UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACION

"ANALISIS COMPUTACIONAL DE LAS SOLUCIONES DE LA ECUACION DE YANG-BAXTER (YBE*) Y TEMAS RELACIONADOS."

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS (COMPUTACION)

P R E S E N T A :

JORGE CERVANTES OJEDA

DIRECTOR DE TESIS:
DR. VLADISLAV KHARTCHENKO.

MEXICO, D. F., **2004**

A mi esposa Carmen
y a mis padres

AGRADECIMIENTOS

Al Dr. Vladislav Khartchenko por haber dirigido esta tesis y por todo lo que pacientemente me enseño durante muchas horas de trabajo con él.

Al M. en C. Ricardo Paramont Hernández por sus valiosas enseñanzas acerca de los Algoritmos Genéticos.

A la Universidad Nacional Autónoma de México por brindarme todos los recursos necesarios para estudiar esta maestría.

Al CONACYT que sin el apoyo económico brindado no hubiera sido posible estudiar la maestría.

CONTENIDO

Capítulo 1 **Introducción.**

ANTECEDENTES

La ecuación de Yang-Baxter apareció por vez primera en el trabajo de E. Artin 1926 y 1947 [R 1] sobre teoría de trenzas. Luego fue introducida en el campo de la Física en un trabajo de Yang 1967 [R 2] como una condición de factorización de la matriz-S en el problema de varios cuerpos en dimensión uno y en un trabajo de Baxter 1972[R 3], 1982[R 4] acerca de módulos con solución exacta en el área de mecánica estadística.

Esta ecuación jugó también un papel importante en el método de diseminación inverso cuántico creado por Faddeev 1984 [R 5] para la construcción de sistemas integrables cuánticos.

Encontrar todas las soluciones de la ecuación de Yang-Baxter es una tarea difícil. Esta es una de las razones principales por las que surgió la teoría de grupos cuánticos. La teoría de grupos cuánticos tiene ahora aplicaciones en la teoría de nudos y la de campos conformales; en el problema de la cuantización del espacio-tiempo y en simetrías cuánticas (ver los libros Chaichian 1996 [R 8]; Kassel 1995[R 7]; Klimyk 1997[R 10]; Jantzen 1996 [R 9]). Actualmente hay también mucha literatura sobre soluciones de la ecuación de Yang-Baxter y sobre sus aspectos algebraicos (ver por ejemplo: Hietarinta 1992 [R 6]).

OBJETIVO

Demostrar que, mediante el uso de la computadora, es posible investigar las soluciones de la ecuación de Yang-Baxter en dimensión n y contribuir así en la investigación de la estructura de dichas soluciones y en general de los grupos cuánticos.

CONTRIBUCION Y RELEVANCIA

La ecuación de Yang-Baxter fue resuelta en dimensión 2 por Jarmo Hietarinta en 1992 [R 6]. La complejidad del problema de resolver dicha ecuación en dimensión superior a 2 crece exponencialmente. Hasta ahora no se conoce la solución general en dimensión mayor a 2. En esta tesis se presentan algunos métodos para encontrar y analizar las soluciones por medio del uso de la computadora. Al usar estos métodos se pueden encontrar soluciones de muy diversos tipos lo cual ayuda en la búsqueda de la solución general.

Algunos métodos son programas en Maple, otros son programas en C++. Se usan técnicas de programación avanzadas como Programación Orientada a Objetos, Recursividad y también técnicas de Inteligencia Artificial, específicamente de algoritmos genéticos [R 13].

Los programas elaborados durante el desarrollo de esta tesis no solamente ayudan en la solución de la ecuación de Yang-Baxter sino que pueden ser usados en otras ramas de la investigación como por ejemplo las Álgebras de Hopf Puntadas de dimensión finita, Álgebras Simétricas Cuánticas, entre otras.

Se presenta en forma implícita la construcción de Álgebras Shufflé y Álgebras Simétricas Cuánticas. En la construcción de estas últimas se siguen las ideas de M. Rosso [R 11], [R 12] pero con la diferencia de que el operador τ no es necesariamente invertible (Khartchenko 2003 [R 14]).

Capítulo 2 **Grupos básicos.**

Un ***semigrupo*** es un conjunto G sobre cuyos elementos se define alguna operación binaria, digamos $\bullet$, que satisface las siguientes condiciones:

- ✓ $a \bullet b \in G$ para todo $a,b \in G$ (cerradura).
- ✓ $(a \bullet b) \bullet c = a \bullet (b \bullet c)$ para todo $a,b,c \in G$ (asociatividad).
- ✓ Existe un elemento $e \in G$ tal que $e \bullet a = a \bullet e = a$ para todo $a \in G$ (identidad).

Un ***grupo*** es un semigrupo G en donde se satisface que

- ✓ Dado $a \in G$ existe un $b \in G$ tal que $a \bullet b = b \bullet a = e$(inversos).

A continuación se describen los grupos básicos utilizados en el desarrollo de ésta tesis.

2.1 ***Grupo de permutaciones.***

Definimos una ***lista*** como un conjunto cuyos elementos guardan un orden específico. Así, pues, existe un primer elemento, un segundo elemento, etcétera.

Denotaremos a una lista mediante corchetes: $[e_1, e_2, \ldots e_n]$ donde e_i es el i-ésimo elemento de la lista.

Una ***permutación-n*** es un mapeo $\sigma : [e_1, e_2, \ldots e_n] \rightarrow L_n$ donde L_n es una lista formada con los mismos elementos $e_1, e_2, \ldots e_n$, en algún orden. Es común denotar también una permutación-n como un mapeo $\sigma : I_n \rightarrow I_n$ siendo I_n el conjunto de listas formadas por una permutación de los primeros n números naturales.

Ejemplo 2—1:

$$\sigma : [a, b, c, d] \rightarrow [d, c, a, b]$$

es una permutación-4.

Denotaremos a ésta permutación de la siguiente manera:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{pmatrix}$$

ya que el primer elemento pasa a ocupar la tercera posición; el segundo pasa a ocupar la cuarta; el tercero, la segunda y el cuarto, la primera.

Un ***ciclo*** es una secuencia finita de números naturales p_i que se interpreta como una permutación de la siguiente manera:

Sea $(p_1\ p_2\ p_3\ p_4)$ un ciclo, éste representa a una permutación en la que el elemento en la posición p_1 pasa a ocupar la posición p_2, el de la posición p_2 pasa a la p_3, el de la p_3 a la p_4 y el de la p_4 a la p_1.

El ciclo: (1 3 2 4) que indica que el primer elemento ocupará la tercera posición; el tercero, la segunda; el segundo, la cuarta y el cuarto, la primera representa la misma permutación del Ejemplo 2—1:

Cuando en una permutación algún elemento no cambia de posición, esa posición no aparece en el ciclo.

Ejemplo 2—2:

$$\begin{pmatrix}1 & 2 & 3 & 4\\ 3 & 1 & 2 & 4\end{pmatrix}=(1\ \ 3\ \ 2)$$

TEOREMA 1 Toda permutación-n puede expresarse mediante una composición finita de ciclos.

Dada una permutación-n

$$\pi=\begin{pmatrix}1 & 2 & 3 & \cdots & n\\ \pi(1) & \pi(2) & \pi(3) & \cdots & \pi(n)\end{pmatrix}$$

definimos el ***soporte*** de π como el conjunto de posiciones que sufren algún cambio bajo la acción de π.

Formamos ahora un ciclo iniciando con alguna posición del soporte de π, suponiendo que 1 está en el soporte, tenemos $(1\ \ \pi(1)\ \ \pi(\pi(1))\ \ \pi(\pi(\pi(1)))\ \ \cdots)$. Si en éste ciclo no están incluidas todas la posiciones que son parte del soporte de π entonces se forma otro ciclo comenzando por alguna de aquellas posiciones que no se incluyeron en el primer ciclo. Este proceso se repite hasta agotar todas las posiciones del soporte de π. La composición de todos los ciclos formados será equivalente a π ya que cada uno de ellos contribuye a "mover" algunas posiciones en la misma manera en la que lo hace π haciendo juntos el mismo trabajo que ésta haría.

Ejemplo 2—3:

$$\begin{pmatrix}1 & 2 & 3 & 4 & 5 & 6\\ 4 & 3 & 6 & 1 & 2 & 5\end{pmatrix}=(1\ \ 4)(2\ \ 3\ \ 6\ \ 5)$$

En este caso los ciclos que forman la permutación son ***independientes*** entre sí, ya que no existe ninguna posición común entre ellos.

Podemos ahora decir que si, por ejemplo, σ = (1 3 2 4) entonces podemos aplicar σ a la lista [perro, gato, león, ratón]:

[perro, gato, león, ratón] σ = [ratón, león, perro, gato]

o también:

[perro, gato, león, ratón] (1 3 2 4) = [ratón, león, perro, gato]

al resultado obtenido podemos aplicarle nuevamente σ ya que éste es una lista:

[ratón, león, perro, gato] (1 3 2 4) = [gato, perro, ratón, león]

nótese que, aplicando nuevamente σ dos veces:

[gato, perro, ratón, león] (1 3 2 4) = [león, ratón, gato, perro]

[león, ratón, gato, perro] (1 3 2 4) = [perro, gato, león, ratón]

obtenemos nuevamente la lista original.

Podemos afirmar ahora que la ***composición*** de dos permutaciones-n es otra permutación-n. Del ejemplo anterior obtenemos:

[perro, gato, león, ratón] (1 3 2 4)(1 3 2 4) = [gato, perro, ratón, león],

que equivale a la permutación (1 2) (3 4) = $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$.

Entonces: (1 3 2 4) (1 3 2 4) = (1 2) (3 4).

Para obtener la ***permutación equivalente de la composición de otras dos***, se traduce cada posición a su nueva posición mediante ambas permutaciones. Por ejemplo: en el caso anterior (1 3 2 4) (1 3 2 4) se traduce de la posición 1 a la posición 3 mediante la primera permutación y luego ésta posición 3 se traduce mediante la segunda permutación a la posición 2. Entonces, en la permutación resultante de ésta composición, la posición 1 debe traducirse a la posición 2. Se procede de la misma manera con cada posición obteniéndose que:

de la posición 2 se pasa a la 4 y luego de la 4 a la 1.

de la posición 3 se pasa a la 2 y luego de la 2 a la 4.

de la posición 4 se pasa a la 1 y luego de la 1 a la 3.

Así, queda configurada la permutación resultante: (1 2) (3 4).

Otro ejemplo es: (2 3 6 5) (1 3 4) = (2 4 1 3 6 5).

Son ***ciclos independientes*** aquellos en los que no existe ninguna posición común entre ellos. La composición de ciclos independientes es una operación conmutativa.

Ejemplo 2—4:

(1 4)(2 3 6 5) = (2 3 6 5)(1 4).

TEOREMA 2.- El conjunto S_n, formado por todas las permutaciones-n, se convierte en un grupo mediante la operación binaria de composición de funciones.

Denotaremos la composición de funciones como un producto. Así, pues, dada una permutación-n σ escribimos:

Composición(σ , σ) = $\sigma \cdot \sigma = \sigma\ \sigma = \sigma^2$

Ejemplo 2—5:

Si σ = (1 3 2 4), entonces

σ^2 = (1 3 2 4)(1 3 2 4) = (1 2)(3 4)

$\sigma^3 = (1\ 2)(3\ 4)(1\ 3\ 2\ 4) = (1\ 4\ 2\ 3)$

$\sigma^4 = (1\ 4\ 2\ 3)(1\ 3\ 2\ 4) = \boldsymbol{id}$

donde ***id*** es la permutación identidad, la cual, no modifica la posición de ningún elemento. La permutación ***id*** juega el papel de la unidad (o elemento neutro de la composición) en el grupo S_n.

2.2 *Semigrupo de trenzas.*

Un ciclo que contiene solamente 2 elementos se llama ***transposición***, ya que solamente intercambia las posiciones de 2 elementos. Una transposición de la forma $(i\ i+1)$ se llama ***transvección-i*** y la denotaremos como t_i. Así, por ejemplo: el ciclo (2 5) es una transposición; la transposición (3 4) es una transvección-3 y se denota mediante t_3.

Dada una transposición T, se cumple siempre que $T^2 = \boldsymbol{id}$, luego $t_i^2 = \boldsymbol{id}$. Se cumple también siempre la relación:

$$t_{i+1}t_it_{i+1} = t_it_{i+1}t_i$$

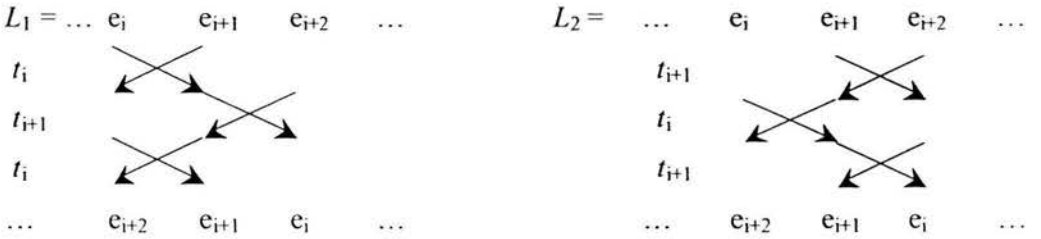
que llamaremos ***relación de trenza*** (en inglés: Braid relation) [R 1].

Comprobación de la relación de trenza:

$t_it_{i+1}t_i = (i\quad i+1)(i+1\quad i+2)(i\quad i+1) = (i\quad i+2)$

$t_{i+1}t_it_{i+1} = (i+1\quad i+2)(i\quad i+1)(i+1\quad i+2) = (i\quad i+2)$

Gráficamente, la relación de trenza puede verificarse así: dadas dos listas iguales $L_1 = L_2$, aplicamos $t_i\,t_{i+1}\,t_i$ a L_1 y $t_{i+1}\,t_i\,t_{i+1}$ a L_2:



quedando la misma lista al final en ambos casos.

TEOREMA 3.- Todos los ciclos se pueden expresar como una composición de un número finito de transposiciones.

En general se cumple que, un ciclo $(p_1\ p_2\ p_3\ \ldots\ p_n) = (p_1\ p_2)(p_1\ p_3)\ldots(p_1\ p_n)$.

Ejemplo 2—6:

$(1\ 4\ 2\ 5\ 3) = (1\ 4)(1\ 2)(1\ 5)(1\ 3)$.

TEOREMA 4.- Todas las transposiciones se pueden expresar como una composición de un número finito de transvecciones t_i.

Una transposición $(i\ n)$ se expresa como una composición de un número finito de transvecciones t_i de la siguiente manera:

$$(i\quad n) = t_i t_{i+1} t_{i+2} \ldots t_{n-2} t_{n-1} t_{n-2} \ldots t_{i+2} t_{i+1} t_i = \prod_{k=i}^{n-1} t_k \cdot \prod_{l=n-2}^{i} t_l$$

Ejemplo 2—7:

$$(2\ 6) = \prod_{k=2}^{5} t_k \prod_{l=4}^{2} t_l = t_2 t_3 t_4 t_5 t_4 t_3 t_2 = (2\ 3)(3\ 4)(4\ 5)(5\ 6)(4\ 5)(3\ 4)(2\ 3)$$

$$(3\ 5) = \prod_{k=3}^{4} t_k \prod_{l=3}^{3} t_l = t_3 t_4 t_3 = (3\ 4)(4\ 5)(3\ 4)$$

$$(1\ 4) = \prod_{k=1}^{3} t_k \prod_{l=2}^{1} t_l = t_1 t_2 t_3 t_2 t_1 = (1\ 2)(2\ 3)(3\ 4)(2\ 3)(1\ 2)$$

TEOREMA 5.- Todos los ciclos se pueden expresar como una composición de un número finito de transvecciones t_i.

Este teorema se demuestra mediante el TEOREMA 3 y el TEOREMA 4.

Ejemplo 2—8:

$(1\ 4\ 2\ 5\ 3) = (1\ 4)(1\ 2)(1\ 5)(1\ 3)$

$= (1\ 2)(2\ 3)(3\ 4)(2\ 3)(1\ 2)(1\ 2)(1\ 2)(2\ 3)(3\ 4)(4\ 5)(3\ 4)(2\ 3)(1\ 2)(1\ 2)(2\ 3)(1\ 2)$

$= t_1\ t_2\ t_3\ t_2\ t_1\ t_1\ t_1\ t_2\ t_3\ t_4\ t_3\ t_2\ t_1\ t_1\ t_2\ t_1$

ésta última expresión puede reducirse mediante $t_i^2 = \boldsymbol{id}$.

$= t_1\ t_2\ t_3\ t_2\ id\ t_1\ t_2\ t_3\ t_4\ t_3\ t_2\ id\ t_2\ t_1$

$= t_1\ t_2\ t_3\ t_2\ t_1\ t_2\ t_3\ t_4\ t_3\ t_2\ t_2\ t_1$

$= t_1\ t_2\ t_3\ t_2\ t_1\ t_2\ t_3\ t_4\ t_3\ id\ t_1$

$= t_1\ t_2\ t_3\ t_2\ t_1\ t_2\ t_3\ t_4\ t_3\ t_1$

la cual puede reducirse aún más si se usa la relación de trenza junto con $t_i\ t_j = t_j\ t_i$ cuando

$|i\text{-}j|>1$

$= t_1\ t_2\ t_3\ t_2\ t_1\ t_2\ t_1\ t_3\ t_4\ t_3$

$= t_1\ t_2\ t_3\ t_1\ t_2\ t_1\ t_1\ t_3\ t_4\ t_3$

$= t_1\ t_2\ t_3\ t_1\ t_2\ t_3\ t_4\ t_3$

$= t_1\ t_2\ t_1\ t_3\ t_2\ t_3\ t_4\ t_3$

$= t_2\ t_1\ t_2\ t_2\ t_3\ t_2\ t_4\ t_3$

$= t_2\ t_1\ t_3\ t_2\ t_4\ t_3$

$= t_2\ t_3\ t_4\ t_1\ t_2\ t_3$

TEOREMA 6.- Todas las permutaciones-n se pueden expresar como una composición de un número finito de transvecciones t_i con $0 < i < n$.

Una manera de demostrar el TEOREMA 6 es aplicando el TEOREMA 1 y luego el TEOREMA 5. Sin embargo dicha demostración no proporciona un método que garantice la representación mínima de toda permutación-n mediante transvecciones t_i. Otra forma de demostrar este teorema que sí garantiza lo anterior y que por lo tanto será usado en los siguientes capítulos es por inducción matemática:

Dada una permutación-n

$$\pi = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \pi(1) & \pi(2) & \pi(3) & \cdots & \pi(n) \end{pmatrix}$$

tomamos como base de la inducción lo siguiente: si $n = 1$, tenemos solamente la permutación: ***id***, la cual es una composición de cero transvecciones.

Hipótesis inductiva: El TEOREMA 6 es cierto para toda $n < k$.

Para completar la inducción, consideramos que la composición de ciclos es asociativa y tenemos entonces que

$$\pi = \pi \cdot id = \pi \cdot (t_{\pi(n)} t_{\pi(n)+1} t_{\pi(n)+2} \ldots t_{n-1} t_{n-1} t_{n-2} t_{n-3} \ldots t_{\pi(n)})$$

$$= (\pi \cdot t_{\pi(n)} t_{\pi(n)+1} t_{\pi(n)+2} \ldots t_{n-1}) \cdot t_{n-1} t_{n-2} t_{n-3} \ldots t_{\pi(n)}$$

$$= (\pi \cdot \prod_{i=\pi(n)}^{n-1} t_i) \cdot \prod_{i=n-1}^{\pi(n)} t_i$$

Definimos ahora la permutación-n $\pi_1 = \pi \cdot t_{\pi(n)} t_{\pi(n)+1} t_{\pi(n)+2} \ldots t_{n-1} = \pi \cdot \prod_{i=\pi(n)}^{n-1} t_i$.

Nótese que, en π_1, se cumple que $\pi_1(n) = n$, entonces la permutación no actúa efectivamente sobre la posición n. En este caso podemos considerar que π_1 es una permutación-$(n-1)$ para la cual se puede aplicar la hipótesis inductiva.

Por último escribimos π en función de π_1 :

$$\pi = \pi_1 \cdot t_{n-1} t_{n-2} \ldots t_{\pi(n)+2} t_{\pi(n)+1} t_{\pi(n)} = \pi_1 \cdot \prod_{i=n-1}^{\pi(n)} t_i$$

lo cual prueba que es efectivamente una composición de transvecciones.

Ejemplo 2—9:

Expresar la siguiente permutación como una composición de transvecciones.

$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 6 & 1 & 2 & 5 \end{pmatrix} = (1\ 4)(2\ 3\ 6\ 5)$ donde $n=6$, $\pi(n)=5$

$= ((1\ 4)(2\ 3\ 6\ 5)t_5)\ t_5 = (1\ 4)(2\ 3\ 5)\ t_5$ donde $n=5$, $\pi(n)=2$

$= ((1\ 4)(2\ 3\ 5)t_2 t_3 t_4)\ t_4 t_3 t_2\ t_5 = (1\ 3\ 4)\ t_4 t_3 t_2 t_5$ donde $n=4$, $\pi(n)=1$

$= ((1\ 3\ 4)t_1t_2t_3)\, t_3t_2t_1\, t_4t_3t_2t_5 = (1\ 2)\, t_3t_2t_1t_4t_3t_2t_5$ donde $n=2$, $\pi(n)=1$

$= t_1\ t_3t_2t_1t_4t_3t_2t_5$

$= t_1\ t_{3\ldots1}t_{4\ldots2}t_5$

Definimos el ***semigrupo de trenzas*** como el conjunto de los símbolos: s_i , $0 < i < n$, y todas sus concatenaciones en donde se cumplen las relaciones [R 1]:

$$s_i s_k = s_k s_i, \qquad i-k>1$$
$$s_{i+1}s_i s_{i+1} = s_i s_{i+1} s_i, \qquad 1 \le i \le n-1$$

Las concatenaciones en este semigrupo son palabras formadas por un número finito de símbolos s_i.

TEOREMA 7.- Si al semigrupo de trenzas se le adjunta la relación

$s_i^2 = \mathbf{id}$

entonces se forma un grupo equivalente a S_n *.[R 1]*

Bajo la composición se pueden obtener las relaciones adicionales siguientes:

$$s_{i+1}s_i^k s_{i+1}s_i = s_i s_{i+1} s_i^2 s_{i+1}^{k-1}, \quad k > 0$$
$$s_{i+1}s_i s_{i-k} s_{i+1} = s_i s_{i+1} s_i s_{i-k}, \quad k > 0$$
$$s_i s_{i+1} s_i s_{i-k} = s_{i-k} s_{i+1} s_i s_{i+1}, \quad k > 1$$

Capítulo 3 **La ecuación de Yang-Baxter (YBE).**

3.1 ***Forma invariante.***

Dado un conjunto M de n símbolos, llamamos V al espacio Vectorial cuya base son los símbolos en M. El producto de dos símbolos en M lo definimos como su concatenación formándose lo que llamaremos una palabra de longitud dos. $V \otimes V$ es un espacio vectorial cuya base son todas las palabras formadas con 2 símbolos de M.

Un operador lineal τ .

$$\tau : V \otimes V \to V \otimes V, \quad \text{cumple siempre que } (xy \cdot \tau) = \sum_{p,q \in M} \alpha_{x\,y}^{p\,q} pq,$$

donde "x" y "y" son dos símbolos en M y $\alpha_{x\,y}^{p\,q}$ es el coeficiente de la palabra "pq" formada por los símbolos p y q también en M asociado con los símbolos "x" y "y" en ese orden.

Al aplicar este operador a una palabra formada con 2 símbolos $x, y \in M$, se obtiene un polinomio con coeficientes $\alpha_{x\,y}^{p\,q}$ sobre todas las palabras formadas con 2 símbolos de M.

Ejemplo 3—1

Si $M = \{a, b\}$, y $\alpha_{a\,b}^{a\,a} = 2$, $\alpha_{a\,b}^{a\,b} = -1$, $\alpha_{ab}^{ba} = 0$, $\alpha_{a\,b}^{b\,b} = 5$ entonces

$$(ab) \cdot \tau = \alpha_{ab}^{aa} aa + \alpha_{ab}^{ab} ab + \alpha_{ab}^{ba} ba + \alpha_{ab}^{bb} bb = 2aa - ab + 0ba + 5bb = 2aa - ab + 5bb.$$

La manera de definir un operador τ es darle valores a cada una de las $\alpha_{x\,y}^{p\,q}$.

Definimos ahora el operador τ_i el cual se aplica a una palabra $P = a_1 a_2 a_3 ... a_m$, con $a_i \in M$. El resultado de la aplicación de τ_i sobre la palabra P es la concatenación de los primeros i-1 símbolos de P con $(a_i a_{i+1}) \cdot \tau$ y con los últimos símbolos de P a partir del (i+2)-ésimo.

$$a_1 a_2 a_3 \cdots a_m \cdot \tau_i = a_1 a_2 a_3 \cdots a_{i-1} (a_i a_{i+1}) \cdot \tau a_{i+2} a_{i+3} \cdots a_m.$$

Dado que $(a_i a_{i+1} \cdot \tau)$ es un polinomio, aplicamos la ley distributiva obteniéndose el siguiente polinomio

$$\sum_{p,q \in M} \alpha_{a_i a_{i+1}}^{p\,q} a_1 a_2 a_3 \cdots a_{i-1} pq a_{i+2} a_{i+3} \cdots a_m.$$

La aplicación de τ_k sobre un polinomio es la suma de las aplicaciones de τ_k a cada término del polinomio.

$$\left(\sum_{p,q \in M} \alpha_{a_i a_{i+1}}^{p\,q} a_1 a_2 a_3 \cdots a_{i-1} pq a_{i+2} a_{i+3} \cdots a_m \right) \tau_k = \sum_{p,q \in M} (\alpha_{a_i a_{i+1}}^{p\,q} a_1 a_2 a_3 \cdots a_{i-1} pq a_{i+2} a_{i+3} \cdots a_m) \tau_k.$$

La aplicación de una τ_k sobre un término con coeficiente es el producto del coeficiente por la aplicación de τ_k en la palabra del término.

$$(\alpha^{pq}_{a_i a_{i+1}} a_1 a_2 a_3 \cdots a_{i-1} pq a_{i+2} a_{i+3} \cdots a_m)\tau_k = \alpha^{pq}_{a_i a_{i+1}} (a_1 a_2 a_3 \cdots a_{i-1} pq a_{i+2} a_{i+3} \cdots a_m)\tau_k$$

Con estas propiedades del operador τ_i es posible calcular cualquier composición de τ_i 's sobre cualquier polinomio.

