
output1_l = length(output1);
output2_l = length(output2);

% Check format
if ~isequal(size(coef_new_filtered), [size(coef_new_filtered, 1)
input1_l*input2_l*...
        (output1_l-1)*(output2_l-1)+input1_l*(output1_l-1)+input2_l*(output2_l-
1)+1])
    disp('Not the good size');
    return;
end

faacets_init;

box1 = num2str(output1_l);
for i = 1:input1_l-1
    box1 = [box1 ' ' num2str(output1_l)];
end

box2 = num2str(output2_l);
for i = 1:input2_l-1
    box2 = [box2 ' ' num2str(output2_l)];
end

s = Faacets.scenario(['{[' box1 ']' ' ' '[' box2 ']}']);

canonical_coeffs = {};

% For every filtered inequality
for i = 1: size(coef_new_filtered, 1)
    i
    coef_new_filtered(i,:);
    coef_i = coef_new_filtered(i, :);
    coef_i(end) = -coef_i(end);

    CG_mat = zeros(input1_l*(output1_l-1)+1, input2_l*(output2_l-1)+1);
    CG_mat(1, 1) = coef_i(end);

    for j = 2:input1_l*(output1_l-1)+1
        CG_mat(j, 1) = coef_i(j-1);
    end

    for j = 2:input2_l*(output2_l-1)+1
        CG_mat(1, j) = coef_i(input1_l*(output1_l-1)+j-1);
    end
```

```
    mid_elements = coef_i(input1_l*(output1_l-1)+input2_l*(output2_l-1)+1 :
end-1);
    mid_mat = reshape(mid_elements, input2_l*(output2_l-1),
input1_l*(output1_l-1));
    mid_mat = mid_mat';

    for j = 1:input1_l*(output1_l-1)
        for k = 1:input2_l*(output2_l-1)
            CG_mat(j+1, k+1) = mid_mat(j, k);
        end
    end

    CG_coef_i = reshape(CG_mat, 1, (input1_l*(output1_l-
1)+1)*(input2_l*(output2_l-1)+1));

    ineq = s.inequality('NGRepr', CG_coef_i);
    tmp = ineq.canonical;
    tmp = tmp(1);
    result = tmp.coeffs;
    result = result';

    canonical_coef{i} = result;
end

[canonical_coef_filtered, idx, idx2] = uniquecell(canonical_coef);

for i = 1:size(canonical_coef_filtered, 2)
    canonical_coef_amount(i) = (idx2==i)*coef_amount'
end

canonical_coef_filtered
canonical_coef_amount

save('3322_canonical_1000_10.mat', 'canonical_coef', 'coef_new_filtered', ...
    'canonical_coef_filtered', 'canonical_coef_amount')
```