4
2y

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A C A T L A N

PAQUETE EN "MATHEMATICA" PARA CALCULOS CON ALGEBRA DE CLIFFORD

T E S I S

P A R A O B T E N E R E L T I T U L O D E :

L I C E N C I A D O E N

M A T E M A T I C A S A P L I C A D A S

Y C O M P U T A C I O N

P R E S E N T A :

OSCAR GABRIEL CABALLERO MARTINEZ

MEXICO, D. F.

1995

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

A mi mamá, por su eterno apoyo y comprensión,

y a mis hermanos

Agradecimientos

A los Doctores Luis Beltrán, por su orientación sobre la publicación del artículo; Favio Dávila, por proporcionarme sus notas que sirvieron mucho en la elaboración de la presente tesis; Humberto Terrones, por haber puesto interés en las clases de Mathematica y la beca que otorgó el CONACyT; David Romeu, por sus múltiples comentarios en las clases de Mathematica, que sirvieron en gran parte para el paquete; Alfredo Gómez, quien siguió muy de cerca la elaboración del paquete y se interesó mucho en que no existiera un paquete similar en el mercado.

Mi más grande admiración y sincero agradecimiento al Dr. José Luis Aragón, por sus consejos, forma de trato, amistad, compañerismo y espacio-tiempo cedido para la elaboración de esta tesis.

A todos ellos, **muchas gracias.**

Contenido

Introducción

Recientemente en un considerable número de publicaciones y libros se ha insistido en que el álgebra de Clifford es un poderoso lenguaje para la Física, ya que es la estructura matemática natural de objetos tales como: vectores, números complejos, cuaternos, matrices de Pauli, espinores de Dirac, etc. Se ha aplicado principalmente en dos campos: Mecánica Clásica y Mecánica Cuántica. Recientemente, se ha vislumbrado la posibilidad de aplicar este lenguaje a la Cristalografía geométrica en dimensiones arbitrarias, para la que el álgebra de vectores tradicional (de Gibbs) presenta muchas limitaciones (una de ellas es la imposibilidad de calcular el producto vectorial o cruz en más de 3 dimensiones).

Las operaciones con álgebra de Clifford en un espacio de dimensión, digamos 3, obliga a trabajar con 2^3 variables. Los cálculos, en principio, son sencillos de hacer, pero el problema es que el número de operaciones involucradas es mayor mientras más dimensiones sean consideradas.

El objetivo de este trabajo es realizar un paquete en el sistema *Mathematica* que realice las operaciones básicas del álgebra de Clifford en un espacio de dimensión arbitraria (finita) y que trabaje simbólicamente. Aparte del problema del alto número de operaciones involucradas en esta álgebra, la utilidad de un paquete es evidente desde el momento en que permite la manipulación de diversos objetos matemáticos (vectores, números complejos, cuaternos, etc.) con un sólo algoritmo computacional.

Mathematica es un sistema general para hacer matemática computacional. Se puede usar como una sofisticada calculadora, pero además de las operaciones numéricas puede realizar operaciones algebraicas y simbólicas. Contiene una gran cantidad de funciones predefinidas y con un lenguaje de programación propio. Puede, finalmente, generar gráficos de mucha complejidad. Se ha escogido este sistema gracias a que maneja, con más eficacia, el cálculo simbólico. Existen otros sistemas, por ejemplo el *Maple*, que también trabajan operaciones algebraicas, pero los cálculos simbólicos no son tan eficientes como el de *Mathematica*. El cálculo simbólico de *Mathematica* será de fundamental importancia para nosostros.

El presente trabajo se divide en cuatro capítulos y dos apéndices. En el primero, se da una introducción general al álgebra de Clifford de un espacio vectorial de dimensión finita. Se introduce el concepto de multivector y las operaciones básicas de esta álgebra. Se describe, como una aplicación práctica, cómo este lenguaje produce expresiones compactas y útiles cuando se formula la geometría analítica (en dimensiones arbitrarias) en estos términos. Se desarrolla también su aplicación a las rotaciones, proyecciones y reflexiones de vectores en cualquier dimensión.

En el segundo capítulo se describe brevemente el sistema *Mathematica*, sus principios, algunas de sus funciones predefinidas, su lenguaje de programación y la sintaxis de un paquete.