Ejemplo 3—2:

Con el operador τ definido con:

$$\begin{bmatrix} \alpha^{aa}_{aa} & \alpha^{ab}_{aa} & \alpha^{ba}_{aa} & \alpha^{bb}_{aa} \\ \alpha^{aa}_{ab} & \alpha^{ab}_{ab} & \alpha^{ba}_{ab} & \alpha^{bb}_{ab} \\ \alpha^{aa}_{ba} & \alpha^{ab}_{ba} & \alpha^{ba}_{ba} & \alpha^{bb}_{ba} \\ \alpha^{aa}_{bb} & \alpha^{ab}_{bb} & \alpha^{ba}_{bb} & \alpha^{bb}_{bb} \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 & 1 \\ 2 & -1 & 0 & 5 \\ 0 & 2 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

calcular:

$$(aabb \cdot \tau_2 \tau_1 \tau_3) = (((aabb) \cdot \tau_2) \cdot \tau_1) \cdot \tau_3$$

$$= ((a(ab) \cdot \tau b) \cdot \tau_1) \cdot \tau_3 = ((a(2aa - ab + 5bb)b) \cdot \tau_1) \cdot \tau_3$$

$$= ((2aaab - aabb + 5abbb) \cdot \tau_1) \cdot \tau_3$$

$$= (2(aaab) \cdot \tau_1 - (aabb) \cdot \tau_1 + 5(abbb) \cdot \tau_1) \cdot \tau_3$$

$$= (2(aa) \cdot \tau ab - (aa) \cdot \tau bb + 5(ab) \cdot \tau bb) \cdot \tau_3$$

$$= (2(4aa + bb)ab - (4aa + bb)bb + 5(2aa - ab + 5bb)bb) \cdot \tau_3$$

$$= (8aaab + 2bbab - 4aabb - bbbb + 10aabb - 5abbb + 25bbbb) \cdot \tau_3$$

$$= 8(aaab) \cdot \tau_3 + 2(bbab) \cdot \tau_3 - 4(aabb) \cdot \tau_3 - (bbbb) \cdot \tau_3 + 10(aabb) \cdot \tau_3$$

$$-5(abbb) \cdot \tau_3 + 25(bbbb) \cdot \tau_3$$

$$= 8aa(ab) \cdot \tau + 2bb(ab) \cdot \tau - 4aa(bb) \cdot \tau - bb(bb) \cdot \tau + 10aa(bb) \cdot \tau$$

$$-5ab(bb) \cdot \tau + 25bb(bb) \cdot \tau$$

$$= 8aa(2aa - ab + 5bb) + 2bb(2aa - ab + 5bb) - 4aa(-aa + bb) - bb(-aa + bb)$$

$$+10aa(-aa + bb) - 5ab(-aa + bb) + 25bb(-aa + bb)$$

$$= 16aaaa - 8aaab + 40aabb + 4bbaa - 2bbab + 10bbbb + 4aaaa - 4aabb + bbaa - bbbb$$

$$-10aaaa + 10aabb + 5abaa - 5abbb - 25bbaa + 25bbbb$$

$$= 10aaaa - 8aaab + 46aabb - 20bbaa - 2bbab + 34bbbb + 5abaa - 5abbb$$

Se cumple la *relación de trenza* para el operador τ cuando

Ecuación 3—1 $\tau_2\tau_1\tau_2 = \tau_1\tau_2\tau_1$

que llamaremos la forma invariante de la ecuación de Yang-Baxter. [R 1].

3.2 *Forma coordenada.*

La ***forma coordenada*** de la ecuación de Yang-Baxter describe las condiciones que deben cumplir las $\alpha_{x\ y}^{p\ q}$ para que la relación de trenza se cumpla.

En el apéndice 11.3 se puede ver cómo se deduce ésta forma de la ecuación de Yang-Baxter que son las

Ecuaciones 3—2 $$\sum_{p,q,r\in M} \alpha_{s\ v}^{p\ q}\alpha_{q\ n}^{r\ m}\alpha_{p\ r}^{u\ w} = \sum_{p,q,r\in M} \alpha_{v\ n}^{p\ q}\alpha_{s\ p}^{u\ r}\alpha_{r\ q}^{w\ m}$$

la cual describe un sistema de ecuaciones donde cada ecuación se forma dándole valores a las variables $m, n, s, u, v, w \in M$.

Si M tiene n elementos, entonces el sistema tiene n^6 ecuaciones cada una con n^3 términos de cada lado de la igualdad con 3 factores cada uno y en total tiene n^4 incógnitas que son las $\alpha_{x\ y}^{p\ q}$.

El conjunto de las $\alpha_{x\ y}^{p\ q}$ tiene las propiedades de un tensor-(2,2). Ver apéndice 11.1.

Capítulo 4 **Álgebras.**

4.1 *Álgebras Libres.*

Definimos el *Producto Tensorial* o *Producto de Kroneker* entre dos espacios lineales V_1 y V_2 como el espacio cuya base es el conjunto de todas las palabras formadas mediante la concatenación de un elemento de la base de V_1 y un elemento de la base de V_2 en ese orden y lo denotamos como

$$V_1 \otimes V_2 .$$

Sea V un espacio lineal, denotamos como $V \otimes V$ al espacio cuya base es el conjunto de todas las palabras de longitud 2 que se pueden formar con los elementos de la base de V.

Si denotamos la unión de dos espacios con el símbolo de la suma, el espacio $V + V \otimes V$ es el espacio cuya base es el conjunto de todas las palabras de longitud 1 y 2 que se pueden formar con los elementos de la base de V.

Denotaremos como $V^{\otimes n}$ al espacio cuya base es el conjunto de todas las palabras de longitud n que se pueden formar con los elementos de la base de V.

Definimos el *Espacio Tensorial* de un espacio V como el espacio cuya base es el conjunto de todas las palabras de cualquier longitud que se pueden formar con los elementos de la base de V y lo denotamos como

$$T(V) = \sum_{i=0}^{\infty} V^{\otimes i} .$$

Un *Álgebra Tensorial* o *Álgebra Libre* se construye sobre un espacio tensorial con la operación de concatenación $\cdot$ y se denota

$$\langle T(V), \cdot \rangle .$$

Cuando V es un espacio cuya base es el conjunto {x, y} al Álgebra Libre anterior se le llama Álgebra Libre de 2 variables ó Álgebra Tensorial del espacio de dimensión 2 ó Álgebra de los polinomios no conmutativos de 2 variables y se denota como

$$k\langle x, y \rangle$$

4.2 *Álgebras Shufflé.*

Las álgebras Shufflé reciben su nombre de la palabra francesa Shufflé o de la inglesa Shuffle que significan "Mezcla". Entonces el álgebra Shufflé sirve para calcular mezclas; solamente que en cierta manera particular que se describe a continuación.

Cuando por ejemplo, barajamos un juego de cartas, primero partimos el juego en dos montones y procedemos a "mezclarlas" quedando un solo montón. Este montón es una

mezcla de los dos montones, entonces podemos decir que la operación de mezclar es una operación binaria aplicada a dos montones de cartas.

Nótese que, al mezclar los dos montones, las cartas de un montón quedan intercaladas entre las del otro montón pero el orden de las cartas de cada uno de los montones no se altera. En otras palabras, si dos cartas en un montón están en cierto orden, entonces después de mezclar la carta que estaba primero sigue estando primero que la otra aunque posiblemente con otras cartas entre ellas (que provienen del otro montón).

Supongamos ahora que originalmente la baraja está ordenada por palo y por número y que al partirla en dos montones lo hacemos exactamente por la mitad. Tenemos entonces que cada montón tiene todas las cartas de dos de los palos de la baraja ordenadas del as hasta el rey.

Si efectuamos una mezcla "perfecta" tendríamos un par de ases seguido de un par de 2's seguido de un par de 3's etcétera con los palos de cada par ordenados siempre en la misma manera. Existen solamente 2 posibles mezclas "perfectas": una que empieza con la primera carta del montón de la mano izquierda y la que empieza con la primera carta del montón de la mano derecha.

Si la mezcla no es perfecta, el montón obtenido tiene un orden distinto. ¿De cuantas maneras distintas se puede efectuar una mezcla? La respuesta es 495,918,532,948,104. ¡Casi 500 billones! La manera de calcular lo anterior es considerar que cada carta de uno de los montones ocupará un "lugar" en el montón resultante. Así, se deberán elegir 26 posiciones del montón resultante para colocar las cartas de un montón y el resto de las posiciones serán para las cartas del otro montón. Debemos entonces calcular el número de combinaciones de 26 objetos tomados de un conjunto de 52.

$$\binom{52}{26}=\frac{52!}{26!(52-26)!}=495,918,532,948,104$$

Supongamos ahora que el corte de la baraja no se hizo exactamente por la mitad. En un montón tenemos k cartas y en el otro 52-k cartas. El numero de posibles mezclas estaría dado por:

$$\binom{52}{k}=\binom{52}{52-k}=\frac{52!}{k!(52-k)!}$$

Para k = 1 obtenemos 52, lo cual es obvio; para k = 2 son 1326.

El álgebra de mezclas o *Álgebra Shufflé* (Clásica) sirve para calcular todas las mezclas posibles de dos conjuntos cuyos elementos guardan un orden específico. Es decir, define una operación binaria que se aplica a 2 listas cuyo resultado es la *suma* de *todas* las posibles mezclas de los elementos de dichas listas sin que se pierda el orden de los elementos dentro de ellas.

4.2.1 Álgebra Shufflé Clásica.

Decimos que una permutación π es una "***permutación (k, l)-Shufflé***" si, al ser aplicada sobre una lista L de $k + l$ elementos, los primeros k elementos de L conservan el mismo orden entre ellos y los últimos l elementos también.

En otras palabras: Una permutación π es una "***permutación (k, l)-Shufflé***" si, para cada $1 \le i < j \le k + l$ es cierto que $1 \le i \le k < j \le k + l$ o bien $\pi(i) < \pi(j)$. Es decir, cuando $1 \le i < j \le k$ o $k < i < j \le k + l$ se debe cumplir $\pi(i) < \pi(j)$.

Ejemplo 4—1:

$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix}$ es una permutación (2,3)-Shufflé, ya que, $\pi(i) < \pi(j)$ se cumple cuando $1 \le i < j \le 2$ y también cuando $2 < i < j \le 5$.

Si aplicamos esta permutación a una lista de 5 elementos:

$$[abcde]\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix} = [cdaeb]$$

observamos que los primeros 2 elementos ab conservan el mismo orden y los últimos 3 cde también.

Definimos el ***producto Shufflé*** $\odot$ de dos listas L_1 y L_2 [R 15] cuyas longitudes son k y l respectivamente como la sumatoria de las aplicaciones de todas las permutaciones (k, l)-Shufflé de $k + l$ elementos sobre la concatenación de las listas L_1 y L_2.

Ecuación 4—1

$$L_1 \odot L_2 = \sum_{\pi \in S(k,l)} L_1 L_2 \cdot \pi$$

donde $S(k, l)$ es el conjunto formado por todas las permutaciones (k, l)-Shufflé de $k + l$ elementos y $L_1 L_2$ es la concatenación de las listas L_1 y L_2 cuyas longitudes son k y l respectivamente.

Ejemplo 4—2:

$$ab \odot cd = \sum_{\pi \in S(2,2)} abcd \cdot \pi$$

donde $S(2,2)$ es el conjunto formado por las siguientes permutaciones:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$$

el cual puede expresarse como:

id, (2 3), (2 4 3), (1 2 3), (1 2 4 3), (1 3)(2 4)

$= \mathbf{id}, t_2, t_2t_3, t_2t_1, t_2t_1t_3, t_2t_1t_3t_2.$

entonces podemos escribir:

$ab \odot cd =$

$$= (abcd)\cdot id + (abcd)\cdot t_2 + (abcd)\cdot t_2 t_3 + (abcd)\cdot t_2 t_1 + (abcd)\cdot t_2 t_1 t_3 + (abcd)\cdot t_2 t_1 t_3 t_2$$

$$= abcd + acbd + acdb + cabd + cadb + cdab$$

Definiremos el producto Shufflé de un polinomio con una palabra mediante la ley distributiva:

$$(\sum_i p_i)\odot q = \sum_i (p_i \odot q)$$

y el producto Shufflé de un polinomio con otro polinomio como:

$$(\sum_i p_i)\odot(\sum_j q_j) = \sum_{i,j} (p_i \odot q_j)\,.$$

Definimos el ***Álgebra Shufflé Clásica*** como un espacio tensorial $T(V)$ sobre un espacio lineal V con la operación de producto Shufflé

$$\langle T(V), \odot \rangle\,.$$

TEOREMA 8.- El Álgebra Shufflé Clásica es conmutativa *y* asociativa.

Es claro que, al barajar cartas, da lo mismo tener inicialmente cualquier montón de cartas en cualquier mano, la izquierda o la derecha. Los resultados son siempre mezclas de cartas tomadas de ambos montones sin que se pierda el orden entre sí de las cartas de cada montón,

$$X \odot Y = Y \odot X\,.$$

También debería ser claro que, si se parte la baraja en tres montones de cartas y se hace una mezcla con dos de ellos y luego el montón resultante se mezcla con el tercero, se obtendrían las mismas mezclas que si se tomara inicialmente otros montones cualesquiera,

$$(X \odot Y)\odot Z = X \odot (Y \odot Z) = (X \odot Z)\odot Y\,.$$

Definimos ahora el operador

$$Shuffle_{m,n}(W) = (w_1 w_2 \ldots w_m \odot w_{m+1} \ldots w_{m+n})\, w_{m+n+1} \ldots w_k$$

donde k es la longitud de la palabra $W = w_1 w_2 \cdots w_k$, w_i son elementos de la base de V, $m+n \le k$ y $0 \le m, n \le k$. Con esto podemos definir una fórmula <u>*recursiva*</u> para calcular un Shufflé con las

Ecuaciones 4—2

$$X \odot Y = Shuffle_{m,n}(XY),$$

$$Shuffle_{0,n}(W) = W,$$

$$Shuffle_{m,0}(W) = W,$$

$$Shuffle_{m,n}(W) = Shuffle_{m,n-1}(W) + Shuffle_{m-1,n}(W \cdot \prod_{i=m}^{m+n-1} t_i)$$

donde m y n son las longitudes de las palabras X y Y respectivamente.

Ejemplo 4—3:

Calcular

$$ab \odot cd = Shuffle_{2,2}(abcd)$$

$$= Shuffle_{2,1}(abcd) + Shuffle_{1,2}(abcd \cdot t_2 t_3)$$

$$= Shuffle_{2,0}(abcd) + Shuffle_{1,1}(abcd \cdot t_2)$$

$$+ Shuffle_{1,1}(abcd \cdot t_2 t_3) + Shuffle_{0,2}(abcd \cdot t_2 t_3 t_1 t_2)$$

$$= abcd + Shuffle_{1,0}(abcd \cdot t_2) + Shuffle_{0,1}(abcd \cdot t_2 t_1)$$

$$+ Shuffle_{1,0}(abcd \cdot t_2 t_3) + Shuffle_{0,1}(abcd \cdot t_2 t_3 t_1) + abcd \cdot t_2 t_3 t_1 t_2$$

$$= abcd + abcd \cdot t_2 + abcd \cdot t_2 t_1 + abcd \cdot t_2 t_3 + abcd \cdot t_2 t_3 t_1 + abcd \cdot t_2 t_3 t_1 t_2$$

$$= abcd + acbd + cabd + acdb + cadb + cdab$$

que es el mismo resultado que se obtuvo en el Ejemplo 4—2:

4.2.2 *Álgebras Shufflé Cuánticas.*

Hay una pequeña gran diferencia entre el Álgebra Shufflé Clásica y las Álgebras Shufflé Cuánticas.

En el Álgebra Shufflé Clásica definimos el operador $\odot$ como:

$$L_1 \odot L_2 = \sum_{\pi \in S(k,l)} (L_1 L_2)\pi = Shuffle_{k,l}(L_1 L_2)$$

donde $S(k, l)$ es el conjunto formado por todas las permutaciones (k, l)-Shufflé de $k + l$ elementos y $L_1 L_2$ es la concatenación de las listas L_1 y L_2 cuyas longitudes son k y l respectivamente.

Si usamos la definición recursiva del Shufflé Clásico y reemplazamos cada t_i de dicha definición por la correspondiente τ_i (la cual se define como en el capítulo 3.1), obtenemos un Álgebra Shufflé Cuántica.

Un ejemplo de un Shufflé Cuántico:

Ejemplo 4—4:

Dado un operador τ definido con un tensor-(2,2) $\alpha_{x\ y}^{p\ q}$ con

$$\begin{bmatrix} \alpha_{a\,a}^{a\,a} & \alpha_{a\,a}^{a\,b} & \alpha_{a\,a}^{b\,a} & \alpha_{a\,a}^{b\,b} \\ \alpha_{a\,b}^{a\,a} & \alpha_{a\,b}^{a\,b} & \alpha_{a\,b}^{b\,a} & \alpha_{a\,b}^{b\,b} \\ \alpha_{b\,a}^{a\,a} & \alpha_{b\,a}^{a\,b} & \alpha_{b\,a}^{b\,a} & \alpha_{b\,a}^{b\,b} \\ \alpha_{b\,b}^{a\,a} & \alpha_{b\,b}^{a\,b} & \alpha_{b\,b}^{b\,a} & \alpha_{b\,b}^{b\,b} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 2 & -1 & 0 & 5 \\ -1 & 4 & -4 & 0 \\ 0 & -1 & 0 & 2 \end{bmatrix}$$

calcularemos:

$$ab \odot a = Shuffle_{2,1}(aba) = Shuffle_{2,0}(aba) + Shuffle_{1,1}(aba \cdot \tau_2)$$

$$= aba + Shuffle_{1,0}(aba \cdot \tau_2) + Shuffle_{0,1}(aba \cdot \tau_2\tau_1)$$

$$= aba + aba \cdot \tau_2 + aba \cdot \tau_2\tau_1$$

$$= aba + a(-aa + 4ab - 4ba) + a(-aa + 4ab - 4ba)\tau_1$$

$$= aba - aaa + 4aab - 4aba + (-aaa + 4aab - 4aba)\tau_1$$

$$= aba - aaa + 4aab - 4aba - aaa \cdot \tau_1 + 4aab \cdot \tau_1 - 4aba \cdot \tau_1$$

$$= aba - aaa + 4aab - 4aba - (aa + 2ab)a + 4(aa + 2ab)b - 4(2aa - ab + 5bb)a$$

$$= aba - aaa + 4aab - 4aba - aaa - 2aba + 4aab + 8abb - 8aaa + 4aba - 20bba$$

$$= -aba - 10aaa + 8aab + 8abb - 20bba$$

4.3 *Álgebras Simétricas Cuánticas.*

4.3.1 Álgebra Simétrica Cuántica de un tensor-(2,2).

Un Álgebra Simétrica Cuántica es una subálgebra del Álgebra Shufflé Cuántica.

Los elementos de un Álgebra Simétrica Cuántica son los elementos de la base M del espacio V y todos aquellos elementos que pueden ser generados mediante productos Shufflé Cuánticos entre los elementos de esta Álgebra.

Por ejemplo, los generadores de un Álgebra Simétrica Cuántica pueden ser las palabras: $\{a\}$ y $\{b\}$. Dado un operador τ definido mediante un tensor-(2,2) como en el capítulo 3.1 y su correspondiente Álgebra Shufflé Cuántica podemos obtener otro elemento del Álgebra mediante $a \odot a$, $a \odot b$, $b \odot a$ ó $b \odot b$. Otros elementos pueden obtenerse mediante el Shufflé del nuevo elemento encontrado con cualquier otro.

Existen Álgebras Simétricas Cuánticas de dimensión infinita y también de dimensión finita.

Normalmente el conjunto de los elementos del Álgebra Simétrica Cuántica asociada a un Álgebra Shufflé Cuántica son un subconjunto de los elementos de esta última. Cuando más, un Álgebra Simétrica Cuántica podría tener los mismos elementos que el Álgebra Shufflé Cuántica asociada.

4.3.2 YBE y la asociatividad de las Álgebras Shufflé y Simétrica Cuánticas.

Un Álgebra Shufflé Cuántica no es necesariamente conmutativa ni asociativa como puede verse en el siguiente ejemplo:

Ejemplo 4—5:

Dado un operador τ definido con un tensor $\alpha_{x\,y}^{p\,q}$ con

$$\alpha_{a\,b}^{a\,a}=2,\ \alpha_{a\,b}^{a\,b}=-1,\ \alpha_{a\,b}^{b\,a}=0,\ \alpha_{a\,b}^{b\,b}=5$$

$$\alpha_{b\,a}^{a\,a}=-1,\ \alpha_{b\,a}^{a\,b}=4,\ \alpha_{b\,a}^{b\,a}=-4,\ \alpha_{b\,a}^{b\,b}=0$$

calcularemos:

$$a\odot b = Shuffle_{1,1}(ab) = Shuffle_{1,0}(ab) + Shuffle_{0,1}(ab\cdot\tau_1)$$
$$= ab + ab\cdot\tau_1 = ab + 2aa - ab + 5bb$$
$$= 2aa + 5bb$$

ahora:

$$b\odot a = Shuffle_{1,1}(ba) = Shuffle_{1,0}(ba) + Shuffle_{0,1}(ba\cdot\tau_1)$$
$$= ba + ba\cdot\tau_1 = ba - aa + 4ab - 4ba$$
$$= -aa + 4ab - 3ba$$

que claramente no son iguales. Similarmente podemos comprobar que

$(a\odot b)\odot a \neq a\odot(b\odot a)$.

Sin embargo existen Álgebras Shufflé Cuánticas que sí son conmutativas y/o asociativas. En el apéndice 11.3 se pueden ver las condiciones necesarias y suficientes para que un Álgebra Shufflé Cuántica sea conmutativa o asociativa. También se demuestra en el mismo apéndice que las condiciones necesarias para la *asociatividad* son <u>las mismas</u> que las de la *relación de trenza* del operador τ usado.

TEOREMA 9 Un Álgebra Shufflé Cuántica es asociativa si, y sólo si, el operador τ que la define cumple con la relación de trenza.

Prueba:

Sean u, v, y w cualesquiera palabras de longitud 1. Tenemos que

$$(u\odot v)\odot w = (uv + (uv)\tau_1)\odot w = uv\odot w + (uv)\tau_1\odot w =$$
$$= (uvw) + (uvw)\tau_2 + (uvw)\tau_2\tau_1 + (uv)\tau_1\odot w =$$
$$= (uvw)(id + \tau_2 + \tau_2\tau_1) + (uv)\tau_1\odot w$$
$$= (uvw)(id + \tau_2 + \tau_2\tau_1) + (uvw)\tau_1(id + \tau_2 + \tau_2\tau_1)$$
$$= (uvw)(id + \tau_2 + \tau_2\tau_1) + (uvw)(\tau_1 + \tau_1\tau_2 + \tau_1\tau_2\tau_1)$$
$$= (uvw)(id + \tau_2 + \tau_2\tau_1 + \tau_1 + \tau_1\tau_2 + \tau_1\tau_2\tau_1)$$

y por otro lado

$$u \odot (v \odot w) = u \odot (vw + (vw)\tau_1) = u \odot vw + u \odot (vw)\tau_1$$
$$= (uvw) + (uvw)\tau_1 + (uvw)\tau_1\tau_2 + u \odot (vw)\tau_1$$
$$= (uvw)(id + \tau_1 + \tau_1\tau_2) + u \odot (vw)\tau_1$$
$$= (uvw)(id + \tau_1 + \tau_1\tau_2) + (uvw)\tau_2(id + \tau_1 + \tau_1\tau_2)$$
$$= (uvw)(id + \tau_1 + \tau_1\tau_2) + (uvw)(\tau_2 + \tau_2\tau_1 + \tau_2\tau_1\tau_2)$$
$$= (uvw)(id + \tau_1 + \tau_1\tau_2 + \tau_2 + \tau_2\tau_1 + \tau_2\tau_1\tau_2)$$

que acomodando términos y comparando los dos resultados se ve que, cuando ambos son equivalentes, se debe cumplir la relación de trenza para el operador τ. No es necesaria la prueba para palabras de longitud diferente de 1 ya que en toda Álgebra Shufflé Cuántica asociativa están presentes las palabras de longitud 1 y por lo tanto es necesaria la condición ya probada para estas palabras.

Ahora sólo falta probar que cuando el operador τ cumple la relación de trenza entonces el Álgebra Shufflé Cuántica es asociativa.

Denotaremos

$$[a;b] = \tau_{b-1}\tau_{b-2}\ldots\tau_a = \prod_{v=b-1}^{a} \tau_v \text{, y}$$

$$[a;b]_t = t_{b-1}t_{b-2}\ldots t_a = \prod_{v=b-1}^{a} t_v .$$

Esta última expresión se interpreta como la permutación que "mueve" hacia la posición a al elemento en la posición b siendo $a \le b$ recorriendo una posición hacia adelante a los elementos ubicados en las posiciones a hasta la b-1. En el caso cuando $a = b$ se tiene la permutación identidad.

Denotaremos también como $S(k,l,m)$ al conjunto de permutaciones que dejan sin cambio a las primeras k posiciones y hacen un Shufflé entre las siguientes l y m posiciones.

LEMA 1: Si $a \le b < c \le d$ *entonces* $[a;b][c;d] = [c;d][a;b]$.

Debido a la linealidad del operador τ, se tiene que cuando $|i-j|>1$ entonces $\tau_i\tau_j = \tau_j\tau_i$.

Todas las τ_i's que intervienen en $[a;b]$ y las τ_j's que intervienen en $[c;d]$ cumplen con $|i-j|>1$, entonces se puede conmutar toda τ_i con toda τ_j.

LEMA 2: Si $a \le c \le b < d$ *entonces* $[a;b][c;d] = [b+1;d][a;b][c;b+1]$.

En este caso solamente algunas de las τ_i's que intervienen en $[a;b]$ conmutan con algunas de las τ_j's que intervienen en $[c;d]$.

Tenemos entonces que

$$\begin{aligned}
&[a;b][c;d] = (\tau_{b-1}\tau_{b-2}\cdots\tau_c\tau_{c-1}\cdots\tau_a)\cdot(\tau_{d-1}\tau_{d-2}\cdots\tau_b\tau_{b-1}\cdots\tau_c)\\
&= \tau_{b-1}\tau_{b-2}\cdots\tau_c\tau_{c-1}\cdots\tau_a\cdot(\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_b\tau_{b-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_{b-2}\cdots\tau_c\tau_{c-1}\cdots\tau_a\cdot\tau_b\tau_{b-1}\cdots\tau_c\\
&= [b+1;d][a,b][c;b+1]
\end{aligned}$$

que es lo que queríamos demostrar.

LEMA 3: Si $c \le a \le b < d$ y τ cumple la relación de trenza entonces $[a;b][c;d] = [c;d][a+1;b+1]$.

Desarrollando

$$\begin{aligned}
&[a;b][c;d] = (\tau_{b-1}\tau_{b-2}\cdots\tau_a)\cdot(\tau_{d-1}\tau_{d-2}\cdots\tau_b\tau_{b-1}\cdots\tau_a\tau_{a-1}\cdots\tau_c)\\
&= \tau_{b-1}\tau_{b-2}\cdots\tau_a(\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_b\tau_{b-1}\cdots\tau_a\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_{b-2}\cdots\tau_a\tau_b\tau_{b-1}\cdots\tau_a\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_{b-2}\cdots\tau_a(\tau_b)\tau_{b-1}\cdots\tau_a\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}(\tau_b)\tau_{b-2}\tau_{b-3}\cdots\tau_a\tau_{b-1}\cdots\tau_a\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_b\tau_{b-2}\tau_{b-3}\cdots\tau_a(\tau_{b-1})\tau_{b-2}\cdots\tau_a\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_b\tau_{b-2}(\tau_{b-1})\tau_{b-3}\tau_{b-4}\cdots\tau_a\tau_{b-2}\cdots\tau_a\tau_{a-1}\cdots\tau_c\\
&\vdots\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_b\tau_{b-2}\tau_{b-1}\tau_{b-3}\tau_{b-2}\tau_{b-4}\tau_{b-3}\cdots\tau_{a+1}\tau_{a+2}\tau_a\tau_{a+1}\tau_a\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_b\tau_{b-2}\tau_{b-1}\tau_{b-3}\tau_{b-2}\tau_{b-4}\tau_{b-3}\cdots\tau_{a+1}\tau_{a+2}(\tau_a\tau_{a+1}\tau_a)\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_b\tau_{b-2}\tau_{b-1}\tau_{b-3}\tau_{b-2}\tau_{b-4}\tau_{b-3}\cdots\tau_{a+1}\tau_{a+2}(\tau_{a+1}\tau_a\tau_{a+1})\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_b\tau_{b-2}\tau_{b-1}\tau_{b-3}\tau_{b-2}\tau_{b-4}\tau_{b-3}\cdots\tau_{a+3}(\tau_{a+1}\tau_{a+2}\tau_{a+1})\tau_a\tau_{a+1}\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_{b-1}\tau_b\tau_{b-2}\tau_{b-1}\tau_{b-3}\tau_{b-2}\tau_{b-4}\tau_{b-3}\cdots\tau_{a+2}\tau_{a+3}(\tau_{a+2}\tau_{a+1}\tau_{a+2})\tau_a\tau_{a+1}\tau_{a-1}\cdots\tau_c\\
&\vdots\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_b\tau_{b-1}\tau_b\tau_{b-2}\tau_{b-1}\tau_{b-3}\tau_{b-2}\tau_{b-4}\cdots\tau_{a+4}\tau_{a+2}\tau_{a+3}\tau_{a+1}\tau_{a+2}\tau_a\tau_{a+1}\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_b\tau_{b-1}(\tau_b)\tau_{b-2}(\tau_{b-1})\tau_{b-3}\cdots(\tau_{a+3})\tau_{a+1}(\tau_{a+2})\tau_a(\tau_{a+1})\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_b\tau_{b-1}\tau_{b-2}\tau_{b-3}\tau_{b-4}\cdots\tau_{a+2}\tau_{a+1}\tau_a(\tau_b\tau_{b-1}\tau_{b-2}\cdots\tau_{a+4}\tau_{a+3}\tau_{a+2}\tau_{a+1})\tau_{a-1}\cdots\tau_c\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1})\tau_b\tau_{b-1}\tau_{b-2}\tau_{b-3}\tau_{b-4}\cdots\tau_{a+2}\tau_{a+1}\tau_a\tau_{a-1}\cdots\tau_c(\tau_b\tau_{b-1}\tau_{b-2}\cdots\tau_{a+4}\tau_{a+3}\tau_{a+2}\tau_{a+1})\\
&= (\tau_{d-1}\tau_{d-2}\cdots\tau_{b+1}\tau_b\tau_{b-1}\tau_{b-2}\tau_{b-3}\tau_{b-4}\cdots\tau_{a+2}\tau_{a+1}\tau_a\tau_{a-1}\cdots\tau_c)(\tau_b\tau_{b-1}\tau_{b-2}\cdots\tau_{a+4}\tau_{a+3}\tau_{a+2}\tau_{a+1})\\
&= [c;d][a+1;b+1]
\end{aligned}$$

LEMA 4: Si $\pi \in S(k,l,m)$ *se tiene que*

$$\pi(i) = i, 1 < i \leq k,$$
$$k < \pi(k+1) < \pi(k+2) < \ldots < \pi(k+l) \leq k+l+m,$$
$$k < \pi(k+l+1) < \pi(k+l+2) < \ldots < \pi(k+l+m) \leq k+l+m$$

y también que

$$\pi = [\pi(k+l+1); k+l+1]_t [\pi(k+l+2); k+l+2]_t \ldots [\pi(k+l+m); k+l+m]_t = \prod_{\lambda=k+l+1}^{k+l+m} [\pi(\lambda); \lambda]_t$$

Es decir, que la permutación π no modifica las primeras k posiciones, envía las siguientes l posiciones hacia posiciones que van desde $k+1$ hasta $k+l+m$ sin que pierdan el orden entre ellas y envía las siguientes m posiciones hacia posiciones que van desde $k+1$ hasta $k+l+m$ sin que pierdan el orden entre ellas.

El LEMA 4 indica también cómo puede expresarse una permutación $\pi \in S(k,l,m)$ en términos del operador t.

LEMA 5: Sean u, v, w palabras de longitudes k, l, m respectivamente, se tiene que

$$(u \odot v) \odot w = (uvw) \left(\sum_{\pi_1 \in S(0,k,l)} \prod_{\lambda=0+k+1}^{0+k+l} [\pi_1(\lambda); \lambda] \right) \left(\sum_{\pi_2 \in S(0,k+l,m)} \prod_{\lambda=0+k+l+1}^{0+k+l+m} [\pi_2(\lambda); \lambda] \right)$$
$$= (uvw) \left(\sum_{\pi_1 \in S(0,k,l)} \sum_{\pi_2 \in S(0,k+l,m)} \prod_{\lambda=0+k+1}^{0+k+l} [\pi_1(\lambda); \lambda] \prod_{\lambda=0+k+l+1}^{0+k+l+m} [\pi_2(\lambda); \lambda] \right)$$

y también que

$$u \odot (v \odot w) = (uvw) \left(\sum_{\pi_3 \in S(k,l,m)} \prod_{\lambda=k+l+1}^{k+l+m} [\pi_3(\lambda); \lambda] \right) \left(\sum_{\pi_4 \in S(0,k,l+m)} \prod_{\lambda=0+k+1}^{0+k+l+m} [\pi_4(\lambda); \lambda] \right)$$
$$= (uvw) \left(\sum_{\pi_3 \in S(k,l,m)} \sum_{\pi_4 \in S(0,k,l+m)} \prod_{\lambda=k+l+1}^{k+l+m} [\pi_3(\lambda); \lambda] \prod_{\lambda=0+k+1}^{0+k+l+m} [\pi_4(\lambda); \lambda] \right).$$

El LEMA 5 indica como pueden expresarse en términos de τ los posibles Shufflés entre tres palabras u, v y w en ese orden. Estas expresiones se obtienen a partir de la definición del producto Shufflé en la sección 4.2.2 pero usando la notación descrita en ésta sección.

Debemos probar que, siempre que el operador τ cumpla con la relación de trenza, las dos expresiones dadas en el LEMA 5 son equivalentes, es decir que:

Ecuación 4—3

$$\left(\sum_{\pi_1 \in S(0,k,l)} \sum_{\pi_2 \in S(0,k+l,m)} \prod_{\lambda=0+k+1}^{0+k+l} [\pi_1(\lambda);\lambda] \prod_{\lambda=0+k+l+1}^{0+k+l+m} [\pi_2(\lambda);\lambda] \right) = \left(\sum_{\pi_3 \in S(k,l,m)} \sum_{\pi_4 \in S(0,k,l+m)} \prod_{\lambda=k+l+1}^{k+l+m} [\pi_3(\lambda);\lambda] \prod_{\lambda=0+k+1}^{0+k+l+m} [\pi_4(\lambda);\lambda] \right)$$

El número de términos en el lado izquierdo de la Ecuación 4—3 está dado por

$$\binom{k+l}{l}\binom{k+l+m}{m} = \frac{(k+l)!}{k!l!}\frac{(k+l+m)!}{(k+l)!m!} = \frac{(k+l+m)!}{k!l!m!}$$

y en el lado derecho por

$$\binom{l+m}{m}\binom{k+l+m}{l+m} = \frac{(l+m)!}{l!m!}\frac{(k+l+m)!}{k!(l+m)!} = \frac{(k+l+m)!}{k!l!m!}$$

que es el mismo.

Entonces sólo basta con encontrar, para cada término T del lado izquierdo, un término T' equivalente del lado derecho.

Un término T se determina mediante una de las permutaciones $\pi_1 \in S(0,k,l)$ y una de las permutaciones $\pi_2 \in S(0,k+l,m)$. Debemos encontrar un par de permutaciones $\pi_3 \in S(k,l,m)$ y $\pi_4 \in S(0,k,l+m)$ tales que

Ecuación 4—4 $$\prod_{\lambda=0+k+1}^{0+k+l} [\pi_1(\lambda);\lambda] \prod_{\lambda=0+k+l+1}^{0+k+l+m} [\pi_2(\lambda);\lambda] = \prod_{\lambda=k+l+1}^{k+l+m} [\pi_3(\lambda);\lambda] \prod_{\lambda=0+k+1}^{0+k+l+m} [\pi_4(\lambda);\lambda].$$

Usaremos el principio de Inducción Matemática sobre el parámetro m como método para encontrar dichas permutaciones.

Base de la Inducción: Cuando $m = 0$ el segundo factor del lado izquierdo de la Ecuación 4—4 no existe. Lo mismo sucede con el primer factor del lado derecho quedando

$$\prod_{\lambda=0+k+1}^{0+k+l} [\pi_1(\lambda);\lambda] = \prod_{\lambda=0+k+1}^{0+k+l} [\pi_4(\lambda);\lambda]$$

donde $\pi_1 \in S(0,k,l)$ y $\pi_4 \in S(0,k,l)$, que provienen del mismo conjunto, entonces siempre podemos encontrar una permutación π_4 que cumpla con lo requerido.

Suposición Inductiva: Cuando $m < n$, para cada par de permutaciones $\pi_1 \in S(0,k,l)$ y $\pi_2 \in S(0,k+l,m)$ existe un par de permutaciones $\pi_3 \in S(k,l,m)$ y $\pi_4 \in S(0,k,l+m)$ tales que la Ecuación 4—4 se cumple.

El paso Inductivo: A partir de dos permutaciones cualesquiera $\pi_1 \in S(0,k,l)$ y $\pi_2 \in S(0,k+l,n-1)$ construimos, mediante la suposición inductiva, el par de permutaciones $\pi_3 \in S(k,l,n-1)$ y $\pi_4 \in S(0,k,l+n-1)$.

Hasta aquí tenemos, por el LEMA 4, que las secuencias

$$0 < \pi_1(k+1) < \pi_1(k+2) < \ldots < \pi_1(k+l) \leq k+l,$$

$$0 < \pi_2(k+l+1) < \pi_2(k+l+2) < \ldots < \pi_2(k+l+n-1) \leq k+l+n-1$$

y

$$k < \pi_3(k+l+1) < \pi_3(k+l+2) < \ldots < \pi_3(k+l+n-1) \leq k+l+n-1,$$

$$0 < \pi_4(k+1) < \pi_4(k+2) < \ldots < \pi_4(k+l+n-1) \leq k+l+n-1$$

describen las permutaciones $\pi_1, \pi_2, \pi_3, \pi_4$ respectivamente.

Para $m = n$ construiremos las permutaciones $\pi_3' \in S(k,l,n)$ y $\pi_4' \in S(0,k,l+n)$ a partir de las encontradas mediante la suposición inductiva. Con $m = n$ estamos agregando una letra al final de la palabra w y $\pi_2(k+l+n)$ es la posición que va a tener esa letra en el término T.

Modificaremos las secuencias dadas arriba para π_3 y π_4 para obtener secuencias que describan permutaciones $\pi_3' \in S(k,l,n)$ y $\pi_4' \in S(0,k,l+n)$ y que además se cumpla la Ecuación 4—4 de la siguiente manera:

Cuando $\pi_2(k+l+n) = k+l+n$ tenemos que, para ese término, la última letra de la palabra w queda en ese mismo lugar. Entonces basta con que las permutaciones π_3' y π_4' hagan lo mismo con esa letra para que se cumpla la Ecuación 4—4, es decir, que las secuencias que describirían a π_3' y π_4' serían:

$$k < \pi_3(k+l+1) < \pi_3(k+l+2) < \ldots < \pi_3(k+l+n-1) < \pi_3'(k+l+n) = k+l+n,$$

$$0 < \pi_4(k+1) < \pi_4(k+2) < \ldots < \pi_4(k+l+n-1) < \pi_4'(k+l+n) = k+l+n$$

que son las mismas secuencias obtenidas mediante la suposición inductiva pero agregando al final un elemento igual a $k + l + n$. En otras palabras hacemos que

$\pi_3'(i) = \pi_3(i)$, para $k+l < i < k+l+n$ y $\pi_3'(k+l+n) = k+l+n$;

$\pi_4'(i) = \pi_4(i)$, para $k < i < k+l+n$ y $\pi_4'(k+l+n) = k+l+n$;

Cuando $\pi_2(k+l+n) < k+l+n$ debemos hacer que:

$\pi_4'(\pi_3'(k+l+n)) = \pi_2(k+l+n)$.

En la secuencia de la suposición inductiva que describe a π_4, insertamos el número $\pi_2(k+l+n)$ en la posición $q = \pi_3'(k+l+n)$ de tal manera que se cumpla:

$$0 < \pi_4(k+1) < \ldots < \pi_4(q-1) < \pi_2(k+l+n) \leq \pi_4(q) < \ldots < \pi_4(k+l+n-1) < k+l+n$$

Como no puede haber dos elementos iguales, sumamos uno a cada elemento posterior al recién insertado quedando:

$$0<\pi_4(k+1)<\ldots<\pi_4(q-1)<\pi_2(k+l+n)<\pi_4(q)+1<\ldots<\pi_4(k+l+n-1)+1\leq k+l+n$$

es decir, que para $i<q$, $\pi_4'(i)=\pi_4(i)$; $\pi_4'(q)=\pi_2(k+l+n)$; y para $i>q$ $\pi_4'(i)=\pi_4(i-1)+1$.

La secuencia para π_3' sería simplemente

$$k<\pi_3(k+l+1)<\pi_3(k+l+2)<\ldots<\pi_3(k+l+n-1)<\pi_3'(k+l+n)<k+l+n$$

es decir, que $\pi_3'(i)=\pi_3(i)$ para $k+l<i<k+l+n$, y $\pi_3'(k+l+n)=q$.

Queda así demostrado que siempre es posible construir permutaciones $\pi_3'\in S(k,l,n)$ y $\pi_4'\in S(0,k,l+n)$ que satisfagan la Ecuación 4—4 a partir de dos permutaciones $\pi_3\in S(k,l,n-1)$ y $\pi_4\in S(0,k,l+n-1)$ que ya satisfacen la misma ecuación.

4.4 *Álgebras de Hopf.*

Definimos una ***Coálgebra*** como un espacio lineal C con un mapa lineal

$$\Delta: C\rightarrow C\otimes C$$

llamado ***coproducto***.

Ejemplo 4—6:

Consideramos n^2 elementos denominados T_i^j como variables formales. En el espacio de dimensión n^2 generado por las T_i^j se puede definir el coproducto siguiente

$$\Delta(T_i^j)=\sum_k T_k^j\otimes T_i^k\ .$$

Cuando $n = 2$ tenemos

$$\Delta(T_1^1)=T_1^1\otimes T_1^1+T_2^1\otimes T_1^2,$$
$$\Delta(T_2^1)=T_1^1\otimes T_2^1+T_2^1\otimes T_2^2,$$
$$\Delta(T_1^2)=T_1^2\otimes T_1^1+T_2^2\otimes T_1^2,$$
$$\Delta(T_2^2)=T_1^2\otimes T_2^1+T_2^2\otimes T_2^2.$$

Ejemplo 4—7:

Si $T_1^2=0$ tenemos solo 3 elementos T_1^1, T_2^1, T_2^2 entonces

$$\Delta(T_1^1) = T_1^1 \otimes T_1^1,$$
$$\Delta(T_2^1) = T_1^1 \otimes T_2^1 + T_2^1 \otimes T_2^2,$$
$$\Delta(T_1^2) = 0,$$
$$\Delta(T_2^2) = T_2^2 \otimes T_2^2.$$

Ejemplo 4—8:

Si $T_1^2 = T_2^1 = 0$ tenemos que

$$\Delta(T_1^1) = T_1^1 \otimes T_1^1,$$
$$\Delta(T_2^1) = 0,$$
$$\Delta(T_1^2) = 0,$$
$$\Delta(T_2^2) = T_2^2 \otimes T_2^2.$$

Sea G un grupo (por ejemplo el de permutaciones con $n!$ elementos). Consideramos el espacio lineal kG generado por G como base, es decir, los elementos del espacio kG tienen la forma

$$\sum_{g \in G} \alpha_g g .$$

Los productos en el ***Álgebra de Grupo*** kG son de la forma

$$\left(\sum_g \alpha_g g \right) \left(\sum_h \alpha_h h \right) = \sum_{g,h} \alpha_g \alpha_h gh .$$

Si definimos el coproducto de g como

$$\Delta(g) = g \otimes g$$

en el Álgebra de Grupo, y se cumple

$$\Delta\left(\sum \alpha_g g \right) = \sum \alpha_g g \otimes g ,$$

entonces kG va a tener la estructura de una Coálgebra.

Un elemento g de cualquier coálgebra se dice que es de ***tipo grupo*** si

$$\Delta(g) = g \otimes g .$$

Vemos entonces que la Coálgebra kG está generada por elementos de tipo grupo.

En el Ejemplo 4—7: y en el Ejemplo 4—8: T_1^1 y T_2^2 son de tipo grupo.

Un elemento a en una Coálgebra se llama ***primitivo*** si tiene la propiedad

$$\Delta(a) = g \otimes a + a \otimes h$$

donde g y h son de tipo grupo.

En el Ejemplo 4—7: T_2^1 es primitivo.

Definimos ahora la counidad en una Coálgebra:

Al mapa

$$\varepsilon : C \to k$$

se le llama ***counidad*** en una Coálgebra si para cualquier elemento c se cumple

$$c = \sum c^{(1)} \varepsilon(c^{(2)}) = \sum \varepsilon(c^{(1)}) c^{(2)}$$

donde

$$\Delta(c) = \sum c^{(1)} \otimes c^{(2)} .$$

Para definir una counidad en el Ejemplo 4—6: vemos que ésta debe cumplir con

$$T_1^1 = T_1^1 \varepsilon(T_1^1) + T_2^1 \varepsilon(T_1^2)$$
$$T_1^1 = \varepsilon(T_1^1) T_1^1 + \varepsilon(T_2^1) T_1^2$$

lo cual se logra solo si

$$\varepsilon(T_1^1) = 1,$$
$$\varepsilon(T_1^2) = 0,$$
$$\varepsilon(T_2^1) = 0$$

También debería cumplir con

$$T_2^1 = T_1^1 \varepsilon(T_2^1) + T_2^1 \varepsilon(T_2^2)$$
$$T_2^1 = \varepsilon(T_1^1) T_2^1 + \varepsilon(T_2^1) T_2^2$$

de donde concluimos que

$$\varepsilon(T_1^1) = 1,$$
$$\varepsilon(T_1^2) = 0,$$
$$\varepsilon(T_2^1) = 0,$$
$$\varepsilon(T_2^2) = 1$$

Con este mapa se verifica fácilmente que también se cumplen

$$T_1^2 = T_1^2 \varepsilon(T_1^1) + T_2^2 \varepsilon(T_1^2)$$
$$T_1^2 = \varepsilon(T_1^2) T_1^1 + \varepsilon(T_2^2) T_1^2$$
$$T_2^2 = T_1^2 \varepsilon(T_2^1) + T_2^2 \varepsilon(T_2^2)$$
$$T_2^2 = \varepsilon(T_1^2) T_2^1 + \varepsilon(T_2^2) T_2^2$$

En general se cumple siempre que, si g es de tipo grupo y diferente de cero, entonces

$\Delta(g) = g \otimes g$ y entonces

$g = \varepsilon(g)g$ por lo tanto

$\varepsilon(g) = 1$

y si a es primitivo entonces

$\Delta(a) = g \otimes a + a \otimes h$ y por lo tanto

$a = \varepsilon(g) \cdot a + \varepsilon(a) \cdot h$ de donde

$0 = \varepsilon(a) \cdot h$ y por lo tanto

$\varepsilon(a) = 0$.

Una Coálgebra es ***Coasociativa*** si

$\Delta(id \otimes \Delta) = \Delta(\Delta \otimes id)$

es decir que

$$C \xrightarrow{\Delta} C \otimes C \xrightarrow{id \otimes \Delta} C \otimes (C \otimes C) = C \xrightarrow{\Delta} C \otimes C \xrightarrow{\Delta \otimes id} (C \otimes C) \otimes C$$

o también equivalentemente

$$\sum c^{(1)} \otimes c^{(2)^{(1)}} \otimes c^{(2)^{(2)}} =$$

$$\sum c^{(1)^{(1)}} \otimes c^{(1)^{(2)}} \otimes c^{(2)} =$$

$$\sum c^{(1)} \otimes c^{(2)} \otimes c^{(3)}.$$

Una ***Subcoálgebra*** es una Coálgebra cuyos elementos forman parte de otra Coálgebra.

Una ***Coálgebra mínima*** es una Coálgebra que no tiene Subcoálgebras.

Una ***Coálgebra de dimensión finita*** es una Coálgebra C cuyo espacio C tiene dimensión finita.

Una ***Coálgebra Puntada*** es una Coálgebra en la que todas sus Subcoálgebras mínimas son de dimensión 1.

Ejemplo 4—9:

Definimos el coproducto en el Álgebra Shufflé como

$$\Delta(u) = \sum_{vw=u} v \otimes w$$

donde u, v y w son palabras y vw es la concatenación de v con w.

Definimos la counidad como

$\varepsilon(\varnothing) = 1$

$\varepsilon(u) = 0$

donde $\varnothing$ es la palabra vacía y u es cualquier otra palabra.

Esta coálgebra tiene las siguientes propiedades:

- Es coasociativa.
- Es puntada.
- La única Subcoálgebra mínima es la que se genera con la palabra vacía.

Una ***Biálgebra*** es un álgebra y una coálgebra tales que cumplen con:

$$\Delta(ab) = \Delta(a) \cdot \Delta(b)$$
$$\varepsilon(ab) = \varepsilon(a)\varepsilon(b)$$

donde

$$(a \otimes b) \cdot (c \otimes d) = ac \otimes bd$$

pero si en lugar de lo anterior se tiene que

$$(a \otimes b) \cdot (c \otimes d) = a\tau(b \otimes c)d$$

entonces es ***Biálgebra Trenzada***.

Una ***Antípoda*** es un mapa

$$S : A \to A$$

que cumple

$$\varepsilon(a) \cdot 1 = \sum a^{(1)} \cdot S(a^{(2)}) = \sum S(a^{(1)}) \cdot a^{(2)},$$
$$\Delta(a) = \sum a^{(1)} \otimes a^{(2)}.$$

Un ***Álgebra de Hopf*** es una Biálgebra con Antípoda.

De acuerdo con Oziewicz 1995 [R 16] si τ cumple con la relación de trenza entonces

$$Shuffle_{\tau} \in \hom(H, H),$$

es decir, que el Shufflé es un homomorfismo entre dos Álgebras de Hopf trenzadas.

Capítulo 5 **Cálculos Diferenciales No Conmutativos.**

5.1 ***Definición.***

Del Cálculo Diferencial "Clásico" sabemos que:

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy$$

Definimos para los Cálculos Diferenciales No Conmutativos *de dos variables* las siguientes

Ecuaciones 5—1

$$df = dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y}$$
$$d(fg) = df \cdot g + f \cdot dg$$

donde f y g son funciones cualesquiera.