En el tercer capítulo, que es el núcleo de este trabajo, se describen los algoritmos que permitirán la programación de las dos operaciones que forman la base del álgebra de Clifford, a partir de las cuales pueden generarse todas las demás. Se incluye además una lista exaustiva de las funciones implícitas que contiene el paquete. Esta última parte es básicamente un manual de referencia para el usuario.

En el último capítulo se describen diversas aplicaciones del álgebra de Clifford. Esto además de dar una muestra del poder de esta herramienta matemática, consiste en un conjunto de ejemplos de la utilización del paquete.

En el apéndice **A** se enlista el paquete que hemos denominado **Clifford**. Finalmente, en el apéndice **B**, se incluye la copia de un artículo elaborado con los resultados de este trabajo en el que se dan detalles más precisos sobre los algoritmos que se desarrollaron.

1 El Algebra de Clifford.

1.1 Introducción y Orígenes del Algebra de Clifford.

En 1844, Herman Grassmann, desarrolló la idea de un *número dirigido.* Consta de un segmento de recta con una magnitud determinada, un sentido y una dirección. La magnitud de la recta es la cantidad de veces que se repite una línea unitaria a lo largo de ella, el sentido lo indica una flecha en uno de los extremos de la recta y la dirección está determinada por el ángulo que se forma entre la recta y una línea horizontal. El número dirigido tomó el nombre de *vector* [11].

Grassmann propone, a la vez, un nuevo concepto llamado *área dirigida*, que puede formarse colocando un vector b (de aquí en adelante todas las variables que representen un vector serán escritas en negritas) en el extremo de un vector a y formar un paralelogramo tirando líneas paralelas (véase Fig. 1).

Este paralelogramo tiene un sentido en contra de las manecillas del reloj, una dirección circular y un módulo igual al área que se forma. Si colocamos primero al vector b y en el extremo de la flecha al vector a, el sentido es al contrario que en el ejemplo anterior (véase Fig. 2).

A este nuevo objeto se le dió el nombre de *bivector* o *2-vector*, el cual tiene dirección (circular), sentido (a favor o en contra de las manecillas del reloj) y magnitud (área).

El área del paralelogramo, o magnitud del bivector está dada por:

$$|\mathbf{a}|\,|\mathbf{b}|\sin\theta,$$

donde θ es al ángulo formado por los dos vectores.

Un bivector se denota como $\mathbf{a} \wedge \mathbf{b}$. A la operación $\wedge$ se le conoce como *producto exterior.* Obsérvese que $-\mathbf{a} \wedge \mathbf{b}$ es igual a $\mathbf{b} \wedge \mathbf{a}$ lo que indica que el producto exterior no conmuta.

El producto exterior de tres vectores produce un *trivector* [21, 2] (ver Figura 3), el de n vectores nos dará un *n-vector.*

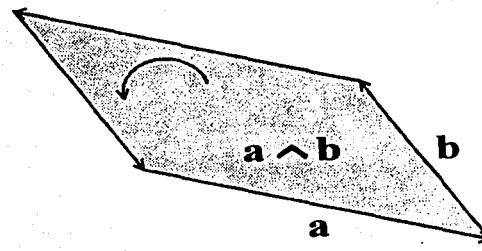

Figura 1: Representación geométrica de un bivector como un área dirigida

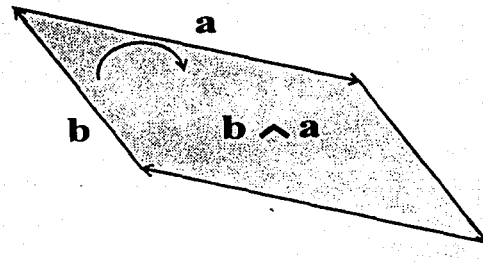

Figura 2: Representación geométrica del bivector en sentido inverso al anterior

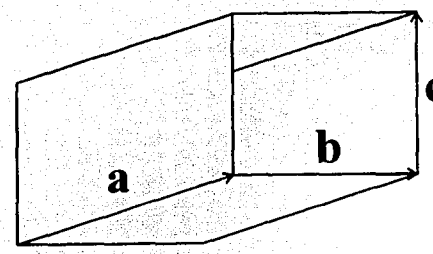

Figura 3: Representación geométrica de un trivector

Casi paralelamente, Clifford combina estas dos ideas y define un nuevo producto de vectores que denomina *Producto Geométrico* ab y que se define como:

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}.$$

donde $\mathbf{a} \cdot \mathbf{b}$ es el *producto escalar* usual que tiene el valor $|\mathbf{a}|\,|\mathbf{b}| \cos\theta$.