Definimos ahora las

Reglas de conmutación 5—2

$$v \cdot dx = dx \cdot A_1^1(v) + dy \cdot A_1^2(v)$$
$$v \cdot dy = dx \cdot A_2^1(v) + dy \cdot A_2^2(v)$$

De acuerdo con las Ecuaciones 5—1 tenemos

$$d(fg) = dx\frac{\partial(fg)}{\partial x} + dy\frac{\partial(fg)}{\partial y} = df \cdot g + f \cdot dg$$

$$= (dx\frac{\partial f}{\partial x} + dy\frac{\partial f}{\partial y}) \cdot g + f \cdot (dx\frac{\partial g}{\partial x} + dy\frac{\partial g}{\partial y})$$

$$= dx\frac{\partial f}{\partial x} \cdot g + dy\frac{\partial f}{\partial y} \cdot g + (f \cdot dx)\frac{\partial g}{\partial x} + (f \cdot dy)\frac{\partial g}{\partial y}$$

que usando las Reglas de conmutación 5—2:

$$= dx\frac{\partial f}{\partial x} \cdot g + dy\frac{\partial f}{\partial y} \cdot g + (dx \cdot A_1^1(f) + dy \cdot A_1^2(f))\frac{\partial g}{\partial x} + (dx \cdot A_2^1(f) + dy \cdot A_2^2(f))\frac{\partial g}{\partial y}$$

$$= dx\frac{\partial f}{\partial x} \cdot g + dy\frac{\partial f}{\partial y} \cdot g + dx \cdot A_1^1(f)\frac{\partial g}{\partial x} + dy \cdot A_1^2(f)\frac{\partial g}{\partial x} + dx \cdot A_2^1(f)\frac{\partial g}{\partial y} + dy \cdot A_2^2(f)\frac{\partial g}{\partial y}$$

$$= dx \cdot (\frac{\partial f}{\partial x} \cdot g + A_1^1(f)\frac{\partial g}{\partial x} + A_2^1(f)\frac{\partial g}{\partial y}) + dy \cdot (\frac{\partial f}{\partial y} \cdot g + A_1^2(f)\frac{\partial g}{\partial x} + A_2^2(f)\frac{\partial g}{\partial y})$$

de donde obtenemos las

Ecuaciones 5—3

$$\frac{\partial(fg)}{\partial x} = \frac{\partial f}{\partial x}\cdot g + A_1^1(f)\frac{\partial g}{\partial x} + A_2^1(f)\frac{\partial g}{\partial y}$$
$$\frac{\partial(fg)}{\partial y} = \frac{\partial f}{\partial y}\cdot g + A_1^2(f)\frac{\partial g}{\partial x} + A_2^2(f)\frac{\partial g}{\partial y}$$

que *definen* los Cálculos Diferenciales No Conmutativos.

Usando

$$(u\cdot v)\cdot dx = u\cdot(v\cdot dx)$$

y desarrollando ambos lados según las Reglas de conmutación 5—2

$$dx\cdot A_1^1(u\cdot v) + dy\cdot A_1^2(u\cdot v) = u\cdot(dx\cdot A_1^1(v) + dy\cdot A_1^2(v))$$
$$= (u\cdot dx)\cdot A_1^1(v) + (u\cdot dy)\cdot A_1^2(v)$$
$$= (dx\cdot A_1^1(u) + dy\cdot A_1^2(u))\cdot A_1^1(v) + (dx\cdot A_2^1(u) + dy\cdot A_2^2(u))\cdot A_1^2(v)$$
$$= dx\cdot A_1^1(u)\cdot A_1^1(v) + dy\cdot A_1^2(u)\cdot A_1^1(v) + dx\cdot A_2^1(u)\cdot A_1^2(v) + dy\cdot A_2^2(u)\cdot A_1^2(v)$$
$$= dx\cdot(A_1^1(u)\cdot A_1^1(v) + A_2^1(u)\cdot A_1^2(v)) + dy\cdot(A_1^2(u)\cdot A_1^1(v) + A_2^2(u)\cdot A_1^2(v))$$

de donde se obtienen las

Ecuaciones 5—4

$$A_1^1(u\cdot v) = A_1^1(u)\cdot A_1^1(v) + A_2^1(u)\cdot A_1^2(v)$$
$$A_1^2(u\cdot v) = A_1^2(u)\cdot A_1^1(v) + A_2^2(u)\cdot A_1^2(v)$$

De manera similar, usando

$$(u\cdot v)\cdot dy = u\cdot(v\cdot dy)$$

y desarrollando ambos lados según las Reglas de conmutación 5—2

$$dx\cdot A_2^1(u\cdot v) + dy\cdot A_2^2(u\cdot v) = u\cdot(dx\cdot A_2^1(v) + dy\cdot A_2^2(v))$$
$$= (u\cdot dx)\cdot A_2^1(v) + (u\cdot dy)\cdot A_2^2(v)$$
$$= (dx\cdot A_1^1(u) + dy\cdot A_1^2(u))\cdot A_2^1(v) + (dx\cdot A_2^1(u) + dy\cdot A_2^2(u))\cdot A_2^2(v)$$
$$= dx\cdot A_1^1(u)\cdot A_2^1(v) + dy\cdot A_1^2(u)\cdot A_2^1(v) + dx\cdot A_2^1(u)\cdot A_2^2(v) + dy\cdot A_2^2(u)\cdot A_2^2(v)$$
$$= dx\cdot(A_1^1(u)\cdot A_2^1(v) + A_2^1(u)\cdot A_2^2(v)) + dy\cdot(A_1^2(u)\cdot A_2^1(v) + A_2^2(u)\cdot A_2^2(v))$$

de donde se obtienen las

Ecuaciones 5—5

$$A_2^1(u\cdot v) = A_1^1(u)\cdot A_2^1(v) + A_2^1(u)\cdot A_2^2(v)$$
$$A_2^2(u\cdot v) = A_1^2(u)\cdot A_2^1(v) + A_2^2(u)\cdot A_2^2(v)$$

Desarrollando xdx, ydx, xdy y ydy con las Reglas de conmutación 5—2 se obtiene:

$$x \cdot dx = dx \cdot A_1^1(x) + dy \cdot A_1^2(x)$$
$$x \cdot dy = dx \cdot A_2^1(x) + dy \cdot A_2^2(x)$$
$$y \cdot dx = dx \cdot A_1^1(y) + dy \cdot A_1^2(y)$$
$$y \cdot dy = dx \cdot A_2^1(y) + dy \cdot A_2^2(y)$$

Cuando las A_i^j se definen como los operadores lineales dados por las

Ecuaciones 5—6

$$A_i^j(x) = \alpha_{1i}^{j1} x + \alpha_{1i}^{j2} y$$
$$A_i^j(y) = \alpha_{2i}^{j1} x + \alpha_{2i}^{j2} y$$

lo anterior se convierte en:

$$x \cdot dx = dx \cdot (\alpha_{11}^{11} x + \alpha_{11}^{12} y) + dy \cdot (\alpha_{11}^{21} x + \alpha_{11}^{22} y)$$
$$x \cdot dy = dx \cdot (\alpha_{12}^{11} x + \alpha_{12}^{12} y) + dy \cdot (\alpha_{12}^{21} x + \alpha_{12}^{22} y)$$
$$y \cdot dx = dx \cdot (\alpha_{21}^{11} x + \alpha_{21}^{12} y) + dy \cdot (\alpha_{21}^{21} x + \alpha_{21}^{22} y)$$
$$y \cdot dy = dx \cdot (\alpha_{22}^{11} x + \alpha_{22}^{12} y) + dy \cdot (\alpha_{22}^{21} x + \alpha_{22}^{22} y)$$

En este caso, cada Cálculo Diferencial No Conmutativo *se define* mediante el tensor-(2,2) α_{kl}^{ij} con $i, j, k, l \in \{1,2,...,n\}$ donde n es el numero de variables en el Cálculo.

Para calcular una derivada parcial en un Cálculo Diferencial No Conmutativo se emplean las Ecuaciones 5—3 seguidas de las Ecuaciones 5—4 y las Ecuaciones 5—5 repetidamente hasta que se puedan aplicar las Ecuaciones 5—6.

Ejemplo 5—1:

Calcular

$$\frac{\partial}{\partial x}(x^2) = \frac{\partial}{\partial x}(x \cdot x) = \frac{\partial x}{\partial x} \cdot x + A_1^1(x) \cdot \frac{\partial x}{\partial x} + A_2^1(x) \cdot \frac{\partial x}{\partial y}$$

$$= x + A_1^1(x)$$

$$= x + \alpha_{11}^{11} x + \alpha_{11}^{12} y$$

$$= (1 + \alpha_{11}^{11}) x + \alpha_{11}^{12} y$$

Ejemplo 5—2:

Calcular

$$\frac{\partial}{\partial x}(x^2 y) = \frac{\partial}{\partial x}(x^2) \cdot y + A_1^1(x^2) \frac{\partial y}{\partial x} + A_2^1(x^2) \frac{\partial y}{\partial y}$$

$$= \left(\frac{\partial x}{\partial x} x + A_1^1(x) \frac{\partial x}{\partial x} + A_2^1(x) \frac{\partial x}{\partial y}\right) \cdot y + (A_1^1(x) A_2^1(x) + A_2^1(x) A_2^2(x))$$

$$= (x + A_1^1(x)) \cdot y + (A_1^1(x) A_2^1(x) + A_2^1(x) A_2^2(x))$$

$$= (x + \alpha_{11}^{11} x + \alpha_{11}^{12} y) \cdot y + (\alpha_{11}^{11} x + \alpha_{11}^{12} y)(\alpha_{12}^{11} x + \alpha_{12}^{12} y) + (\alpha_{12}^{11} x + \alpha_{12}^{12} y)(\alpha_{12}^{21} x + \alpha_{12}^{22} y)$$

$$= (\alpha_{11}^{11}\alpha_{12}^{11} + \alpha_{12}^{11}\alpha_{12}^{21})xx + (1 + \alpha_{11}^{11} + \alpha_{11}^{11}\alpha_{12}^{12} + \alpha_{12}^{11}\alpha_{12}^{22})xy$$

$$+(\alpha_{11}^{12}\alpha_{12}^{11} + \alpha_{12}^{12}\alpha_{12}^{21})yx + (\alpha_{11}^{12} + \alpha_{11}^{12}\alpha_{12}^{12} + \alpha_{12}^{12}\alpha_{12}^{22})yy$$

o también:

$$\frac{\partial}{\partial x}(x^2 y) = \frac{\partial x}{\partial x} \cdot xy + A_1^1(x)\frac{\partial}{\partial x}(xy) + A_2^1(x)\frac{\partial}{\partial y}(xy)$$

$$= xy + A_1^1(x)(\frac{\partial x}{\partial x} y + A_1^1(x)\frac{\partial y}{\partial x} + A_2^1(x)\frac{\partial y}{\partial y}) + A_2^1(x)(\frac{\partial x}{\partial y} y + A_1^2(x)\frac{\partial y}{\partial x} + A_2^2(x)\frac{\partial y}{\partial y})$$

$$= xy + A_1^1(x)(y + A_2^1(x)) + A_2^1(x) A_2^2(x)$$

$$= xy + (\alpha_{11}^{11} x + \alpha_{11}^{12} y)(y + \alpha_{12}^{11} x + \alpha_{12}^{12} y) + (\alpha_{12}^{11} x + \alpha_{12}^{12} y)(\alpha_{12}^{21} x + \alpha_{12}^{22} y)$$

$$= (\alpha_{11}^{11}\alpha_{12}^{11} + \alpha_{12}^{11}\alpha_{12}^{21})xx + (1 + \alpha_{11}^{11} + \alpha_{11}^{11}\alpha_{12}^{12} + \alpha_{12}^{11}\alpha_{12}^{22})xy$$

$$+(\alpha_{11}^{12}\alpha_{12}^{11} + \alpha_{12}^{12}\alpha_{12}^{21})yx + (\alpha_{11}^{12} + \alpha_{11}^{12}\alpha_{12}^{12} + \alpha_{12}^{12}\alpha_{12}^{22})yy$$

5.2 ***Relación entre Cálculos Diferenciales No Conmutativos y las Álgebras Simétricas Cuánticas.***

Si definimos un Álgebra Simétrica Cuántica de n generadores y un Cálculo Diferencial No Conmutativo de n variables con un mismo tensor-(2,2) α_{ki}^{jl} diremos que ambos son correspondientes entre sí.

Sea F un Álgebra Libre y $f(x_1, x_2, \ldots, x_n) \in F$ un polinomio homogéneo de grado n entonces $f(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) \in SimQ_\alpha$ es cero en el Álgebra Simétrica Cuántica correspondiente a un Cálculo Diferencial No Conmutativo si, y sólo si, todas sus derivadas de orden n son cero en ese Cálculo. Es decir,

$$f(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) = 0 \Leftrightarrow \frac{\partial^n f}{\partial x_{i_1} \partial x_{i_2} \ldots \partial x_{i_n}} = 0, \quad 1 \le i_1, i_2, \ldots, i_n \le n.$$

Aquí las $\overline{x}_i$ denotan a los mismos elementos x_i considerados como elementos del Álgebra Shuffé Cuántica, es decir que, si por ejemplo:

$$f(x_1, x_2) = x_1 x_2$$

entonces

$$f(\overline{x}_1, \overline{x}_2) = \overline{x}_1 \overline{x}_2 = x_1 \odot x_2 .$$

Ejemplo 5—3:

Dado el siguiente polinomio en un Cálculo de una variable

$$f(x) = \beta x^2$$

calcular:

$$\frac{\partial}{\partial x} f(x) = \beta \frac{\partial}{\partial x} x^2 = \beta \frac{\partial}{\partial x}(x \cdot x) = \beta(\frac{\partial x}{\partial x} \cdot x + \alpha x \frac{\partial x}{\partial x}) = \beta(1+\alpha)x,$$

$$\frac{\partial^2}{\partial x^2} f(x) = \beta(1+\alpha)\frac{\partial x}{\partial x} = \beta(1+\alpha).$$

Entonces $f(\overline{x}_1) \in SimQ_\alpha$ es cero en si, y sólo si, $\beta(1+\alpha) = 0$.

Nótese que, en el ejemplo anterior, α no tiene subíndices ni superíndices. Esto se debe a que sólo existe una variable en el Cálculo así que la única relación que se debe definir en este Cálculo es:

$$x \cdot dx = \alpha \cdot dx \cdot x$$

y, para el Álgebra Simétrica Cuántica correspondiente, la relación:

$$(xx)\tau = \alpha \cdot xx.$$

Dos polinomios homogéneos $f_1, f_2 \in F$ de grado n son iguales en $SimQ_\alpha$ si, y sólo si, todas las derivadas de orden n de la diferencia $f_1 - f_2$ son cero en el Cálculo Diferencial No Conmutativo correspondiente a $SimQ_\alpha$.

Ejemplo 5—4:

Determinar las condiciones necesarias para que $f_1(x,y) = xy$ sea igual a $f_2(x,y) = yx$.

Como los polinomios son de grado 2 se deben calcular todas sus segundas derivadas que son cuatro:

$$\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial x \partial y}, \frac{\partial^2 f}{\partial y \partial x}, \frac{\partial^2 f}{\partial y^2}$$

así, para f_1:

$$\frac{\partial}{\partial x} xy = \frac{\partial x}{\partial x} y + A_1^1(x)\frac{\partial y}{\partial x} + A_2^1(x)\frac{\partial y}{\partial y} = y + (\alpha_{12}^{11} x + \alpha_{12}^{12} y) = \alpha_{12}^{11} x + (1 + \alpha_{12}^{12}) y,$$

$$\frac{\partial}{\partial y} xy = \frac{\partial x}{\partial y} y + A_1^2(x)\frac{\partial y}{\partial x} + A_2^2(x)\frac{\partial y}{\partial y} = \alpha_{12}^{21} x + \alpha_{12}^{22} y,$$

$$\frac{\partial^2}{\partial x^2} xy = \frac{\partial}{\partial x}(\alpha_{12}^{11} x + (1 + \alpha_{12}^{12}) y) = \alpha_{12}^{11},$$

$$\frac{\partial^2}{\partial x \partial y} xy = \frac{\partial}{\partial y}(\alpha_{12}^{11} x + (1 + \alpha_{12}^{12}) y) = (1 + \alpha_{12}^{12}),$$

$$\frac{\partial^2}{\partial y \partial x} xy = \frac{\partial}{\partial x}(\alpha_{12}^{21} x + \alpha_{12}^{22} y) = \alpha_{12}^{21},$$

$$\frac{\partial^2}{\partial y^2} xy = \frac{\partial}{\partial y}(\alpha_{12}^{21} x + \alpha_{12}^{22} y) = \alpha_{12}^{22};$$

para f_2:

$$\frac{\partial}{\partial x} yx = \frac{\partial y}{\partial x} x + A_1^1(y)\frac{\partial x}{\partial x} + A_2^1(y)\frac{\partial x}{\partial y} = \alpha_{21}^{11} x + \alpha_{21}^{12} y,$$

$$\frac{\partial}{\partial y} yx = \frac{\partial y}{\partial y} x + A_1^2(y)\frac{\partial x}{\partial x} + A_2^2(y)\frac{\partial x}{\partial y} = x + (\alpha_{21}^{21} x + \alpha_{21}^{22} y) = (1 + \alpha_{21}^{21}) x + \alpha_{21}^{22} y,$$

$$\frac{\partial^2}{\partial x^2} yx = \frac{\partial}{\partial x}(\alpha_{21}^{11} x + \alpha_{21}^{12} y) = \alpha_{21}^{11},$$

$$\frac{\partial^2}{\partial x \partial y} yx = \frac{\partial}{\partial y}(\alpha_{21}^{11} x + \alpha_{21}^{12} y) = \alpha_{21}^{12},$$

$$\frac{\partial^2}{\partial y \partial x} yx = \frac{\partial}{\partial x}((1 + \alpha_{21}^{21}) x + \alpha_{21}^{22} y) = (1 + \alpha_{21}^{21}),$$

$$\frac{\partial^2}{\partial y^2} yx = \frac{\partial}{\partial y}((1 + \alpha_{21}^{21}) x + \alpha_{21}^{22} y) = \alpha_{21}^{22}.$$

Entonces para que $xy = yx$ se deben cumplir la relaciones siguientes:

$$\alpha_{12}^{11} = \alpha_{21}^{11},$$

$$1 + \alpha_{12}^{12} = \alpha_{21}^{12},$$

$$\alpha_{12}^{21} = 1 + \alpha_{21}^{21},$$

$$\alpha_{12}^{22} = \alpha_{21}^{22}$$

que llamamos las condiciones de *conmutatividad*.

Capítulo 6 **Algunas Soluciones de YBE.**

En este capítulo se presentan algunas de las soluciones generales de YBE conocidas hasta ahora y su relación con las Álgebras de Hopf.

Dada una solución R podemos encontrar otra solución R' mediante

$$R' = k(Q \otimes Q)R(Q \otimes Q)^{-1}$$

siendo k algún escalar, Q una matriz de $N \times N$ elementos no singular y N la dimensión del tensor R.

6.1 ***Soluciones en forma de tensor-(1,1).***

Las soluciones en forma de tensor-(1,1) son aquellas en las que las α_{ki}^{jl} son multiplicadas por el operador de Kroneker

$$\delta_m^n = \begin{cases} 0 & si \quad m \neq n \\ 1 & si \quad m = n \end{cases}$$

de la siguiente manera:

Ecuaciones 6—1 $$\alpha_{ki}^{jl} = \alpha_k^j \delta_i^j \delta_k^l$$

que tiene la forma de un tensor-(1,1), lo que significa que muchas de las α_{ki}^{jl} se hacen cero, pero α_k^j no es un tensor-(1,1) ya que las Ecuaciones 6—1 no son invariantes bajo cambio de base.

En estas condiciones YBE en su forma coordenada (Ecuaciones 3—2) se convierte en:

$$\sum_{p,q,r \in M} \alpha_{q\ p}^{p\ q} \alpha_{qr}^{rq} \alpha_{pr}^{r\ p} = \sum_{p,q,r \in M} \alpha_{q\ p}^{p\ q} \alpha_{r\ p}^{pr} \alpha_{r\ q}^{qr}$$

la cual, al hacer una rotación en los nombres de los índices del lado derecho queda como

$$\sum_{p,q,r \in M} \alpha_{q\ p}^{p\ q} \alpha_{qr}^{rq} \alpha_{pr}^{r\ p} = \sum_{r,p,q \in M} \alpha_{pr}^{r\ p} \alpha_{qr}^{r\ q} \alpha_{q\ p}^{pq}$$

que claramente se cumple para cualesquiera valores de las α_{ki}^{jl}. Así pues, cualquier matriz α_k^j define una solución de YBE por medio de las Ecuaciones 6—1.

6.2 ***Soluciones en forma de tensor-(1,2).***

Las soluciones en forma de tensor-(1,2) son aquellas de la forma

$\alpha_{ki}^{jl} = \alpha_{ki}^{l}\delta_{i}^{j}$.

Las Ecuaciones 3—2 quedan entonces como:

$$\sum_{q\in M} \alpha_{sv}^{vq}\alpha_{qn}^{nm}\alpha_{vn}^{nw} = \sum_{r\in M} \alpha_{vn}^{nw}\alpha_{sn}^{nr}\alpha_{rw}^{wm}$$

de donde

$$\alpha_{vn}^{w}\left(\sum_{q\in M} \alpha_{sv}^{q}\alpha_{qn}^{m}\right) = \alpha_{vn}^{w}\left(\sum_{r\in M} \alpha_{sn}^{r}\alpha_{rw}^{m}\right)$$

si se define

$$B_v = \begin{pmatrix} \alpha_{1v}^{1} & \cdots & \alpha_{iv}^{1} \\ \vdots & \ddots & \vdots \\ \alpha_{1v}^{i} & \cdots & \alpha_{iv}^{i} \end{pmatrix}$$

se obtiene

$$\alpha_{vn}^{w}(B_v B_n) = \alpha_{vn}^{w}(B_n B_w)$$

ó también

Ecuaciones 6—2

$$\alpha_{vn}^{w}(B_v B_n - B_n B_w) = 0$$

Las Ecuaciones 6—2 tienen dos posibles soluciones:

$\alpha_{vn}^{w} = 0$ y $B_v B_n - B_n B_w = 0$.

6.3 *Teorema de Hietarinta.*

Las soluciones de YBE en dimensión 2 fueron dadas a conocer por Jarmo Hietarinta [R 6]. Dichas soluciones se presentan a continuación, por visibilidad, como matrices de la forma:

$$\begin{bmatrix} \alpha_{aa}^{aa} & \alpha_{aa}^{ab} & \alpha_{aa}^{ba} & \alpha_{aa}^{bb} \\ \alpha_{ab}^{aa} & \alpha_{ab}^{ab} & \alpha_{ab}^{ba} & \alpha_{ab}^{bb} \\ \alpha_{ba}^{aa} & \alpha_{ba}^{ab} & \alpha_{ba}^{ba} & \alpha_{ba}^{bb} \\ \alpha_{bb}^{aa} & \alpha_{bb}^{ab} & \alpha_{bb}^{ba} & \alpha_{bb}^{bb} \end{bmatrix}$$

aunque en realidad son tensores-(2,2).

Soluciones con 4 parámetros (por visibilidad los puntos representan ceros):

$$\begin{bmatrix} k & . & . & . \\ . & p & . & . \\ . & . & q & . \\ . & . & . & s \end{bmatrix}$$

Soluciones con 3 parámetros:

$$\begin{bmatrix} k^2 & . & . & . \\ . & kp & k^2-pq & . \\ . & . & kq & . \\ . & . & . & k^2 \end{bmatrix}, \begin{bmatrix} k^2 & . & . & . \\ . & kp & k^2-pq & . \\ . & . & kq & . \\ . & . & . & -pq \end{bmatrix}, \begin{bmatrix} p & q & r & s \\ . & p & . & r \\ . & . & p & q \\ . & . & . & p \end{bmatrix}$$

$$, \begin{bmatrix} . & 1/(q+k) & 1/(q-k) & . \\ . & . & . & 1/(p-k) \\ . & . & . & 1/(p+k) \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} p & . & . & s \\ . & p & p-q & . \\ . & . & q & . \\ . & . & . & -q \end{bmatrix}, \begin{bmatrix} k^2 & kp & -kp & pq \\ . & k^2 & . & kq \\ . & . & k^2 & -kq \\ . & . & . & k^2 \end{bmatrix},$$

$$\begin{bmatrix} . & . & . & p \\ . & . & k & . \\ . & k & . & . \\ q & . & . & . \end{bmatrix}, \begin{bmatrix} . & p & p & . \\ . & . & k & q \\ . & k & . & q \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} . & k(-q+k) & -k(q+k) & p^2 \\ . & . & . & q(q-k) \\ . & . & . & -q(q+k) \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} . & p & . & q \\ . & . & . & . \\ . & k & . & . \\ . & . & . & . \end{bmatrix}.$$

Soluciones con 2 parámetros:

$$\begin{bmatrix} p^2+2pq-q^2 & . & . & p^2-q^2 \\ . & p^2+q^2 & p^2-q^2 & . \\ . & p^2-q^2 & p^2+q^2 & . \\ p^2-q^2 & . & . & p^2-2pq-q^2 \end{bmatrix}, \begin{bmatrix} p+q & . & . & . \\ . & q & . & q \\ . & . & p+q & . \\ . & p & . & p \end{bmatrix},$$

$$\begin{bmatrix} . & p & . & . \\ . & . & . & . \\ . & . & . & q \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} . & p & . & . \\ . & . & . & q \\ . & . & . & . \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} . & p & q & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix}, \begin{bmatrix} . & . & . & . \\ . & p & q & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix}.$$

Soluciones sin parámetros:

$$\begin{bmatrix} 1 & . & . & 1 \\ . & -1 & . & . \\ . & . & -1 & . \\ . & . & . & 1 \end{bmatrix}, \begin{bmatrix} 1 & . & . & i \\ . & 1 & 1 & . \\ . & 1 & -1 & . \\ i & . & . & 1 \end{bmatrix}, \begin{bmatrix} 1 & . & . & . \\ . & . & 1 & . \\ . & 1 & . & . \\ . & . & . & 1 \end{bmatrix}, \begin{bmatrix} 1 & . & . & . \\ . & . & . & 1 \\ . & 1 & . & . \\ . & . & . & 1 \end{bmatrix}, \begin{bmatrix} 1 & . & . & . \\ . & . & 1 & . \\ . & 1 & . & . \\ . & . & . & . \end{bmatrix},$$

$$\begin{bmatrix} 1 & 1 & 1 & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & 1 \end{bmatrix}.$$

Capítulo 7 Algunas clasificaciones de las Álgebras Simétricas Cuánticas de 2 generadores.

Una Álgebra Simétrica Cuántica de 2 generadores es una subálgebra generada por estos generadores por medio del producto Shufflé.

7.1 *Álgebras Simétricas Cuánticas Conmutativas.*

Un Álgebra Simétrica Cuántica de 2 generadores será conmutativa si, y solo si, para cualesquiera elementos de ella u y v se cumple que

$u \odot v = v \odot u$.

En el apéndice 11.3 puede verse cómo se obtienen las condiciones necesarias para la conmutatividad siguientes:

$$\alpha_{12}^{11} = \alpha_{21}^{11}$$
$$\alpha_{12}^{12} = \alpha_{21}^{12} + 1$$
$$\alpha_{12}^{21} = \alpha_{21}^{21} - 1$$
$$\alpha_{12}^{22} = \alpha_{21}^{22}$$

que si escribimos las α_{ki}^{jl} en una matriz de 4x4 significa que

$$\begin{pmatrix} * & * & * & * \\ A & B+1 & C & D \\ A & B & C+1 & D \\ * & * & * & * \end{pmatrix}.$$

Los asteriscos en este caso indican que no hay restricción alguna en ese lugar.

7.2 *Álgebras Simétricas Cuánticas Anticonmutativas.*

Un Álgebra Simétrica Cuántica de 2 generadores será anticonmutativa si, y solo si, para los generadores x y y se cumple que

$x \odot y = -y \odot x$.

En el apéndice 11.3 puede verse cómo se obtienen las condiciones necesarias para la anticonmutatividad siguientes:

$$\alpha_{11}^{11} = -1$$
$$\alpha_{11}^{12} = \alpha_{11}^{21} = \alpha_{11}^{22} = 0$$
$$\alpha_{12}^{11} = -\alpha_{21}^{11}$$
$$\alpha_{12}^{12} = -\alpha_{21}^{12} - 1$$
$$\alpha_{12}^{21} = -\alpha_{21}^{21} - 1$$
$$\alpha_{12}^{22} = -\alpha_{21}^{22}$$
$$\alpha_{22}^{11} = \alpha_{22}^{12} = \alpha_{22}^{21} = 0$$
$$\alpha_{22}^{22} = -1$$

que si escribimos las α_{ki}^{jl} en una matriz de 4x4 significa que

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ A & B & C & D \\ -A & -1-B & -1-C & -D \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

Capítulo 8 Soluciones de YBE como conjuntos algebraicos.

Una ***variedad*** se define como el conjunto de puntos que satisfacen

$$f_i(\vec{x}) = 0$$

siendo $\vec{x} = (x_1, x_2, \ldots, x_n)$ y cada f_i un polinomio de las variables x_i.

El conjunto de las soluciones de la ecuación de Yang-Baxter es una *variedad* descrita mediante los polinomios homogéneos de grado 3

$$\sum_{p,q,\in M} \alpha_{s\ v}^{p\ q} \alpha_{q\ n}^{r\ m} \alpha_{p\ r}^{u\ w} - \sum_{p,q,\in M} \alpha_{v\ n}^{q\ p} \alpha_{s\ q}^{u\ r} \alpha_{r\ p}^{w\ m} = 0 .$$

Como puede verse en el Capítulo 6 esta variedad está formada por varias familias de soluciones con diversos números de parámetros. Decimos que cada familia es un subespacio cuya dimensión es el número de parámetros de la familia.

Si la dimensión de una familia de soluciones es un espacio de dimensión k, entonces el espacio tangente a esa solución tendrá también dimensión k.

La dimensión del espacio tangente a una solución dada se puede obtener de la siguiente manera:

Diferenciamos YBE en su forma coordenada obteniéndose un sistema de ecuaciones donde intervienen las variables en YBE y sus diferenciales. Considerando a los diferenciales de las variables como nuevas variables, el sistema es lineal respecto a éstas.

Sustituimos la solución dada en el nuevo sistema quedando un sistema de la forma

$$A \cdot d\vec{x} = 0$$

donde A es la matriz de coeficientes de las variables.

La dimensión de la solución de este sistema es igual a la dimensión del espacio tangente a la solución de YBE dada inicialmente. Dicha dimensión se obtiene mediante la fórmula:

$$NumVars - Rank(A)$$

donde $NumVars$ es el número de variables en $\vec{x}$.

De esta manera es posible clasificar una solución específica de YBE por lo menos en cuanto a su dimensión.

En el apéndice 11.4 puede verse una manera de calcular la dimensión del espacio tangente a una solución específica de YBE.

Capítulo 9 Soluciones de YBE por medio de algoritmos genéticos.

Los algoritmos genéticos pueden ser usados para obtener soluciones de YBE aproximadas. Durante el desarrollo de la presente tesis se usó el algoritmo genético descrito en el apéndice 11.5 que está basado en el algoritmo publicado por Ricardo Paramont Hernández [R 13].

El problema se presentó como un problema de optimización restringida. Se busca minimizar la función objetivo sujeta a las restricciones dadas por una lista de ecuaciones. En nuestro caso la función objetivo es el error total de la solución dada al sustituirla en YBE. Esta función es

$$ft(\alpha) = \sum_{s,v,m,n,u,w \in M} \left| \sum_{p,q,r \in M} \alpha_{s\ v}^{p\ q} \alpha_{q\ n}^{r\ m} \alpha_{p\ r}^{u\ w} - \sum_{p,q,r \in M} \alpha_{v\ n}^{q\ p} \alpha_{s\ q}^{u\ r} \alpha_{r\ p}^{w\ m} \right|.$$

Las restricciones están dadas por

$$\sum_{p,q,r \in M} \alpha_{s\ v}^{p\ q} \alpha_{q\ n}^{r\ m} \alpha_{p\ r}^{u\ w} - \sum_{p,q,r \in M} \alpha_{v\ n}^{q\ p} \alpha_{s\ q}^{u\ r} \alpha_{r\ p}^{w\ m} = 0$$

para cada s, v, m, n, u, w en M.

Nótese que las restricciones equivalen precisamente a YBE.

Una solución (aproximada) se dice que es "factible" cuando cumple con las restricciones dadas. El algoritmo genético busca una mejor solución de entre las que son factibles. En el caso presente, una solución factible es ya una solución de YBE por lo tanto no es necesario buscar entre las factibles una que sea mejor que la otra ya que solamente buscamos soluciones de YBE sin importar cual de todas es. Esto puede notarse también en el hecho de que la función objetivo evalúa con cero a toda solución factible y con un valor mayor que cero a las que no lo son.

Por lo anterior se decidió considerar un margen de error para determinar si una solución es factible o no. Así, el algoritmo genético buscará primero soluciones factibles (cercanas a YBE) y luego una mejor solución de entre aquellas que están "cerca" de ser soluciones.

Con este algoritmo se pudieron encontrar soluciones aproximadas interesantes con muy poco error. Por ejemplo:

$$\begin{pmatrix} 0.626053823672301 & 3.98548649188464 & 3.98548649188464 & -3.20193662004033 \\ -1.41807502421449 & 2.65911821408454 & -0.626053823672301 & -3.98548649188464 \\ -1.41807502421449 & -0.626053823672301 & 2.65911821408454 & -3.98548649188464 \\ 1.56775366769465 & 1.41807502421449 & 1.41807502421449 & 0.626053823672301 \end{pmatrix}$$

con un error total de $3.0750057765502 \times 10^{-8}$.

De esta solución podemos calcular sus 4 coberturas posibles:

$$\alpha_{ki}^{kl}=\begin{pmatrix}\alpha_{11}^{11}+\alpha_{21}^{21} & \alpha_{11}^{12}+\alpha_{21}^{22}\\ \alpha_{12}^{11}+\alpha_{22}^{21} & \alpha_{12}^{12}+\alpha_{22}^{22}\end{pmatrix}=\begin{pmatrix}3.28517203775684 & 0\\ 0 & 3.28517203775684\end{pmatrix}$$

$$\alpha_{ki}^{jk}=\begin{pmatrix}\alpha_{11}^{11}+\alpha_{21}^{12} & \alpha_{11}^{21}+\alpha_{21}^{22}\\ \alpha_{12}^{11}+\alpha_{22}^{12} & \alpha_{12}^{21}+\alpha_{22}^{22}\end{pmatrix}=\begin{pmatrix}3.28517203775684 & 0\\ 0 & 3.28517203775684\end{pmatrix}$$

$$\alpha_{ki}^{il}=\begin{pmatrix}\alpha_{11}^{11}+\alpha_{12}^{21} & \alpha_{11}^{12}+\alpha_{12}^{22}\\ \alpha_{21}^{11}+\alpha_{22}^{21} & \alpha_{21}^{12}+\alpha_{22}^{22}\end{pmatrix}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$$

$$\alpha_{ki}^{ji}=\begin{pmatrix}\alpha_{11}^{11}+\alpha_{12}^{12} & \alpha_{11}^{21}+\alpha_{12}^{22}\\ \alpha_{21}^{11}+\alpha_{22}^{12} & \alpha_{21}^{21}+\alpha_{22}^{22}\end{pmatrix}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$$

Estas cuatro coberturas pueden ser utilizadas para clasificar la solución de YBE dada al compararlas con las coberturas de las familias de soluciones de Hietarinta (ver Capítulo 6.3). Cuando las coberturas son diferentes se puede asegurar que la solución dada NO pertenece a la familia en cuestión. Cuando sí coinciden NO se puede asegurar nada, sin embargo, por eliminación se podría llegar a encontrar la familia a la que pertenece la solución dada.

En el caso de dimensión mayor a 2, no se tiene el conjunto completo de familias de soluciones pero este puede ser un método útil en la investigación de la clasificación de todas las soluciones de YBE.

Capítulo 10 **Conclusión.**

Durante el desarrollo de la presente tesis se pudo comprobar que el uso de la computadora puede ayudar en la investigación de la solución de YBE en dimensión mayor a 2 y para el estudio de temas relacionados como son las Álgebras Simétricas Cuánticas entre otros.

El desarrollo de programas en Maple V, Maple 7 y Maple 8 resultó ser la herramienta más adecuada para este estudio ya que el cálculo simbólico viene ya implementado en ese paquete de software. El uso del lenguaje C++ no es muy recomendable ya que requiere un profundo conocimiento del mismo y de la POO[1] para poder sacar provecho de el. Como puede verse en el programa del apéndice 11.2 es necesario declarar cada operación con cada tipo de dato lo cual hace muy extenso el programa.

Con los programas desarrollados es posible llevar a cabo otras investigaciones futuras sobre estos temas.

Uno de los resultados destacables de esta tesis es la prueba del TEOREMA 9 que, aunque es bien conocido (bajo una restricción adicional de invertibilidad del operador), parece no haber una publicación disponible que la contenga.

Quedan abiertas algunas líneas de investigación como la continuación de los trabajos de J. Hietarinta [R 6] en el sentido de dar una clasificación de todas las soluciones de YBE en dimensión 3.

Otra hipótesis fundamental es que las soluciones de mayor dimensión (mayor número de parámetros) son siempre diagonales para cualquier dimensión de YBE, es decir, que se reducen a la forma de tensor-(1,1).

También es muy interesante encontrar algunas nuevas clases de soluciones de YBE que produzcan Álgebras Simétricas Cuánticas especiales como por ejemplo conmutativas en dimensión arbitraria.

Otro problema actual es la conjetura de Andruskiewitsch-Schneider que establece que cualquier subálgebra de Hopf trenzada de dimensión finita de un Álgebra Shufflé en característica cero es una Álgebra Simétrica Cuántica. Este problema también admite métodos computacionales para su solución.

[1] POO = Programación Orientada a Objetos

Capítulo 11 **Apéndices.**

11.1 *Propiedades de los tensores.*

Supongamos una base e_i de un espacio V de dimensión n. Cada elemento $\vec{x}$ se representa como

$\vec{x} = \alpha^1 e_1 + \alpha^2 e_2 + \cdots + \alpha^n e_n$, o bien como

$\vec{x} = (\alpha^1, \alpha^2, \cdots, \alpha^n)$,

$\vec{x} = \alpha^i e_i$

donde las α^k son las coordenadas del vector $\vec{x}$.

Sea e_i' otra base del mismo espacio. Tenemos que

$$
\begin{aligned}
e_1' &= \xi_1^1 e_1 + \xi_1^2 e_2 + \cdots + \xi_1^n e_n, \\
e_2' &= \xi_2^1 e_1 + \xi_2^2 e_2 + \cdots + \xi_2^n e_n, \\
&\vdots \\
e_n' &= \xi_n^1 e_1 + \xi_n^2 e_2 + \cdots + \xi_n^n e_n
\end{aligned}
$$

que en forma tensorial es

$$e_j' = \xi_j^i e_i .$$

También podemos expresar

$$e_i = \mu_i^j e_j'$$

por lo tanto

$$e_j' = \xi_j^i \mu_i^k e_k'$$

de donde deducimos que

$$\xi_j^i \mu_i^k = \delta_j^k ,$$

que es la matriz identidad.

Definimos las matrices

$$Q = \left[\xi_j^i \right],$$

$$T = \left[\mu_l^k \right]$$

entonces

$$QT = \mathrm{E} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Por el otro lado tenemos que

$$e_i = \mu_i^j e_j' = \mu_i^j \xi_j^k e_k$$

entonces

$$\mu_i^j \xi_j^k = \delta_i^k \text{, y}$$

$$TQ = \mathrm{E},$$

por lo tanto tenemos que

$$T = Q^{-1} \text{ y } Q = T^{-1}.$$

Ahora escribimos

$$\vec{x} = \alpha^i \mu_i^j e_j'$$

y definimos

$$\beta^j = \alpha^i \mu_i^j,$$

que son las coordenadas de $\vec{x}$ en la base e_j', y entonces decimos que

$$\beta = \alpha T.$$

Si el vector $\vec{x}$ se comporta de esta manera al cambiar de base entonces $\vec{x}$ es un **tensor-(1,0)**.

Sea V un espacio lineal y

$V^* = \{$ mapas lineales de V a $\boldsymbol{k}\}$

se cumple que

$$\dim(V^*) = \dim(V).$$

El mapa

$$e_*^1 : e_1 \to 1$$

$$e_*^1 : e_2 \to 0$$

$$\vdots$$

$$e_*^1 : e_n \to 0$$

nos da

$$e_*^1(\vec{x}) = e_*^1(\alpha^i e_i) = \alpha^1,$$

y en general tenemos

$$e_*^k(\vec{x}) = e_*^k(\alpha^i e_i) = \alpha^k .$$

Los mapas e_*^k forman una base del espacio V^* (del espacio dual de V^*). Así, un elemento f_* de V cumple

$$f_* = \lambda_1 e_*^1 + \lambda_2 e_*^2 + \cdots + \lambda_n e_*^n = \lambda_i e_*^i$$

donde

$$\lambda_i = f_*(e_i).$$

En una base nueva digamos e_*^i.

$$f_* = \gamma_1 e_*^1 + \gamma_2 e_*^2 + \cdots + \gamma_n e_*^n = \gamma_i e_*^i$$

con

$$\gamma_i = f_*(e_i').$$

Sustituyendo e_i' por su equivalente en la expresión anterior

$$\gamma_i = f_*(e_i') = f_*(\xi_i^j e_j) = \xi_i^j f_*(e_j) = \xi_i^j \lambda_j .$$

Lo anterior nos dice que, las coordenadas de f_* en la nueva base están dadas por

$$\gamma_i = \xi_i^j \lambda_j$$

que en forma matricial es

$$\gamma = M\lambda = T^{-1}\lambda .$$

Si f_* se comporta así entonces es un **tensor-(0,1)**.

Un **tensor-(1,1)** se comporta según

$$D' = T^{-1} D T ,$$

donde D es una matriz de coeficientes en una base y D' la matriz de coeficientes correspondiente en una base nueva.

En general un **tensor-(m, n)** se comporta según

$$D' = (T^{\otimes n})^{-1} D (T^{\otimes m}).$$

11.2 *Álgebra Shufflé Cuántica con un tensor (1,1) en C++.*

```
////////////////////////////////////////////////////////////////////////
//
// Shuffle.cpp
//
// Implementa el álgebra de shuffle ( mezclas ).
//
// el operador / significa Shuffle y tiene la misma prioridad que *
//
////////////////////////////////////////////////////////////////////////
//
// Alumno:   Jorge Cervantes Ojeda
// Materia:  Seminario de Investigación I
// Profesor: Vladislav Khartchenko
//
////////////////////////////////////////////////////////////////////////

#include <iostream.h>
#include <stdlib.h>
#include <string.h>

#include <fstream.h>
//#include <stdio.h>
//#include <stdlib.h>
//#include <math.h>
#include <conio.h>

unsigned long int count=0;
unsigned long int concatenaciones = 0;
unsigned long int shuffles = 0;

unsigned long int comparaciones = 0;
unsigned long int adiciones = 0;
unsigned long int asignaciones = 0;

////////////////////////////////////////////////////////////////////////
//
// 1.- utilerías
//
////////////////////////////////////////////////////////////////////////
void pausa()
{
        cout<<"...";
        cout.flush();

        char c;
        while( kbhit() )
                c = getch();

        while( !kbhit() )
                ;// esperar a que el usuario oprima una tecla

        c = getch();

        cout<<"\b\b\b";
}


////////////////////////////////////////////////////////////////////////
//
// 2.- manejo de errores
//
// En caso de cualquier error se llama a ésta función
//
////////////////////////////////////////////////////////////////////////
void ERROR( char * mensaje )
{
        cout << endl << "ERROR: " << mensaje << endl;
        pausa();
```

```
        // Termina la ejecución de todo el programa
        // con código de error diferente de cero
        exit(1);
};

////////////////////////////////////////////////////////////////////////
//
// clase Tensor (1,1) en archivo de texto
//
////////////////////////////////////////////////////////////////////////
class FileTensor_1_1
{
  int rows, cols;
  long int * elements;
public:
  FileTensor_1_1(char * filename);
  ~FileTensor_1_1();
  long int & operator()(int r, int c);
};

FileTensor_1_1 Coeficientes("Coef.txt");

////////////////////////////////////////////////////////////////////////
//
// clase Palabra
//
// Se sobrecargan operadores para poder hacer:
//
// '=' asignación con un caracter simple
// '=' asignación con apuntador a caracteres
// '=' asignación con otra Palabra
// '*='concatenación con un caracter
// '*='concatenación con otra Palabra
// '=='comparación de igualdad con otra Palabra
// '>' comparación de superioridad con otra Palabra
//
////////////////////////////////////////////////////////////////////////
class Polinomio;
class Termino;


class Palabra
{
        char * cadena;
public:
  // constructores:
  Palabra( char  aChar = 0 ); // con caracter
        Palabra( char * aPtr ); // con apuntador a caracteres
        Palabra( Palabra & P1 ); // con otra Palabra
  // destructor
        ~Palabra();

  // asignación
  Palabra & operator = ( char c );
        Palabra & operator = ( char* aPtr );
        Palabra & operator = ( Palabra & P1 );

  // suma
        Polinomio operator + ( Palabra & P1 );
        Polinomio operator + ( Termino & T1 );
        Polinomio operator + ( Polinomio & P1 );

  // concatenación
        Palabra   operator * ( Palabra & P1 );
        Termino   operator * ( Termino & T1 );
        Polinomio operator * ( Polinomio & P1 );

  // concatenación y asignación
        Palabra & operator *= ( Palabra & P1 );
```

```
  // comparación
       int operator != ( Palabra& P1 );
       int operator == ( Palabra& P1 );

  // Mezcla
  Polinomio operator / ( char * P1 );
  Polinomio operator / ( Palabra & P1 );
  Polinomio operator / ( Termino & T1 );
  Polinomio operator / ( Polinomio & P1 );

  // Otros
  char Cabeza();
  char * Cola();
  size_t length();

  friend ostream & operator << ( ostream & out, Palabra & P1 );
};

////////////////////////////////////////////////////////////////////////
// clase Termino
////////////////////////////////////////////////////////////////////////

class Termino
{
  long int El_Coeficiente;
  Palabra La_Palabra;
public:
  // constructores
  Termino(char * c );
  Termino(Palabra & P1 );
  Termino(long int coef, char * c );
  Termino(long int coef, Palabra & P1 );
  // destructor
  ~Termino();

  // asignación
  Termino & operator = ( Palabra & P1 );
  Termino & operator = ( Termino & T1 );

  // suma
       Polinomio operator + ( Palabra& P1 );
       Polinomio operator + ( Termino& T1 );
       Polinomio operator + ( Polinomio& P1 );

  // multiplicación por entero
  Termino   operator * ( long int n );

  // concatenación
       Termino   operator * ( Palabra& P1 );
       Termino   operator * ( Termino& T1 );
       Polinomio operator * ( Polinomio& P1 );

  // suma y asignaciòn
  Termino & operator += ( Termino& T1 );

  // multiplicación por entero y asignación
  Termino & operator *= ( long int n );

  // concatenación y asignación
       Termino & operator *= ( Palabra& P1 );
       Termino & operator *= ( Termino& T1 );

  // comparación
       int operator != ( Termino & T1 );

  // Mezcla
  Polinomio operator /( Palabra& P1 );
  Polinomio operator /( Termino& T1 );
  Polinomio operator /( Polinomio& P1 );
```

```
  // obtener
  int Coef();
  Palabra & Pal();

  friend ostream & operator << ( ostream & out, Termino & T1 );
};

////////////////////////////////////////////////////////////////////////
// clase TerminoList
////////////////////////////////////////////////////////////////////////

class TerminoList
{
protected:
  int size;
  Termino ** List;
  // clear all
  void clear();
public:
  // constructors
  TerminoList();
  TerminoList(int s, Termino ** L);
  // destructor
  ~TerminoList();

  // asignación
  TerminoList & operator = ( TerminoList TL );
  TerminoList & operator = ( Termino T1 );
  TerminoList & operator = ( Palabra P1 );

  // add element
  void add( Termino & E, int pos );
  // remove element
  void rem( int pos );

  // search the first element lower or equal to x
  int Search( Termino & x, int & r );
};


////////////////////////////////////////////////////////////////////////
// clase Polinomio
////////////////////////////////////////////////////////////////////////

class Polinomio: public TerminoList
{
public:
  // constructores
  Polinomio();
  Polinomio(Palabra & P1);
  Polinomio(Termino & T1);
  Polinomio(Polinomio & P1);
  Polinomio(int s, Termino ** L);
  // destructor
  ~Polinomio();

  // asignación
  Polinomio & operator = ( Palabra & P1 );
  Polinomio & operator = ( Termino & T1 );
  Polinomio & operator = ( Polinomio & P1 );

  // suma
  Polinomio operator + ( Palabra & P1 );
  Polinomio operator + ( Termino & T1 );
  Polinomio operator + ( Polinomio & P1 );

  // suma y asignación
  Polinomio & operator += ( Palabra & P1 );
  Polinomio & operator += ( Termino & T1 );
  Polinomio & operator += ( Polinomio & P1 );
```

```
  // multiplicación por entero
  Polinomio operator * ( long int n );

  // concatenación
  Polinomio operator * ( Palabra & P1 );
  Polinomio operator * ( Termino & T1 );
  Polinomio operator * ( Polinomio & P1 );

  // multiplicación por entero y asignación
  Polinomio & operator *= ( long int n );

  // Mezcla
  Polinomio operator /( Palabra& P1 );
  Polinomio operator /( Termino& T1 );
  Polinomio operator /( Polinomio& P1 );

  // otros
  int Size(void);
  Termino & operator[](int i);
  int operator != ( Polinomio & P1 );

  friend ostream & operator << ( ostream & out, Polinomio & P1 );
};



////////////////////////////////////////////////////////////////////////
//
// clase Matriz en archivo de texto
//
////////////////////////////////////////////////////////////////////////
FileTensor_1_1:: FileTensor_1_1(char * filename)
{
  ifstream f;
       f.open( filename );
       if( f.flags() & ios::failbit )
              ERROR("No se puede abrir el archivo");

  f>>rows>>cols;
  elements = new long int [rows*cols];
  for( int r=0; r<rows; r++ )
    for( int c=0; c<cols; c++ )
      f>>elements[r*cols+c];
  f.close();
}

FileTensor_1_1:: ~FileTensor_1_1()
{
  if( elements != NULL )
    delete elements;
  elements = NULL;
}

long int & FileTensor_1_1::operator () ( int r, int c )
{
  return elements[r*cols+c];
};

////////////////////////////////////////////////////////////////////////
// operadores globales
////////////////////////////////////////////////////////////////////////
Termino operator * ( int c, Palabra & P1 )
{
  return Termino( c, P1 );
};

Termino operator * ( int c, Termino & T1 )
{
  return Termino( c*T1.Coef(), T1.Pal() );
```

```
};

Polinomio operator *( int c, Polinomio & P1 )
{
  Polinomio Temp(P1);
  int s = P1.Size();
  for( int i=0; i<s; i++ )
    Temp[i] *= c;
  return Temp;
};

Palabra operator * ( char c, Palabra & P1 )
{
  return Palabra(c) *= P1;
};

Termino operator * ( char c, Termino & T1 )
{
  return Termino( T1.Coef(), c*T1.Pal() );
};


Polinomio operator *( char c, Polinomio & P1 )
{
  Polinomio Temp(P1);
  int s = P1.Size();
  for( int i=0; i<s; i++ )
    Temp[i].Pal() = c*Temp[i].Pal();
  return Temp;
};



ostream & operator << ( ostream & out, Palabra & P1 )
{
  out << P1.cadena;
  return out;
};
ostream & operator << ( ostream & out, Termino & T1 )
{
  if( T1.Coef() >=  0 ) out << "+";
  if( T1.Coef() == -1 ) out << "-";
  else
  if( T1.Coef() !=  1 ) out << T1.Coef();
  out << T1.Pal();
  return out;
};
ostream & operator << ( ostream & out, Polinomio & P1 )
{
  int s = P1.Size();
  if( s == 0 )
    out << 0;
  for( int i=0; i<s; i++)
    out << P1[i];
  return out;
};



////////////////////////////////////////////////////////////////////////
// Implementación de métodos de Palabra
////////////////////////////////////////////////////////////////////////

// constructores:
Palabra:: Palabra( char  aChar ) // con caracter
{
       cadena = NULL;
```

```
        *this = aChar; // ver abajo asignación con caracter
};
Palabra::     Palabra( char* aPtr  ) // con apuntador a caracteres
{
        cadena = NULL;
        *this = aPtr;  // ver abajo asignación con apuntador a caracteres
};
Palabra::     Palabra( Palabra& P1 ) // con otra Palabra
{
        cadena = NULL;
        *this = P1;  // ver abajo asignación con otra Palabra
};

// destructor
Palabra::     ~Palabra()
{
        if( cadena != NULL )
                delete [] cadena;
        cadena = NULL;
};

// asignacion
Palabra& Palabra:: operator = ( char c )
{
        char *temp = new char[2];
        temp[0] = c;
        temp[1] = 0;
        *this = temp; // ver abajo asignación con apuntador a caracteres
        delete temp;

        return *this;
};

Palabra& Palabra:: operator=( char* aPtr )
{
  count++;
  asignaciones++;
  if( cadena == aPtr )
    return *this;

        if( cadena != NULL ) delete cadena;
        cadena = new char [strlen(aPtr)+1];
        strcpy( cadena, aPtr );

        return *this;
};

Palabra& Palabra:: operator=( Palabra& P1 )
{
        return *this = P1.cadena;
        // ver asignación con apuntador a caracteres
};

  // suma
Polinomio Palabra::operator + ( Palabra & P1 )
{
  return Polinomio(*this) + P1;
}
Polinomio Palabra::operator + ( Termino & T1 )
{
  return Polinomio(*this) + T1;
}
Polinomio Palabra::operator + ( Polinomio & P1 )
{
  return Polinomio(*this) + P1;
}

// concatenación
Palabra Palabra:: operator*( Palabra & P1 )
{
  count++;
```

```
  concatenaciones++;
       int n = length()+P1.length()+1;
       char * temp = new char[ n ];
       strcpy(temp, cadena );
       strcat(temp, P1.cadena );

  Palabra P(temp);
       delete [] temp;

       return P;
};

Termino Palabra::operator * ( Termino & T1 )
{
  return Termino( T1.Coef(), (*this)*T1.Pal() );
};

Polinomio Palabra::operator * ( Polinomio & P1 )
{
  Polinomio Temp(P1);
  int s = P1.Size();

  for( int i=0; i<s; i++ )
    Temp[i] = (*this)*Temp[i];

  return Temp;
};

// concatenación y asignación
Palabra & Palabra::operator *= ( Palabra & P1 )
{
  count++;
  concatenaciones++;

       int n = length()+P1.length()+1;
       char * temp = new char[ n ];
       strcpy(temp, cadena );
       strcat(temp, P1.cadena );

  if( cadena != NULL ) delete [] cadena;

  cadena = temp;

  return (*this);
};

// comparación
int Palabra:: operator != ( Palabra & P1 )
{
  count++;
  comparaciones++;
  return strcmp( cadena, P1.cadena );
};
int Palabra:: operator == ( Palabra & P1 )
{
  return !(*this != P1);
}

// Mezcla ((((((((((((((((((((((((((((((((((((((((((((((((((Shuffle))))))))))))))))))))))))))))))))))))))))))))))))))
Polinomio Palabra:: operator / ( char * P1 )
{
  count++;
  shuffles++;
  if( cadena[0] == 0 )
    return Palabra(P1);

  if( P1[0] == 0 )
    return *this;

  long int Coeficiente=1;
  for( int i=length()-1; i>=0; i-- )
```

```
    Coeficiente *= Coeficientes( cadena[i], P1[0] );

  Polinomio resultado;
  if( Coeficiente != 0 )
    resultado  = Coeficiente*(   P1[0]*( (*this)         / (P1+1) ));
  resultado   +=              Cabeza()*( Palabra(Cola()) /  P1    );

  return resultado;
};
Polinomio Palabra:: operator / ( Palabra & P1 )
{
  return (*this) / P1.cadena;
};
Polinomio Palabra:: operator / ( Termino & T1 )
{
  return T1.Coef() * ( (*this) / T1.Pal() );
};
Polinomio Palabra:: operator / ( Polinomio & P1 )
{
  Polinomio Temp;
  int s = P1.Size();
  for( int i=0; i<s; i++ )
    Temp += (*this) / P1[i];
  return Temp;
};

// Otros
char Palabra::Cabeza()
{
  return cadena[0];
};
char * Palabra::Cola()
{
  return cadena + 1;
};
size_t Palabra::length()
{
  return strlen( cadena );
};
```

```
////////////////////////////////////////////////////////////////////////
// Implementación de métodos de Termino
////////////////////////////////////////////////////////////////////////

// constructores
Termino:: Termino(char * c ):
  El_Coeficiente(1),
  La_Palabra(c)
{};
Termino:: Termino( Palabra & Pal ):
  El_Coeficiente(1),
  La_Palabra(Pal)
{};
  Termino:: Termino(long int coef, char * c ):
  El_Coeficiente(coef),
  La_Palabra(c)
{};
Termino:: Termino(long int coef, Palabra & Pal ):
  El_Coeficiente(coef),
  La_Palabra(Pal)
{};
// destructor
Termino:: ~Termino(){};

// asignación
```

```
Termino & Termino::operator = ( Palabra & P1 )
{
  El_Coeficiente = 1;
  La_Palabra     = P1;
  return *this;
};
Termino & Termino::operator = ( Termino & T1 )
{
  El_Coeficiente = T1.El_Coeficiente;
  La_Palabra     = T1.La_Palabra;
  return *this;
};

// suma
Polinomio Termino::operator + ( Palabra& P1 )
{
  return Polinomio(*this)+P1;
};
Polinomio Termino::operator + ( Termino& T1 )
{
  return Polinomio(*this)+T1;
};
Polinomio Termino::operator + ( Polinomio& P1 )
{
  return Polinomio(*this)+P1;
};

// multiplicación por entero
Termino   Termino::operator * ( long int n )
{
  return n*(*this);
};

// concatenación
Termino   Termino::operator * ( Palabra& P1 )
{
  return Termino( El_Coeficiente, La_Palabra*P1 );
};
Termino   Termino::operator * ( Termino& T1 )
{
  return Termino( El_Coeficiente*T1.El_Coeficiente, La_Palabra*T1.La_Palabra );
};
Polinomio Termino::operator * ( Polinomio& P1 )
{
  Polinomio Temp(P1);
  int s = P1.Size();
  for(int i=0; i<s;i++ )
    Temp[i] = (*this)*Temp[i];
  return Temp;
};

// suma y asignación
Termino & Termino::operator += ( Termino& T1 )
{
  El_Coeficiente += T1.Coef();
  return (*this);
};

// multiplicación por entero y asignación
Termino & Termino::operator *= ( long int n )
{
  El_Coeficiente *= n;
  return (*this);
};

// concatenación y asignación
Termino & Termino::operator *= ( Palabra& P1 )
{
  La_Palabra *= P1;
  return (*this);
};
```

```
Termino & Termino::operator *= ( Termino& T1 )
{
  El_Coeficiente *= T1.Coef();
  La_Palabra     *= T1.Pal();
  return (*this);
};

// comparación
int Termino::operator != ( Termino & T1 )
{
  return La_Palabra != T1.La_Palabra;
};

// Mezcla
Polinomio Termino::operator /( Palabra& P1 )
{
  return El_Coeficiente * (La_Palabra / P1);
};
Polinomio Termino::operator /( Termino& T1 )
{
  return (El_Coeficiente*T1.El_Coeficiente)*(La_Palabra / T1.La_Palabra);
};
Polinomio Termino::operator /( Polinomio& P1 )
{
  Polinomio Temp;
  int s = P1.Size();
  for(int i=0; i<s; i++)
    Temp += (*this) / P1[i];
  return Temp;
};

// obtener
int Termino::Coef()
{
  return El_Coeficiente;
};
Palabra & Termino::Pal()
{
  return La_Palabra;
};


////////////////////////////////////////////////////////////////////////
// Implementación de métodos de TerminoList
////////////////////////////////////////////////////////////////////////

// constructors
TerminoList::TerminoList()
{
  size = 0;
  List = NULL;
};
TerminoList:: TerminoList(int s, Termino ** L): size(s)
{
  List = new Termino * [size];
  for( int i=0; i<size; i++ )
    List[i] = new Termino(*(L[i]));
};
// destructor
TerminoList::~TerminoList()
{
  clear();
};

// clear all
void TerminoList::clear()
{
  if( List == NULL )
    return;

  for( int i=0; i<size; i++ )
```

```
    if( List[i] != NULL )
      delete List[i];
  delete [] List;
  List = NULL;
  size = 0;
};

// add element
void TerminoList::add( Termino & E, int pos )
{
  Termino **Temp = new Termino * [size+1];
  for( int i=0; i<size; i++ )
    if( i<pos )
      Temp[i] = List[i];
    else
      Temp[i+1] = List[i];
  Temp[pos] = new Termino(E);
  delete [] List;
  List = Temp;
  size++;
};

// remove element
void TerminoList::rem( int pos )
{
  Termino **Temp = new Termino * [size-1];
  for( int i=0; i<size; i++ )
    if( i<pos )
      Temp[i] = List[i];
    else
    if( i>pos )
      Temp[i-1] = List[i];
  delete List[pos];
  delete [] List;
  List = Temp;
  size--;
};

// search the first element lower or equal to x
int TerminoList::Search( Termino & x, int & r )
{
  r = -1;
  if( size == 0 ) return 0;

  int l=0, h=size, i=size/2;
  while( l<h && i<size )
  {
    r = *List[i] != x;
    if( r == 0 ) return i;
    if( r <  0 ) h = i;
    if( r >  0 ) l = i+1;
    i = (l+h)/2;
  }
  return i;
};

////////////////////////////////////////////////////////////////////////
// Implementación de métodos de Polinomio
////////////////////////////////////////////////////////////////////////

// constructores
Polinomio:: Polinomio(){};
Polinomio:: Polinomio( Palabra & P1 )
{
  *this = P1;
};
Polinomio:: Polinomio( Termino & T1 )
{
  *this = T1;
};
Polinomio:: Polinomio( Polinomio & P1 )
```

```
{
  *this = P1;
};
Polinomio:: Polinomio(int s, Termino ** L): TerminoList(s, L)
{};
// destructor
Polinomio:: ~Polinomio(){};

// asignación
Polinomio& Polinomio::operator=(Polinomio& P1)
{
  clear();
  int s = P1.Size();
  List = new Termino * [s];
  for( size=0; size<s; size++ )
    List[size] = new Termino(*P1.List[size]);

  return *this;
};
Polinomio& Polinomio::operator=(Termino& T1)
{
  clear();
  List = new Termino * [1];
  List[0] = new Termino(T1);
  size = 1;
  return *this;
};
Polinomio& Polinomio::operator=(Palabra& P1)
{
  *this = Termino(P1);
  return *this;
};

// suma
Polinomio Polinomio::operator+(Polinomio& P1)
{
  Polinomio Temp(*this);
  int s = P1.Size();
  for( int i=0; i<s; i++ )
    Temp += P1[i];

  return Temp;
};
Polinomio Polinomio::operator+(Termino& T1)
{
  count++;
  adiciones++;
  Polinomio Temp;
  Temp = *this;

  int r, i = Temp.Search( T1, r );
  if( i == Temp.size
      ||
      Temp[i] != T1
    )
    Temp.add( T1, i );

  else
  {
    Temp[i] += T1;
    if( Temp[i].Coef() == 0 )
      Temp.rem(i);
  }

  return Temp;
};
Polinomio Polinomio::operator+(Palabra& P1)
{
  return (*this) + Termino(P1);
};
```

```
// suma y asignación
Polinomio & Polinomio::operator += ( Palabra & P1 )
{
  *this += Termino(P1);
  return (*this);
};
Polinomio & Polinomio::operator += ( Termino & T1 )
{
  count++;
  adiciones++;
  int r, i = Search( T1, r );
  if( i == size
      ||
      r != 0
    )
    add( T1, i );

  else
  {
    (*this)[i] += T1;
    if( (*this)[i].Coef() == 0 )
      rem(i);
  }

  return (*this);
};
Polinomio & Polinomio::operator += ( Polinomio & P1 )
{
  for( int i=0; i<P1.Size(); i++ )
    (*this) += P1[i];
  return (*this);
};

// multiplicación por entero
Polinomio Polinomio::operator * ( long int n )
{
  Polinomio Temp(*this);
  for( int i=0; i<size; i++ )
    Temp[i] *= n;
  return Temp;
};

// concatenación
Polinomio Polinomio::operator*(Palabra& P1)
{
  Polinomio Temp(*this);

  for( int i=0; i<size; i++)
    Temp[i] *= P1;

  return Temp;
};
Polinomio Polinomio::operator*(Termino& T1)
{
  Polinomio Temp(*this);

  for( int i=0; i<size; i++)
    Temp[i] *= T1;

  return Temp;
};
Polinomio Polinomio::operator*(Polinomio& P1)
{
  Polinomio Temp;

  for( int i=0; i<size; i++)
    Temp += (*this)[i] * P1;

  return Temp;
};
```

```
// multiplicación por entero y asignación
Polinomio & Polinomio::operator *= ( long int n )
{
  for( int i=0; i<size; i++ )
    (*this)[i] *= n;
  return (*this);
}

// Mezcla
Polinomio Polinomio::operator / ( Palabra& P1 )
{
  if( size == 0 )
    return Polinomio(P1);

  Polinomio Temp;

  for( int i=0; i<size; i++)
    Temp += (*this)[i] / P1;

  return Temp;
};
Polinomio Polinomio::operator /( Termino& T1 )
{
  return T1.Coef()* ( (*this) / T1.Pal() );
};
Polinomio Polinomio::operator /( Polinomio& P1 )
{
  if( size == 0 )
    return P1;

  Polinomio Temp;

  for( int i=0; i<size; i++)
    Temp += (*this)[i] / P1;

  return Temp;
};

// Otros
int Polinomio::Size(void)
{
  return size;
};

Termino & Polinomio::operator[](int i)
{
  return *List[i];
};

int Polinomio:: operator != ( Polinomio & P1 )
{
  if( size == 0
      &&
      P1.Size() == 0
    )
    return 0;

  if( size != P1.Size() )
    return size - P1.Size();

  for( int i=0; i<size && i<P1.Size(); i++ )
  {
    int d = (*this)[i] != P1[i];
    if( d != 0 )
      return d;
  }
  return 0;
};

//////////////////////////////////////////////////////////////
//
```

```
// Programa principal para probar las clases
//
////////////////////////////////////////////////////////////

int main()
{

  Polinomio S1(Palabra("ff")), S2=Palabra("gg");

  cout << 5*(S1*(S1/S2)) << endl;
  cout << 5*('y'*(S2/S1)) << endl<< endl;
  cout << S1/(2*S2) << endl;
  cout << (2*S1)/S2 << endl<< endl;

  char c[2][80];
  while(1)
  {
    for( int i=0; i<2; i++ )
    {
      cout<<"escriba la palabra "<<i+1<<": ";
      cin>>c[i];
      if( c[i][0] == '?' )
        return 0;
    }

    count = concatenaciones = comparaciones = shuffles = adiciones = asignaciones = 0;
    Polinomio  S=Palabra(c[0])/(char*)(c[1]);
    {
      cout<<"el shuffle es:"<<endl<<S<<endl;

      cout
        << "concatenaciones " << concatenaciones  << endl
        << "shuffles        " << shuffles         << endl
        << "comparaciones   " << comparaciones    << endl
        << "adiciones       " << adiciones        << endl
        << "asignaciones    " << asignaciones     << endl
        << "total           " << count            << endl
        << endl;
    }
  }

  return 0;
}
```

11.3 *Álgebra Shufflé Cuántica con tensores (2,2) en Maple.*

Hoja de Maple que implementa el Álgebra Shufflé Cuántica

```
> restart;
```

Definimos los generadores del Álgebra Shufflé y la dimensión del espacio vectorial que generan.

```
> simbolos := [a,b]; dim := nops(simbolos);
```

$$simbolos := [a, b]$$

$$dim := 2$$

El siguiente arreglo se define solamente para poder mostrar en pantalla las soluciones de una manera amigable.

```
> R:=array(1..dim*dim,1..dim*dim);
```

$$R := \text{array}(1 .. 4, 1 .. 4, [\ ])$$

```
> for p from 0 to dim-1 do
> for q from 0 to dim-1 do
> for r from 0 to dim-1 do
> for s from 0 to dim-1 do
>   R[dim*p+q+1,dim*r+s+1]:=
A[simbolos[p+1],simbolos[q+1],simbolos[r+1],simbolos[s+1]];
>   alias(A[p*dim^3+q*dim^2+r*dim+s+1]=R[dim*p+q+1,dim*r+s+1]);
> od;od;od;od;
> evalm(R);
```

$$\begin{bmatrix} A_1 & A_2 & A_3 & A_4 \\ A_5 & A_6 & A_7 & A_8 \\ A_9 & A_{10} & A_{11} & A_{12} \\ A_{13} & A_{14} & A_{15} & A_{16} \end{bmatrix}$$

Procedimiento: palabras
parámetros: x : polinomio Shufflé
regresa una lista con las palabras del polinomio x

```
> palabras := proc( x )
> option remember;
>   return sort(select(type,convert(indets(x),list),symbol));
> end:
> untrace(palabras);
```

Ejemplo:

```
> palabras(2*A[a,a,a,a]*ba-3*A[a,a,b,a]*aac+aac);
```

$$[aac, ba]$$

Procedimiento: presentar
parámetros: x: polinomio Shufflé
regresa el polinomio x agrupado por palabras y en orden

```
> presentar := proc( x )
> option remember;
>   return sort(collect(expand(x),palabras(x)),palabras(x));
> end:
```

Ejemplo:

```
> presentar(2*A[a,a,a,a]*ba-3*A[a,a,b,a]*aa+aa);
```

$$(-3 A_3 + 1)\, aa + 2 A_1\, ba$$

Procedimiento: encadenar
parámetros: x,y : polinomios (o monomios) Shufflé
Encadena los monomios de x con los de y. Esta operación es la multiplicación NO conmutativa de polinomios

```
> encadenar:= proc( x::{name, `*`,`+`}, y::{name, `*`,`+`} )
> option remember;
> local E;
>     if type(x,`+`) then E:= map(encadenar,x,y);
>   elif type(x,`*`) then E:= x/palabras(x)[1]*encadenar(palabras(x)[1],y);
>   elif type(y,`+`) then E:= map2(encadenar,x,y);
>   elif type(y,`*`) then E:= y/palabras(y)[1]*encadenar(x,palabras(y)[1]);
>   else E:= cat(x,y);
```

```
>   end if;
>   return presentar(E);
>end:
>untrace(encadenar);
```

Ejemplo: $(A_1 a + 2A_6 b)(2A_1 aa - 3A_3 ba)$

```
>encadenar(A[a,a,a,a]*a+2*A[a,b,a,b]*b,2*A[a,a,a,a]*aa-3*A[a,a,b,a]*ba);
```

$$2A_1^{\,2}\,aaa - 3A_1A_3\,aba + 4A_6A_1\,baa - 6A_6A_3\,bba$$

Procedimiento: τ
parámetros: i,j : símbolos
ó i: entero, j: polinomio Shufflé
Si los parámetros son símbolos. este procedimiento regresa el polinomio correspondiente a dichos símbolos de acuerdo al tensor usado A.
En este caso la función τ es de la forma:

$$\tau : (y_i, y_j) \to \sum_{p,q} A_{i,j}^{p,q}\, y_p y_q$$

Si los parámetros con un entero y un polinomio, la función tau es de la forma τ_i que se aplica como una transveccion del i-ésimo generador con el (i+1)-ésimo de cada monomio del polinomio. Entonces:
τ_i (monomio) = (coeficiente del monomio)*(primeros i-1 símbolos en el monomio)· τ (i-ésimo símbolo, (i+1)-ésimo símbolo· (resto de los símbolos)

```
>tau := proc(
>i::{integer,name},
>j::{name,`*`,`+`} )
>option remember;
>   local h,m,t,TAU,p,q;
>   if type(i, name) then
>     TAU := 0;
>     for p in simbolos do
>       for q in simbolos do
>         TAU := TAU + A[i,j,p,q]*cat(p,q);
>       end;
>     end;
>   elif type(j,`+`) then TAU := map2(tau,i,j);
>   elif type(j,`*`) then TAU := j/palabras(j)[1]*tau(i, palabras(j)[1]);
>   else
>     h := substring(j,1..(i-1));
>     m := tau(substring(j,i),substring(j,i+1));
>     t := substring(j,i+2..length(j));
>     TAU := encadenar(h,encadenar(m,t))
>   end;
>   return presentar(TAU);
>end:
>untrace(tau);
```

Ejemplos:

```
>tau(a,b);
```

$$A_5\,aa + A_6\,ab + A_7\,ba + A_8\,bb$$

```
>tau(3, 5*ababa-2*aaabb+aaaaaa);
```

$$A_1\,aaaaaa - 2A_5\,aaaab + A_2\,aaabaa - 2A_6\,aaabb + A_3\,aabaaa - 2A_7\,aabab + A_4\,aabbaa$$
$$- 2A_8\,aabbb + 5A_5\,abaaa + 5A_6\,ababa + 5A_7\,abbaa + 5A_8\,abbba$$

Procedimiento: Shuffle
parámetros: x,y: polinomios
Aplica el producto Shufflé entre cada uno de los monomios de un polinomio y cada uno de los del otro.
El producto Shufflé se define como:

$$Shuffle : (u,v) \to \sum_{\pi \in S(k)} \pi^b (uv)$$

S(k) = permutaciones k-Shufflé

π^b es la representación de π en términos de τ_i's

```
>Shuffle := proc(
>x::{name,`*`,`+`},
>y::{name,`*`,`+`} )
>local Sh;
>    if type(x,`+`) then Sh := map( Shuffle, x, y );
>  elif type(y,`+`) then Sh := map2( Shuffle, x, y );
>  elif type(x,`*`) then Sh := x/palabras(x)[1]*Shuffle( palabras(x)[1], y );
>  elif type(y,`*`) then Sh := y/palabras(y)[1]*Shuffle( x, palabras(y)[1] );
>  else Sh := kSh(length(x), encadenar(x,y) );
>  end;
>  return presentar(Sh);
>end:
>untrace(Shuffle); forget(Shuffle);
```

Procedimiento kSh
parámetros: k : entero
w: polinomio
regresa el producto Shufflé de la palabra "x" formada por los primeros k simbolos de w con la palabra "y" que se forma con el resto de los símbolos de w.

La forma en que se calcula el producto Shufflé en este procedimiento es recursiva:
Shuffle(x, "") = x
Shuffle("", y) = y
Shuffle(x, y) = x[1] · ((k-1)-Shuffle de x[2..k]· y)
+ y[1] · (k-Shuffle de tau[k] de tau[k-1] de ... de tau[1] de x· y[2..n])
donde k = longitud de x
n = longitud de y

```
>kSh := proc(
>k::integer,
>w::{name,`*`,`+`} )
>option remember;
>local part1,part2,kSh,i,v;
>  if k<=0 or length(w) <= k then return w; end;
>    if type(w,`+`) then kSh := map2( kSh, k, w );
>  elif type(w,`*`) then kSh := w/palabras(w)[1]*kSh(k,palabras(w)[1]);
>  else
>    part1 := postkShuffle( k-1, w );
>    v := w;
>    for i from k by -1 to 1 do
>      v := tau(i,v);
>    end;
>    part2 := postkShuffle( k, v );
>    kSh := part1 + part2;
>  end;
>  return presentar(kSh);
>end:
>untrace(kSh); forget(kSh);
```

Procedimiento postkShuffle
parámetros: k : entero
w : polinomio
si w es un polinomio: regresa un polinomio formado por el postkShuffle de cada término de w.
si w es un monomio: regresa un polinomio formado por el coeficiente de w por el postkShufflé de w.
si w es una palabra: regresa el encadenamiento del primer simbolo de w con el k-Shufflé del resto w.

```
>postkShuffle := proc( k::integer, w::{name,`*`,`+`} )
>option remember;
>local pkSh;
>    if type(w,`+`) then pkSh := map2( postkShuffle, k, w );
>  elif type(w,`*`) then pkSh := w/palabras(w)[1]*postkShuffle(k,palabras(w)[1]);
>  else pkSh := encadenar(substring(w, 1), kSh(k, substring(w,2..length(w)) ) );
>  end;
>  return presentar(pkSh);
>end:
>untrace(postkShuffle);
```

Ejemplo:

```
>Shuffle(b,a);
```

$$A_9\,aa + A_{10}\,ab + (1 + A_{11})\,ba + A_{12}\,bb$$

Las condiciones para que el Álgebra Shufflé (y también el Álgebra Simétrica Cuántica) sea conmutativa son

```
>conmutatividad := []:
>for u in simbolos do
>  for v in simbolos do
>    if u<>v then
>      conmutatividad := [ op(conmutatividad), Shuffle(u,v)-Shuffle(v,u) ];
>    end;
>  od;
>od;
>conmutatividad := map( coeffs, conmutatividad, palabras(conmutatividad) ):
>conmutatividad := convert( conmutatividad, set ):
>conmutatividad := solve( conmutatividad ):
>subs(conmutatividad, evalm(R));
```

$$\begin{bmatrix} A_1 & A_2 & A_3 & A_4 \\ A_9 & -1+A_{10} & 1+A_{11} & A_{12} \\ A_9 & A_{10} & A_{11} & A_{12} \\ A_{13} & A_{14} & A_{15} & A_{16} \end{bmatrix}$$

Las condiciones para que el Álgebra Shufflé (y también el Álgebra Simétrica Cuántica) sea anticonmutativa son

```
>anticonmut := []:
>for u in simbolos do
>  for v in simbolos do
>    anticonmut := [ op(anticonmut), Shuffle(u,v)+Shuffle(v,u) ];
>  od;
>od;
>anticonmut := map( coeffs, anticonmut, palabras(anticonmut) ):
>anticonmut := convert( anticonmut, set ):
>anticonmut := solve( anticonmut ):
>subs(anticonmut, evalm(R));
```

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ -A_9 & -1-A_{10} & -1-A_{11} & -A_{12} \\ A_9 & A_{10} & A_{11} & A_{12} \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

El producto Shuffle es asociativo si y solo si τ satisface la relación de trenza

$$\tau_i \tau_{i+1} \tau_i = \tau_{i+1} \tau_i \tau_{i+1}$$

$$a \otimes (b \otimes c) = (a \otimes b) \otimes c$$

```
>trenza := []:
>for x in simbolos do
>  for y in simbolos do
>    for z in simbolos do
>    trenza := [ op(trenza), presentar(tau(1,tau(2,tau(1,cat(x,y,z))))) -
tau(2,tau(1,tau(2,cat(x,y,z))))) ];
>    od;
>  od;
>od;
```

Calculamos número de ecuaciones y número de términos en cada una

```
>nops(trenza),nops(trenza[1]);
```

$$8, 8$$

```
>asociatividad := []:
>for x in simbolos do
>  for y in simbolos do
>    for z in simbolos do
>      asociatividad := [ op(asociatividad),
```

```
>                          presentar(Shuffle( Shuffle( x, y ), z ) - Shuffle( x, Shuffle( y,
z ))) ]:
>     od;
>   od;
>od:
```

Calculamos número de ecuaciones y número de términos en cada una

```
>nops(asociatividad),nops(asociatividad[1]);
```

$$8, 8$$

Verificamos que ambas condiciones sean las mismas

```
>for ii from 1 to nops(trenza) do
>  ii,presentar(trenza[ii] - asociatividad[ii]);
>od;
```

$$1, 0$$

$$1, 0$$

$$3, 0$$

$$4, 0$$

$$5, 0$$

$$6, 0$$

$$7, 0$$

$$8, 0$$

Construimos la Ecuación de Yang-Baxter en forma coordenada:

```
>YBE := map( coeffs, trenza, palabras(trenza) );
```

$$YBE := [-A_2^2 A_{13} - A_4 A_6 A_{13} - A_1 A_2 A_5 + A_3^2 A_{13} + A_1 A_3 A_9 - A_4 A_5^2 + A_4 A_9^2 + A_2 A_{11} A_5$$
$$- A_3 A_6 A_9 + A_4 A_{11} A_{13}, -A_1 A_2 A_{11} - A_1 A_2 A_7 + A_1 A_3 A_{10} - A_3^2 A_5 + A_4 A_9 A_{10}$$
$$+ A_2 A_{11} A_6 + A_1^2 A_2 + A_2^2 A_9 + A_3^2 A_{14} - A_4 A_5 A_7 - A_3 A_6 A_{11} - A_2^2 A_{15} + A_4 A_{11} A_{14}$$
$$+ A_1 A_3 A_6 - A_4 A_6 A_{15} - A_1^2 A_3, A_3^2 A_{15} + A_4 A_{11} A_{15} + A_1 A_3 A_{11} + A_2 A_9 A_3 + A_4 A_9 A_{11}$$
$$- A_4 A_5 A_7 - A_4 A_8 A_{13} - A_2 A_4 A_{13} - A_3 A_5 A_2 - A_1 A_4 A_9 + A_2 A_{11} A_7 - A_3 A_8 A_9, A_1^2 A_4$$
$$+ A_1 A_3 A_8 + A_3^2 A_{16} + A_2 A_9 A_4 + A_1 A_3 A_{12} - A_3 A_7 A_2 + A_2 A_{11} A_8 - A_4 A_7^2 - A_1 A_4 A_{11}$$
$$+ A_4 A_9 A_{12} - A_4 A_8 A_{15} - A_1 A_3^2 - A_2 A_4 A_{15} + A_4 A_{11} A_{16} - A_3^2 A_7 - A_3 A_8 A_{11},$$
$$A_4 A_{12} A_{13} + A_1 A_4 A_5 - A_3 A_6 A_{10} - A_4 A_5 A_6 - A_1 A_2 A_6 + A_2 A_{12} A_5 - A_3 A_5 A_2$$
$$- A_4 A_6 A_{14} + A_4 A_9 A_{10} + A_3 A_4 A_{13} - A_2^2 A_{14} + A_2 A_9 A_3, -A_2^2 A_{16} + A_1 A_2^2 - A_3 A_6 A_{12}$$
$$- A_1 A_2 A_{12} + A_1 A_4 A_6 + A_2^2 A_{10} + A_4 A_{12} A_{14} + A_4 A_{10}^2 - A_1^2 A_4 + A_3 A_4 A_{14} - A_3 A_5 A_4$$
$$- A_4 A_6 A_{16} - A_1 A_2 A_8 - A_4 A_5 A_8 + A_2 A_{12} A_6 + A_2 A_{10} A_3, A_2 A_{10} A_3 + A_2 A_{12} A_7$$
$$+ A_4 A_{10} A_{11} + A_4 A_{12} A_{15} + A_3 A_2 A_{11} - A_1 A_4 A_{10} - A_4 A_7 A_6 - A_2 A_4 A_{14} - A_4 A_8 A_{14}$$
$$+ A_3 A_4 A_{15} + A_1 A_4 A_7 - A_3 A_7 A_2 - A_3 A_2 A_6 - A_3 A_8 A_{10}, A_4 A_{12} A_{16} + A_4 A_{10} A_{12}$$
$$+ A_3 A_2 A_{12} - A_2 A_4 A_{16} - A_3 A_2 A_8 + A_1 A_2 A_4 + A_2 A_{10} A_4 - A_1 A_3 A_4 - A_1 A_4 A_{12}$$
$$+ A_2 A_{12} A_8 + A_1 A_4 A_8 - A_3 A_8 A_{12} - A_3 A_7 A_4 - A_4 A_8 A_{16} + A_3 A_4 A_{16} - A_4 A_7 A_8,$$

$$-A_8 A_5^{\,2} - A_5 A_2 A_9 - A_6 A_1 A_5 + A_3 A_7 A_{13} + A_3 A_5 A_9 - A_6 A_2 A_{13} + A_4 A_{15} A_{13}$$
$$- A_8 A_6 A_{13} - A_7 A_6 A_9 + A_4 A_{13} A_9 + A_2 A_{13} A_1 + A_2 A_{15} A_5, A_2^{\,2} A_{13} + A_1 A_2 A_5$$
$$- A_8 A_6 A_{15} + A_4 A_{15} A_{14} - A_7 A_5 A_3 + A_3 A_7 A_{14} - A_2 A_{11} A_5 + A_3 A_5 A_{10} - A_7 A_6 A_{11}$$
$$- A_8 A_5 A_7 + A_4 A_{13} A_{10} - A_1 A_3 A_5, A_3 A_7 A_{15} - A_8^{\,2} A_{13} + A_4 A_{11} A_{13} - A_8 A_5 A_7$$
$$+ A_4 A_{15}^{\,2} + A_2 A_{15} A_7 - A_4 A_5 A_9 + A_2 A_{13} A_3 + A_3 A_5 A_{11} - A_3 A_6 A_5 - A_4 A_6 A_{13}$$
$$- A_8 A_7 A_9, -A_3^{\,2} A_5 + A_1 A_4 A_5 + A_4 A_{12} A_{13} + A_2 A_{15} A_8 - A_7^{\,2} A_3 + A_3 A_7 A_{16}$$
$$+ A_4 A_{15} A_{16} - A_8 A_7^{\,2} + A_2 A_4 A_{13} - A_3 A_6 A_7 - A_4 A_5 A_{11} + A_1 A_7 A_8 - A_4 A_6 A_{15}$$
$$- A_8^{\,2} A_{15} + A_3 A_5 A_{12} - A_8 A_7 A_{11}, -A_8 A_5 A_6 - A_5 A_2 A_{10} + A_3 A_8 A_{13} + A_2 A_{16} A_5$$
$$+ A_2 A_{14} A_1 - A_6 A_2 A_{14} + A_6 A_1^{\,2} - A_7 A_6 A_{10} - A_6^{\,2} A_1 + A_3 A_6 A_9 - A_7 A_5 A_2 - A_8 A_6 A_{14}$$
$$+ A_4 A_{14} A_9 + A_4 A_{16} A_{13} - A_1 A_2 A_5 + A_1 A_8 A_5, -A_7 A_6 A_{12} + A_2^{\,2} A_{14} - A_8^{\,2} A_5$$
$$+ A_4 A_{16} A_{14} + A_4 A_{14} A_{10} + A_1 A_2 A_6 + A_3 A_6 A_{10} - A_1 A_4 A_5 - A_4 A_5 A_7 - A_2 A_{12} A_5$$
$$- A_8 A_6 A_{16} + A_3 A_8 A_{14}, -A_8^{\,2} A_{14} - A_7^{\,2} A_2 + A_1 A_7 A_8 + A_2 A_{14} A_3 + A_2 A_{16} A_7$$
$$+ A_3 A_8 A_{15} + A_4 A_{15} A_{16} - A_8 A_7 A_6 - A_8 A_7 A_{10} - A_4 A_5 A_{10} - A_4 A_6 A_{14} + A_3 A_6 A_{11}$$
$$+ A_4 A_{11} A_{14} + A_1 A_3 A_6 - A_3 A_5 A_2 - A_3 A_6^{\,2}, -A_4 A_7^{\,2} - A_8^{\,2} A_7 + A_4 A_{12} A_{14} + A_2 A_{16} A_8$$
$$+ A_4 A_{16}^{\,2} + A_1 A_8^{\,2} + A_2 A_4 A_{14} + A_3 A_6 A_{12} - A_8^{\,2} A_{16} + A_1 A_4 A_6 - A_3 A_5 A_4 + A_3 A_8 A_{16}$$
$$- A_8 A_7 A_{12} - A_3 A_6 A_8 - A_4 A_5 A_{12} - A_4 A_6 A_{16}, A_8 A_9^{\,2} - A_6 A_{11} A_9 - A_{10} A_1 A_5 - A_{12} A_5^{\,2}$$
$$- A_{11} A_5 A_1 + A_8 A_{11} A_{13} - A_2 A_9^{\,2} + A_1 A_7 A_9 - A_9 A_1^{\,2} - A_6 A_{12} A_{13} - A_2 A_{10} A_{13}$$
$$+ A_3 A_7 A_{13} + A_6 A_9 A_1 + A_3 A_5^{\,2} + A_6 A_{11} A_5 + A_1^{\,2} A_5, A_3 A_7 A_{14} - A_{12} A_5 A_7 + A_6 A_9 A_2$$
$$- A_6 A_{11}^{\,2} - A_6 A_{12} A_{15} - A_1 A_3 A_9 + A_3 A_6 A_5 - A_2 A_9 A_{11} + A_8 A_9 A_{10} + A_1 A_2 A_5$$
$$- A_2 A_{10} A_{15} + A_6^{\,2} A_{11} - A_3 A_5 A_{11} + A_8 A_{11} A_{14}, A_7 A_6 A_{11} - A_3 A_5 A_{10} - A_4 A_{13} A_{10}$$
$$- A_4 A_9^{\,2} + A_3 A_6 A_9 - A_8 A_{12} A_{13} - A_{12} A_5 A_7 + A_1 A_3 A_5 + A_3 A_7 A_{15} + A_7 A_5 A_3$$
$$+ A_8 A_{11} A_{15} - A_1 A_3 A_9, -A_{12} A_7^{\,2} + A_3 A_7 A_{16} + A_3 A_5 A_8 - A_9 A_3^{\,2} + A_8 A_9 A_{12} + A_1 A_4 A_5$$
$$- A_8 A_{11}^{\,2} - A_4 A_9 A_{11} - A_{10} A_3 A_7 + A_1 A_7 A_{12} - A_8 A_{12} A_{15} - A_4 A_{10} A_{15} + A_6 A_{11} A_8$$
$$+ A_6 A_9 A_4 + A_8 A_{11} A_{16} - A_{11} A_7 A_3, A_4 A_5^{\,2} + A_8 A_{12} A_{13} - A_6 A_{11} A_{10} - A_2 A_9 A_{10}$$

$$+A_7A_2A_9-A_2A_{11}A_5-A_2A_9A_1-A_2A_{10}A_{14}+A_4A_7A_{13}+A_8A_9A_{10}+A_1A_2A_5$$
$$-A_6A_{12}A_{14},\ A_8A_{10}^{\ 2}-A_{10}A_1A_8-A_6A_{11}A_{12}+A_8A_{12}A_{14}+A_4A_5A_6-A_{12}A_5A_8$$
$$+A_5A_2^{\ 2}-A_6A_{12}A_{16}+A_6^{\ 2}A_{12}+A_6A_{10}A_2-A_2A_{10}A_{16}+A_4A_7A_{14}+A_7A_2A_{10}$$
$$-A_2A_9A_{12}-A_1A_4A_9-A_4A_5A_{11},\ A_3A_5A_2+A_8A_{12}A_{15}-A_2A_9A_3+A_4A_5A_7$$
$$-A_4A_{14}A_{10}-A_8A_{12}A_{14}+A_4A_7A_{15}-A_4A_9A_{10},\ -A_4A_{10}A_{16}+A_4A_7A_{16}-A_9A_3A_4$$
$$+A_4A_5A_8+A_6A_{12}A_8+A_8A_{10}A_{12}+A_5A_2A_4+A_2A_{12}A_7+A_6A_{10}A_4-A_3A_8A_{10}$$
$$-A_8A_{11}A_{12}-A_{11}A_7A_4-A_8A_7A_{12}-A_4A_9A_{12},\ A_6A_{13}A_1-A_6A_{15}A_9+A_8A_{13}A_9$$
$$-A_2A_{14}A_{13}-A_{15}A_5A_1+A_7A_5A_9+A_7A_5^{\ 2}-A_6A_{16}A_{13}+A_8A_{15}A_{13}-A_{13}A_1^{\ 2}$$
$$-A_{16}A_5^{\ 2}+A_6A_{15}A_5-A_2A_{13}A_9+A_5^{\ 2}A_1-A_{14}A_1A_5+A_7^{\ 2}A_{13},\ A_6A_2A_{13}-A_2A_{14}A_{15}$$
$$+A_6^{\ 2}A_{15}+A_7^{\ 2}A_{14}-A_{14}A_1A_7+A_7A_5A_{10}-A_{13}A_1A_3+A_8A_{13}A_{10}-A_{16}A_5A_7$$
$$+A_7A_5A_6+A_5^{\ 2}A_2+A_8A_{15}A_{14}-A_6A_{15}A_{11}-A_6A_{16}A_{15}-A_2A_{13}A_{11}-A_{15}A_5A_3,$$
$$-A_{14}A_3A_5-A_8A_{16}A_{13}+A_7^{\ 2}A_5+A_6A_{13}A_3-A_4A_{14}A_{13}-A_{16}A_5A_7-A_{13}A_1A_3$$
$$-A_{15}A_7A_1+A_3A_5^{\ 2}+A_8A_{11}A_{13}+A_7A_5A_{11}+A_8A_{15}^{\ 2}+A_7^{\ 2}A_{15}-A_4A_{13}A_9$$
$$-A_8A_{15}A_9+A_6A_{15}A_7,\ -A_4A_{15}A_{14}-A_8A_{11}A_{15}+A_4A_5^{\ 2}-A_3A_7A_{14}+A_8A_6A_{15}$$
$$-A_3^{\ 2}A_{13}+A_4A_6A_{13}+A_8A_{12}A_{13}+A_8A_5A_7-A_4A_{11}A_{13}-A_3A_7A_{15}+A_{12}A_5A_7,$$
$$A_6A_1A_5-A_6A_{16}A_{14}-A_2A_{14}^{\ 2}-A_2A_{10}A_{13}+A_8A_{14}A_9+A_7A_6A_9-A_2A_{13}A_1$$
$$-A_6A_{15}A_{10}+A_8A_7A_{13}-A_2A_{15}A_5+A_8A_{16}A_{13}+A_8A_5^{\ 2},\ -A_{16}A_5A_8-A_2A_{13}A_{12}$$
$$+A_8A_5A_6+A_6A_2A_{14}+A_6^{\ 2}A_{16}-A_2A_{14}A_{16}-A_6A_{12}A_{15}+A_8A_{14}A_{10}-A_6A_{16}^{\ 2}$$
$$+A_7A_6A_{10}-A_{13}A_1A_4-A_{15}A_5A_4-A_{14}A_1A_8+A_8A_7A_{14}+A_5A_6A_2+A_8A_{16}A_{14},$$
$$-A_8A_{15}A_{10}-A_2A_{13}A_3-A_4A_{13}A_{10}-A_4A_{14}^{\ 2}-A_8A_{16}A_{14}+A_8A_{11}A_{14}-A_2A_{15}A_7$$
$$+A_8A_5A_7+A_7A_6A_{11}+A_3A_6A_5+A_8A_{15}A_{16}+A_8A_7A_{15},\ -A_4A_7A_{15}-A_4A_{12}A_{13}$$
$$-A_3A_4A_{13}+A_7A_6A_{12}-A_3A_8A_{14}+A_8A_{12}A_{14}-A_8A_{12}A_{15}-A_4A_{16}A_{14}+A_4A_6A_{14}$$
$$+A_4A_5A_6+A_8^{\ 2}A_5+A_8A_6A_{16},\ A_{10}A_{11}A_5-A_4A_{13}A_5+A_{12}A_{11}A_{13}-A_{13}A_1A_3$$
$$+A_{11}A_3A_{13}-A_2A_{10}A_{13}-A_4A_{14}A_{13}+A_{12}A_9^{\ 2}-A_{14}A_3A_9-A_5A_2A_9+A_{11}A_1A_9$$
$$+A_3A_5A_9,\ A_{12}A_{11}A_{14}-A_4A_7A_{13}+A_2A_9A_{10}+A_{12}A_9A_{10}-A_3^{\ 2}A_{13}-A_4A_{15}A_{14}$$

$$+A_2A_9A_1+A_6A_{11}A_{10}+A_3A_6A_9-A_1A_3A_9-A_7A_2A_9-A_2A_{10}A_{15},-A_{15}A_5A_4$$
$$+A_{11}A_3A_{15}+A_{12}A_{11}A_{15}-A_2A_{13}A_{12}+A_9A_3A_7-A_1A_{12}A_9-A_{11}A_1^{\,2}+A_{10}A_{11}A_7$$
$$+A_{11}^{\,2}A_1+A_{10}A_9A_3-A_4A_{16}A_{13}-A_3A_{16}A_9+A_{12}A_9A_{11}-A_2A_{11}A_5-A_3A_{15}A_1$$
$$+A_1A_3A_9,A_8A_{10}A_{11}-A_4A_7A_{15}+A_{12}^{\,2}A_9-A_3^{\,2}A_{15}-A_2A_{11}A_7+A_3A_8A_9$$
$$-A_4A_{15}A_{16}-A_2A_{12}A_{15}-A_1A_3A_{11}+A_{12}A_{11}A_{16}+A_1A_4A_9+A_4A_9A_{10},A_2A_9A_{11}$$
$$-A_6A_9A_2-A_4A_{14}^{\,2}+A_{12}A_{10}A_5-A_2A_{13}A_3-A_4A_6A_{13}+A_4A_5A_9-A_{14}A_3A_{10}$$
$$+A_{12}^{\,2}A_{13}+A_{12}A_9A_{10}-A_2A_{10}A_{14}+A_4A_{11}A_{13},A_{10}^{\,2}A_2-A_{14}A_3A_{12}+A_{12}A_{10}^{\,2}$$
$$+A_{12}A_{10}A_6-A_3A_4A_{13}+A_{12}^{\,2}A_{14}+A_6A_9A_4-A_1A_4A_9+A_2^{\,2}A_9-A_4A_8A_{13}$$
$$-A_2A_9A_8-A_{10}A_1A_{12}-A_4A_{16}A_{14}-A_2A_{10}A_{16}+A_4A_{11}A_{14}+A_2A_{11}A_{10},A_4A_{11}A_{15}$$
$$-A_1A_2A_{11}+A_2A_{11}^{\,2}-A_3A_{16}A_{10}+A_{12}A_{10}A_{11}-A_{10}A_1A_{12}-A_2A_{11}A_6-A_2A_{12}A_{14}$$
$$+A_{12}A_{10}A_7+A_4A_9A_7-A_4A_6A_{15}+A_{12}^{\,2}A_{15}+A_2A_9A_3+A_{10}^{\,2}A_3-A_3A_{15}A_2$$
$$-A_4A_{16}A_{14},-A_2A_{11}A_8+A_4A_{10}^{\,2}-A_4A_8A_{15}+A_{12}^{\,2}A_{10}+A_2A_9A_4-A_3A_4A_{15}$$
$$+A_4A_{11}A_{16}+A_4A_9A_8+A_2A_{11}A_{12}+A_{12}^{\,2}A_{16}-A_2A_{12}A_{16}-A_1A_4A_{11}-A_4A_{16}^{\,2}$$
$$-A_3A_{16}A_{12}+A_8A_{10}A_{12}-A_1A_{12}^{\,2},-A_7A_{13}A_1+A_{10}A_{15}A_5-A_8A_{14}A_{13}-A_5A_{10}A_9$$
$$-A_7A_{14}A_9-A_6A_9A_5+A_{11}A_7A_{13}+A_7A_5A_9+A_{11}A_5A_9+A_{10}A_{13}A_1-A_6A_{10}A_{13}$$
$$+A_{12}A_{13}A_9-A_8A_{13}A_5+A_{12}A_{15}A_{13},A_{12}A_{13}A_{10}-A_8A_7A_{13}-A_3A_7A_{13}+A_2A_{10}A_{13}$$
$$+A_{12}A_{15}A_{14}-A_8A_{15}A_{14}+A_5A_2A_9-A_3A_5A_9,A_{11}^{\,2}A_5+A_9A_7^{\,2}+A_{12}A_{15}^{\,2}-A_6A_{11}A_5$$
$$-A_{11}A_5A_1+A_3A_5A_9+A_{10}A_{13}A_3-A_8A_{16}A_{13}+A_{11}A_7A_{15}-A_8A_{15}A_5+A_{10}A_{15}A_7$$
$$-A_{16}A_7A_9-A_{15}A_7A_1-A_{12}A_5A_9+A_{12}A_{11}A_{13}-A_6A_{12}A_{13},A_4A_{13}A_{10}+A_8A_{15}A_{10}$$
$$-A_3A_7A_{15}+A_4A_5A_9+A_{12}^{\,2}A_{13}+A_{12}A_{15}A_{16}-A_8A_7A_{15}-A_3A_5A_{11}-A_6A_{12}A_{15}$$
$$-A_7A_6A_{11}+A_8A_7A_9-A_8A_{15}A_{16},A_6A_9A_1-A_6A_{10}A_{14}+A_8A_9A_5-A_5A_2A_9$$
$$-A_6^{\,2}A_9+A_{10}A_{16}A_5-A_5A_{10}^{\,2}-A_8A_{14}^{\,2}-A_8A_6A_{13}+A_6A_{11}A_9+A_{12}A_{16}A_{13}$$
$$+A_{10}A_{14}A_1-A_7A_{14}A_{10}-A_7A_{13}A_2+A_8A_{11}A_{13}+A_{12}A_{14}A_9,A_{12}A_{16}A_{14}-A_4A_7A_{13}$$
$$+A_6A_{11}A_{10}-A_4A_5A_9+A_8A_{11}A_{14}+A_{12}A_{14}A_{10}+A_6A_9A_2-A_{12}A_{10}A_5-A_8^{\,2}A_{13}$$
$$+A_2A_{10}A_{14}-A_7A_{14}A_{12}-A_8A_{16}A_{14},-A_2A_{15}A_7+A_8A_{11}A_{15}+A_6A_{11}^{\,2}-A_6A_{12}A_{14}$$
$$+A_{14}A_3A_{10}-A_8A_{16}A_{14}-A_{12}A_{10}A_5+A_{12}A_{11}A_{14}+A_8A_7A_9-A_6^{\,2}A_{11}+A_3A_6A_9$$
$$+A_{12}A_{15}A_{16}-A_8A_6A_{15}-A_2A_{11}A_5,-A_4A_7A_{15}+A_6A_9A_4+A_8^{\,2}A_9+A_6A_{11}A_{12}$$

$$+A_4A_{14}A_{10}+A_8A_{11}A_{16}+A_{10}A_{16}A_8-A_4A_5A_{11}-A_6A_{12}A_{16}+A_{12}A_{16}^2-A_8A_{16}^2$$
$$-A_{16}A_7A_{12}-A_8^2A_{15}+A_{12}^2A_{14}-A_{12}^2A_5-A_6A_{11}A_8, A_{16}A_{11}A_{13}-A_{14}A_{11}A_9$$
$$+A_{15}A_1A_9+A_{14}A_{11}A_5+A_3A_{15}A_{13}-A_{12}A_{13}A_5-A_{11}A_{13}A_1+A_{14}A_9A_1-A_{10}A_9^2$$
$$-A_5A_{10}A_9-A_{10}^2A_{13}-A_{12}A_{14}A_{13}+A_{13}A_1^2-A_9^2A_1+A_{16}A_9^2+A_{13}A_3A_5,$$
$$-A_{12}A_{15}A_{14}-A_{14}A_{11}^2+A_{16}A_{11}A_{14}+A_{14}A_{11}A_6-A_9^2A_3-A_{10}A_9A_{11}+A_{14}A_9A_2$$
$$+A_3A_{15}A_{14}+A_2A_{13}A_1+A_{16}A_9A_{10}+A_6A_{13}A_3-A_{10}^2A_{15}-A_{11}A_3A_{13}+A_{15}A_1A_{10}$$
$$-A_{10}A_9A_7-A_{12}A_{13}A_7, A_{13}A_1A_3-A_{12}A_9^2+A_{16}A_{11}A_{15}-A_{12}A_{13}A_{10}-A_{11}A_1A_9$$
$$-A_{10}A_{11}A_5-A_{12}A_{15}A_5+A_3A_7A_{13}-A_{12}A_{16}A_{13}+A_{11}A_7A_{14}+A_3A_{15}^2+A_{14}A_3A_9,$$
$$A_3A_8A_{13}+A_{16}A_9A_{12}+A_4A_{14}A_9+A_{16}^2A_{11}-A_{11}A_3A_{15}-A_{16}A_{11}^2-A_9A_{11}A_3$$
$$-A_{12}A_{15}A_{16}-A_{10}A_{11}A_7+A_8A_{11}A_{14}-A_{12}A_{15}A_7+A_{15}A_1A_{12}-A_{12}A_{10}A_{15}$$
$$+A_3A_{15}A_{16}+A_{13}A_1A_4-A_{12}A_9A_{11}, A_2A_{13}A_1-A_2A_9^2+A_4A_{13}A_5-A_{10}^2A_{14}$$
$$+A_4A_{15}A_{13}-A_{10}A_9A_6-A_{12}A_{14}^2+A_{12}A_{14}A_5+A_{12}A_{16}A_{13}-A_2A_{13}A_{11}+A_{10}A_{14}A_1$$
$$-A_6A_{12}A_{13}-A_{14}A_{11}A_{10}+A_2A_{15}A_9+A_{16}A_9A_{10}-A_{10}^2A_9, -A_{12}A_{11}A_{14}+A_6A_{12}A_{14}$$
$$+A_4A_6A_{13}-A_8A_9A_{10}-A_{12}A_9A_{10}+A_2A_{10}A_{14}-A_4A_{11}A_{13}+A_2A_{10}A_{15}-A_4A_9^2$$
$$+A_2^2A_{13}+A_4A_{15}A_{14}-A_8A_{12}A_{13}, A_4A_{15}^2+A_7A_{14}A_{12}-A_6A_{12}A_{15}-A_6A_{11}A_{10}$$
$$-A_{12}A_{16}A_{14}+A_{14}A_3A_{10}-A_{12}A_{14}A_{10}+A_4A_7A_{13}-A_{12}A_9A_{10}-A_2A_9A_{11}$$
$$+A_{12}A_{15}A_{16}+A_2A_{13}A_3, -A_8A_{12}A_{15}+A_4A_{14}A_{10}-A_{12}A_{11}A_{16}+A_8A_{12}A_{14}$$
$$-A_4A_{11}A_{15}+A_4A_8A_{13}-A_4A_9A_{11}+A_2A_{12}A_{15}+A_4A_{15}A_{16}-A_{12}^2A_9+A_2A_4A_{13}$$
$$-A_8A_{10}A_{11}, -A_{14}A_9A_5+A_7A_{13}A_5-A_{16}A_{14}A_{13}+A_{15}A_7A_{13}-A_{10}A_{14}A_{13}$$
$$-A_{16}A_{13}A_5+A_{16}A_{13}A_9-A_{10}A_{13}A_9-A_{14}A_{15}A_9-A_{13}A_9A_1-A_{15}A_{13}A_1+A_{14}A_{13}A_1$$
$$+A_{15}A_5A_9+A_{13}A_5A_1+A_{16}A_{15}A_{13}+A_{14}A_{15}A_5, A_{10}A_{15}A_5-A_{14}A_{15}A_{11}-A_7A_{14}A_9$$
$$+A_{14}A_{15}A_7-A_{14}A_{15}A_{10}+A_{13}A_5A_2-A_{10}A_{13}A_{11}-A_{16}A_{13}A_7+A_7A_{13}A_6$$
$$+A_{16}A_{13}A_{10}+A_2A_{14}A_{13}+A_{14}A_{15}A_6-A_3A_{15}A_{13}-A_{13}A_9A_3, -A_{16}A_{15}A_9$$
$$-A_{11}A_{13}A_1-A_{12}A_{14}A_{13}-A_{15}^2A_1+A_7^2A_{13}-A_{16}^2A_{13}-A_{12}A_{13}A_9-A_{14}A_{11}A_5$$
$$+A_{13}A_3A_5-A_{16}A_{15}A_5+A_{16}A_{11}A_{13}+A_{15}A_5A_{11}+A_{14}A_{15}A_7+A_{14}A_{13}A_3+A_{15}^2A_7$$

$$+A_{16}A_{15}^{\,2},A_{12}A_{15}A_5-A_3A_{15}^{\,2}+A_8A_{15}A_{14}-A_{11}A_7A_{14}+A_{12}A_{16}A_{13}-A_{11}A_3A_{13}$$
$$+A_8A_7A_{13}-A_{16}A_{11}A_{15}-A_{12}A_{11}A_{13}-A_{12}A_{15}A_{14}+A_4A_{13}A_5+A_4A_{14}A_{13},A_{16}A_{14}A_9$$
$$+A_{16}^{\,2}A_{13}-A_{14}A_9A_6-A_6A_{16}A_{13}+A_6A_{13}A_1+A_{16}A_{14}A_5-A_{10}^{\,2}A_{13}+A_8A_{13}A_5$$
$$+A_8A_{15}A_{13}-A_{14}A_{15}A_{10}+A_{14}^{\,2}A_1-A_{16}A_{14}^{\,2}-A_{10}A_{14}^{\,2}+A_6A_{15}A_9-A_{15}A_{13}A_2$$
$$-A_2A_{13}A_9,-A_{12}A_{13}A_{10}-A_8A_{16}A_{13}-A_4A_{13}A_9-A_8A_{14}A_9+A_6A_{15}A_{10}+A_8A_{15}A_{14}$$
$$-A_4A_{15}A_{13}+A_6A_{16}A_{14}+A_6A_2A_{13}+A_2A_{14}^{\,2}+A_8A_6A_{13}-A_{12}A_{15}A_{14},-A_{16}^{\,2}A_{14}$$
$$-A_6A_{16}A_{15}+A_{16}^{\,2}A_{15}-A_{12}A_{13}A_{10}-A_2A_{13}A_{11}+A_{14}^{\,2}A_3-A_{12}A_{14}^{\,2}-A_{16}A_{15}A_{10}$$
$$-A_{14}A_{11}A_6-A_{15}^{\,2}A_2+A_{16}A_{14}A_7+A_8A_{15}^{\,2}+A_6A_{13}A_3+A_8A_7A_{13}+A_6A_{15}A_{11}$$
$$+A_{16}A_{11}A_{14},-A_{12}^{\,2}A_{13}+A_4A_6A_{13}-A_4A_{15}^{\,2}-A_4A_{11}A_{13}-A_8A_{11}A_{14}+A_8A_{16}A_{14}$$
$$-A_{12}A_{15}A_{16}+A_4A_{14}^{\,2}+A_6A_{12}A_{15}+A_8^{\,2}A_{13}]$$

```
> YBE := convert(YBE,set):
```

y la resolvemos:

- En Maple 4 tarda unos minutos (en dimensión 2).
- En Maple 7 y 8 no termina !

```
> solve(YBE);
Warning, computation interrupted

>
```

11.4 *Espacio tangente a una solución de YBE en Maple.*

```
> restart;
```
La dimension en YBE
```
> dim := 2;
```
$$dim := 2$$

Definimos la ecuacion de Yang-Baxter
```
> EQN := (i,j,k,l,m,n)->Sum(Sum(Sum( A[i,j,p,q]*A[p,k,l,r]*A[q,r,m,n], p=0..dim-1),
q=0..dim-1), r=0..dim-1 )
>       - Sum(Sum(Sum( A[j,k,q,r]*A[i,r,p,n]*A[p,q,l,m], p=0..dim-1), q=0..dim-1), r=0..dim-1
) = 0;
```
$$EQN := (i,j,k,l,m,n) \to \left(\sum_{r=0}^{dim-1}\left(\sum_{q=0}^{dim-1}\left(\sum_{p=0}^{dim-1} A_{i,j,p,q}\,A_{p,k,l,r}\,A_{q,r,m,n}\right)\right)\right)$$
$$-\left(\sum_{r=0}^{dim-1}\left(\sum_{q=0}^{dim-1}\left(\sum_{p=0}^{dim-1} A_{j,k,q,r}\,A_{i,r,p,n}\,A_{p,q,l,m}\right)\right)\right)=0$$

```
> YBE := {seq(seq(seq(seq(seq(seq( EQN(i,j,k,l,m,n), i=0..dim-1), j=0..dim-1), k=0..dim-1),
l=0..dim-1), m=0..dim-1), n=0..dim-1)}:
> YBE := map(value,YBE):
```
Las variables en YBE
```
> variables := [seq( seq( seq( seq( A[i,j,k,l], l=0..dim-1), k=0..dim-1), j=0..dim-1),
i=0..dim-1)];
```
$$variables := [A_{0,0,0,0}, A_{0,0,0,1}, A_{0,0,1,0}, A_{0,0,1,1}, A_{0,1,0,0}, A_{0,1,0,1}, A_{0,1,1,0}, A_{0,1,1,1}, A_{1,0,0,0},$$
$$A_{1,0,0,1}, A_{1,0,1,0}, A_{1,0,1,1}, A_{1,1,0,0}, A_{1,1,0,1}, A_{1,1,1,0}, A_{1,1,1,1}]$$

Los diferenciales de las variables como variables adicionales d_i
```
> Dvariables := { seq([op(map(D,variables))][j]=[seq(d[i-1],
i=1..nops(variables))][j],j=1..nops(variables)) };
```
$$Dvariables := \{D(A_{0,0,0,0}) = d_0, D(A_{0,0,0,1}) = d_1, D(A_{0,0,1,0}) = d_2, D(A_{0,0,1,1}) = d_3,$$
$$D(A_{0,1,0,0}) = d_4, D(A_{0,1,0,1}) = d_5, D(A_{0,1,1,0}) = d_6, D(A_{0,1,1,1}) = d_7, D(A_{1,0,0,0}) = d_8,$$
$$D(A_{1,0,0,1}) = d_9, D(A_{1,0,1,0}) = d_{10}, D(A_{1,0,1,1}) = d_{11}, D(A_{1,1,0,0}) = d_{12},$$
$$D(A_{1,1,0,1}) = d_{13}, D(A_{1,1,1,0}) = d_{14}, D(A_{1,1,1,1}) = d_{15}\}$$

Calculamos el diferencial de YBE donde aparecerán los diferenciales de las variables
```
> DYBE := D(YBE):
```
Cambiamos el nombre de los diferenciales
```
> DYBE := subs(Dvariables,DYBE):
```
Ahora DYBE tiene las siguientes variables
```
> indets(DYBE);
```
$$\{A_{0,0,0,0}, A_{0,0,1,0}, A_{0,0,0,1}, A_{0,0,1,1}, A_{1,0,0,0}, A_{0,1,0,0}, A_{1,0,0,1}, A_{1,1,0,0}, A_{0,1,1,0}, A_{1,0,1,0},$$
$$A_{1,0,1,1}, A_{1,1,1,0}, A_{0,1,0,1}, A_{0,1,1,1}, A_{1,1,0,1}, A_{1,1,1,1}, d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9,$$
$$d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}\}$$

Sustituimos alguna solución de YBE y la presentamos como una matriz Q
```
> Q:=matrix(dim*dim,dim*dim):
> for ii from 0 to dim-1 do
> for jj from 0 to dim-1 do
> for kk from 0 to dim-1 do
> for ll from 0 to dim-1 do
>   assign(x[ii*dim^3+jj*dim^2+kk*dim+ll]=A[ii,jj,kk,ll]);
>   assign(Q[ii*dim+jj+1,kk*dim+ll+1]=x[ii*dim^3+jj*dim^2+kk*dim+ll]);
> od;od;od;od;
> Datos:=[
```

```
> x[ 0]=  3.8124061263256
,x[ 1]=  1.18063361559907
,x[ 2]=  1.18063361559907
,x[ 3]=  1.38948874587342
,x[ 4]=  2.91626792333692
,x[ 5]= -0.295642040435327
,x[ 6]=  3.43215829735266
,x[ 7]= -3.65413190360989
,x[ 8]= -4.03711514634775
,x[ 9]=  1.78575041839071
,x[10]= -1.94204991939728
,x[11]=  2.10165124624652
,x[12]=  0.891288817004647
,x[13]=  1.0489585952058
,x[14]=  1.0489585952058
,x[15]=  4.04375476314957
>]:
> subs(Datos,evalm(Q));
```

$$\begin{bmatrix} 3.8124061263256, & 1.18063361559907, & 1.18063361559907, & 1.38948874587342 \\ 2.91626792333692, & -0.295642040435327, & 3.43215829735266, & -3.65413190360989 \\ -4.03711514634775, & 1.78575041839071, & -1.94204991939728, & 2.10165124624652 \\ 0.891288817004647, & 1.0489585952058, & 1.0489585952058, & 4.04375476314957 \end{bmatrix}$$

Inicia aquí el cálculo de la dimensión del espacio tangente a la solución dada.
Definimos el nuevo sistema de ecuaciones sustituyendo la solución dada en el diferencial de YBE.

```
> sistema := subs( Datos, DYBE ):
```

El nuevo sistema de ecuaciones tiene el siguiente número de ecuaciones y las siguientes variables

```
> nops(sistema); indets(sistema);
```

64

$$\{d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}\}$$

Construimos la matriz de coeficientes del sistema de ecuaciones

```
> M := matrix( nops(sistema), dim^4 ):
> for ii from 1 to nops(sistema) do
> for jj from 1 to dim^4 do
>   M[ii,jj]:= coeff( lhs(sistema[ii]) , d[jj-1]);
> od;
> od:
```

La dimensión del espacio tangente a la solución dada es:

```
> DimEspTan:=dim^4-linalg[rank](M);
```

$$DimEspTan := 5$$

Veremos cuantas ecuaciones NO se cumplen

```
> subs(Datos,value(YBE));
```

$$\{-0.10\,10^{-7}=0, -0.11\,10^{-7}=0, 0.28\,10^{-7}=0, -0.23\,10^{-7}=0, 0.3\,10^{-7}=0, 0.12\,10^{-7}=0,$$
$$0.=0, 0.6\,10^{-7}=0, 0.10\,10^{-7}=0, -0.1\,10^{-7}=0, -0.6\,10^{-8}=0, 0.18\,10^{-7}=0,$$
$$-0.13\,10^{-7}=0, 0.2\,10^{-7}=0, -0.17\,10^{-7}=0, 0.2\,10^{-8}=0, -0.4\,10^{-7}=0, 0.7\,10^{-8}=0,$$
$$0.11\,10^{-7}=0, -0.42\,10^{-7}=0, 0.4\,10^{-8}=0, 0.1\,10^{-7}=0, -0.26\,10^{-7}=0, -0.1\,10^{-8}=0,$$
$$0.1\,10^{-8}=0, 0.71\,10^{-7}=0, -0.2\,10^{-7}=0, -0.3\,10^{-7}=0, -0.18\,10^{-7}=0, -0.7\,10^{-8}=0,$$
$$-0.15\,10^{-7}=0, -0.5\,10^{-8}=0, 0.6\,10^{-8}=0, 0.16\,10^{-7}=0\}$$

Definimos la expresion para calcular el error

```
> Error := Sum(Sum(Sum(Sum(Sum(Sum(
>   ( (lhs( EQN(i,j,k,l,m,n) ))^2,
> i=0..dim-1), j=0..dim-1), k=0..dim-1), l=0..dim-1), m=0..dim-1), n=0..dim-1);
```

$$Error := \sum_{n=0}^{1}\left(\sum_{m=0}^{1}\left(\sum_{l=0}^{1}\left(\sum_{k=0}^{1}\left(\sum_{j=0}^{1}\left(\sum_{i=0}^{1}\left(\left(\sum_{r=0}^{1}\left(\sum_{q=0}^{1}\left(\sum_{p=0}^{1} A_{i,j,p,q}\,A_{p,k,l,r}\,A_{q,r,m,n}\right)\right)\right)\right)\right)\right)\right)\right)$$

$$-\left(\sum_{r=0}^{1}\left(\sum_{q=0}^{1}\left(\sum_{p=0}^{1} A_{j,k,q,r} A_{i,r,p,n} A_{p,q,l,m}\right)\right)^{2}\right)$$

y calculamos el error de la solución dada

```
> subs(Datos, evalf(Error));
```

$$0.24072\ 10^{-13}$$

```
>
```

Definimos un procedimiento para hacer una Busqueda por gradiente

```
> BusquedaGradiente:=proc(DatosParam, etaini)
> local
Datos,DatosNuevos,GradError,ValError,ValErrorNuevo,ValGradError,EtaGradError,ii,i,eta;
>   Datos := value(DatosParam);
>   GradError := linalg[grad](value(Error), variables ):
>   ValError := subs( Datos, evalf(Error) ); print(ValError);
>   eta := eval(etaini);
>   for ii from 1 to 3 while ValError > 0 do
>     ValGradError := convert(subs(Datos,eval(GradError)),Vector):
>     EtaGradError := eta*ValGradError / sqrt(LinearAlgebra[DotProduct](ValGradError,
ValGradError)):
>     DatosNuevos := [seq(x[i-1]=evalf(rhs(Datos[i])-EtaGradError[i]),i=1..nops(Datos))];
>     if DatosNuevos = Datos then break; end if;
>     ValErrorNuevo := evalf(subs( DatosNuevos, value(Error) ) );
>     if ValErrorNuevo < ValError
>     then
>       Datos := eval( DatosNuevos );
>       ValError := eval( ValErrorNuevo );
>       eta := eta * 1.4142;
>       print( eta, ValError );
>     else
>       eta := eta * 0.1;
>       print( eta );
>     end if;
>     ii := 1:
>   end do;
>   return Datos;
> end:
```

Mejoramos la solución mediante Busqueda por Gradiente

```
> DatosNuevos := BusquedaGradiente( Datos, 1 );
```

$$0.24072\ 10^{-13}$$

$$0.1$$

$$0.01$$

$$0.001$$

$$0.0001$$

$$0.00001$$

$$0.1\ 10^{-5}$$

$$0.1\ 10^{-6}$$

$$0.1\ 10^{-7}$$

$$0.1\ 10^{-8}$$

$$0.14142\ 10^{-8},\ 0.22434\ 10^{-13}$$

$$0.199996164\ 10^{-8},\ 0.17477\ 10^{-13}$$

$$0.2828345751\ 10^{-8},\ 0.16574\ 10^{-13}$$

$$0.2828345751\ 10^{-9}$$

$$0.3999846561\ 10^{-9},\ 0.15470\ 10^{-13}$$

$$0.5656583007\ 10^{-9},\ 0.13995\ 10^{-13}$$

$$0.7999539688\ 10^{-9},\ 0.12285\ 10^{-13}$$

$$0.7999539688\ 10^{-10}$$

$$DatosNuevos := [A_{0,0,0,0} = 3.812406126,\ A_{0,0,0,1} = 1.180633615,\ A_{0,0,1,0} = 1.180633616,$$
$$A_{0,0,1,1} = 1.389488746,\ A_{0,1,0,0} = 2.916267923,\ A_{0,1,0,1} = -0.2956420403,$$
$$A_{0,1,1,0} = 3.432158297,\ A_{0,1,1,1} = -3.654131904,\ A_{1,0,0,0} = -4.037115146,$$
$$A_{1,0,0,1} = 1.785750418,\ A_{1,0,1,0} = -1.942049920,\ A_{1,0,1,1} = 2.101651246,$$
$$A_{1,1,0,0} = 0.8912888158,\ A_{1,1,0,1} = 1.048958596,\ A_{1,1,1,0} = 1.048958596,$$
$$A_{1,1,1,1} = 4.043754763]$$

Presentamos la solución mejorada como una matriz

```
> subs(DatosNuevos,evalm(Q));
```

$$\begin{bmatrix} 3.812406126 & 1.180633615 & 1.180633616 & 1.389488746 \\ 2.916267923 & -0.2956420403 & 3.432158297 & -3.654131904 \\ -4.037115146 & 1.785750418 & -1.942049920 & 2.101651246 \\ 0.8912888158 & 1.048958596 & 1.048958596 & 4.043754763 \end{bmatrix}$$

Sustituimos la solución dada por la solución mejorada

```
> Datos := eval(DatosNuevos);
```

$$Datos := [A_{0,0,0,0} = 3.812406126,\ A_{0,0,0,1} = 1.180633615,\ A_{0,0,1,0} = 1.180633616,$$
$$A_{0,0,1,1} = 1.389488746,\ A_{0,1,0,0} = 2.916267923,\ A_{0,1,0,1} = -0.2956420403,$$
$$A_{0,1,1,0} = 3.432158297,\ A_{0,1,1,1} = -3.654131904,\ A_{1,0,0,0} = -4.037115146,$$
$$A_{1,0,0,1} = 1.785750418,\ A_{1,0,1,0} = -1.942049920,\ A_{1,0,1,1} = 2.101651246,$$
$$A_{1,1,0,0} = 0.8912888158,\ A_{1,1,0,1} = 1.048958596,\ A_{1,1,1,0} = 1.048958596,$$
$$A_{1,1,1,1} = 4.043754763]$$

>

Ahora podemos repetir el Cálculo de la dimensión del espacio tangente con la solución mejorada

>

11.5 *Algoritmo Genético usado para resolver YBE.*

El algoritmo genético usado durante el desarrollo de ésta tesis está descrito en [R 13] y fue desarrollado por R. P. Hernández y A. Kuri.

A éste algoritmo se le hicieron algunos ajustes finos para mejorar su desempeño al resolver YBE considerando las características propias de dicha ecuación.

El algoritmo es un Algoritmo Genético Ecléctico (EGA por sus siglas en inglés) y se llama así porque toma elementos de diversas estrategias para obtener un buen optimizador. Se centra la atención en la resolución de problemas de optimización restringida sobre dominios de variable múltiple.

Las principales características del algoritmo son:

- ✓ Parámetros autoajustables.
- ✓ Invocación selectiva de un buscador local aleatorio.
- ✓ Apareamiento determinístico.
- ✓ Cruzamiento anular.
- ✓ Calificación determinística para manejo de restricciones.
- ✓ Elitismo total.

Algunos parámetros propios del algoritmo en general son insertados en el genoma de cada individuo. El parámetro usado es el promedio de lo que se tiene en toda la población. De esta manera el parámetro "evoluciona" hacia un valor óptimo entre un límite mínimo y máximo. Algunos parámetros de este tipo son: probabilidad de cruzamiento, probabilidad de mutación, tamaño de la población, entre otros. Para esta tesis se fijaron la probabilidad de cruzamiento en 1 y la probabilidad de mutación entre 0.9 y 1. Esto último permite mantener mayor diversidad genética ya que la mutación se hace después de cada cruzamiento. Así los hijos serán parecidos a sus padres pero con una diferencia adicional evitándose en cierta medida la convergencia prematura.

El buscador local será invocado con mayor frecuencia cuando los mejores individuos provengan de una búsqueda local y con menor frecuencia cuando provengan de cruzamientos entre otros individuos. En el caso de esta tesis se forzó a que la invocación fuera muy frecuente ya que así mejoraba el desempeño del algoritmo. De hecho se hacen entre 10 y 20 invocaciones por cada ciclo de cruzamientos. A diferencia del algoritmo original, no se conservan como individuos a las mutaciones exitosas que se van haciendo sino que solamente se conserva al mejor individuo encontrado en cada invocación. El número de pasos en cada invocación del buscador local se fijó en un 10% de la longitud del cromosoma que en este caso es alrededor de 520 bits.

El apareamiento es determinístico siguiendo el esquema de "Vasconcelos", es decir, que el mejor individuo se cruza con el peor, el segundo mejor con el segundo peor, etc. Esto ayuda a conservar la diversidad genética.

El cruzamiento anular significa que el genoma de cada individuo es considerado como un anillo. Al hacer un cruzamiento con otro individuo, se "corta" el anillo en dos

lugares al azar iguales para ambos individuos y se intercambian los segmentos correspondientes entre ellos.

Al calificar a los individuos se hace de manera determinística (no aleatoria). Los individuos factibles son siempre mejores que los no factibles. Los individuos factibles se califican con la función objetivo y los no factibles mediante la trasgresión total.

El elitismo total se refiere a que solamente un número fijo de los primeros mejores individuos pasarán a la siguiente generación.

Capítulo 12 **Referencias.**

[R 1] **E. Artin,** *Theory of braids,* Ann. Math. 48, 101-126, 1947.

[R 2] **C. N. Yang,** *Some exact results for the many-body problem in one dimension with repulsive delta-function interaction*, Phys. Rev. Lett., 19, 1312-1315, 1967.

[R 3] **R. J. Baxter,** *Partition function for the eight-vertex lattice model,* Ann. Physics 70, 193-228, 1972.

[R 4] **R. J. Baxter,** *Exactly solved models in statistical mechanics,* Academic press, London, 1982.

[R 5] **L. D. Faddeev,** *Integrable models in (1+1)-dimensional quantum field theory,* Lectures at les Houches, 1982. Elsevier Sciences Publishes B. V., Amsterdam, New York 1984.

[R 6] **J. Hietarinta,** *Solving the constant quantum Yang-Baxter equation in 2 dimensions with massive use of factorizing Gröbner basis computations,* International conference on symbolic and algebraic computation, ACM Press, 350-357, New York, NY, USA, 1992.

[R 7] **C. Kassel,** *Quantum Groups,* Springer Verlag, New York, 1995.

[R 8] **Chaichan and A. Demichev,** *Introduction to quantum groups,* World Scientific, Singapore, New Yersey, London, Hong Kong, 1996.

[R 9] **J. C. Jantzen,** *Lectures on quantum groups.* American mathematical society, Graduate studies in mathematics, V. 6, 1996.

[R 10] **A. Klimyk, K. Schmüdgen,** *Quantum groups,* Springer Verlag, Berlin, Heidelberg, New York, 1997.

[R 11] **M. Rosso,** *Groupes quantiques et algèbres de battages quantiques,* C.R. Acad. Sci. Paris 320, 145-148, 1995.

[R 12] **M. Rosso,** *Quantum groups and quantum shuffles,* Invent. Math. 113, 399-416, 1998.

[R 13] **R. P. Hernández García, A.F. Kuri**. *Using the EGA, a Non-Traditional GA, and Deterministic Ranking for Optimization of Constrained Functions.* Memorias del Congreso Mexicano de Computación Evolutiva COMCEV'03, pp.13-25. 2003. S. Botello, A. Hernández y C. Coello (editores). ISBN 968-57-33-00-7.

[R 14] **Vladislav Khartchenko.** *Constants of coordinate differential calculi defined by Yang-Baxter operators*, Journal of Algebra, V267, w1(2003), 96-129.

[R 15] **Moss E. Sweedler.** *Hopf Algebra*, Benjamin 1969 Cap. XII: Shuffle Clásico.

[R 16] **Z. Oziewicz, E. Paal, J. Rozanski.** *Derivations in braided geometry* Acta Physica Polonica B 1995.