Al resultado anterior se le conoce con el nombre de *Multivector*. Estos objetos son el estudio principal del *Álgebra de Clifford*. Las principales operaciones entre ellos como: sumas, multiplicaciones, productos escalar y exterior, son el objetivo principal del presente capítulo.

Veamos un poco más en detalle cómo se realiza un producto geométrico. Para ello, a los vectores a y b en R^2, los representaremos en términos de una base canónica $\mathbf{e}_1 = (1,0)$ y $\mathbf{e}_2 = (0,1)$:

$$\mathbf{a} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2,$$
$$\mathbf{b} = b_1\mathbf{e}_1 + b_2\mathbf{e}_2,$$

entonces

$$\mathbf{ab} = a_1b_1\mathbf{e}_1\mathbf{e}_1 + a_1b_2\mathbf{e}_1\mathbf{e}_2 + a_2b_1\mathbf{e}_2\mathbf{e}_1 + a_2b_2\mathbf{e}_2\mathbf{e}_2. \tag{1}$$

Puesto que $\mathbf{e}_1 \cdot \mathbf{e}_2 = 0$ y $\mathbf{e}_2 \wedge \mathbf{e}_1 = -\mathbf{e}_1 \wedge \mathbf{e}_2$, entonces

$$\begin{aligned} \mathbf{e}_1\mathbf{e}_2 &= \mathbf{e}_1 \cdot \mathbf{e}_2 + \mathbf{e}_1 \wedge \mathbf{e}_2 = 0 + \mathbf{e}_1 \wedge \mathbf{e}_2 = \mathbf{e}_1 \wedge \mathbf{e}_2, \\ \mathbf{e}_2\mathbf{e}_1 &= \mathbf{e}_2 \wedge \mathbf{e}_1 = -\mathbf{e}_1 \wedge \mathbf{e}_2 = -\mathbf{e}_1\mathbf{e}_2. \end{aligned} \tag{2}$$

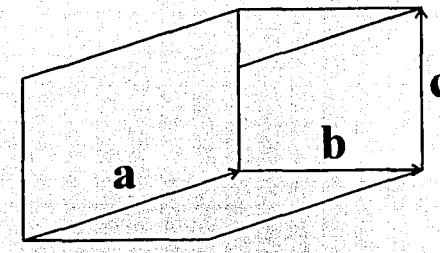

Figura 3: Representación geométrica de un trivector

Casi paralelamente, Clifford combina estas dos ideas y define un nuevo producto de vectores que denomina *Producto Geométrico* **ab** y que se define como:

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}.$$

donde $\mathbf{a} \cdot \mathbf{b}$ es el *producto escalar* usual que tiene el valor $|\mathbf{a}|\,|\mathbf{b}| \cos\theta$.

Al resultado anterior se le conoce con el nombre de *Multivector*. Estos objetos son el estudio principal del *Álgebra de Clifford*. Las principales operaciones entre ellos como: sumas, multiplicaciones, productos escalar y exterior, son el objetivo principal del presente capítulo.

Veamos un poco más en detalle cómo se realiza un producto geométrico. Para ello, a los vectores **a** y **b** en R^2, los representaremos en términos de una base canónica $\mathbf{e}_1 = (1,0)$ y $\mathbf{e}_2 = (0,1)$:

$$\mathbf{a} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2,$$
$$\mathbf{b} = b_1\mathbf{e}_1 + b_2\mathbf{e}_2,$$

entonces

$$\mathbf{ab} = a_1b_1\mathbf{e}_1\mathbf{e}_1 + a_1b_2\mathbf{e}_1\mathbf{e}_2 + a_2b_1\mathbf{e}_2\mathbf{e}_1 + a_2b_2\mathbf{e}_2\mathbf{e}_2. \tag{1}$$

Puesto que $\mathbf{e}_1 \cdot \mathbf{e}_2 = 0$ y $\mathbf{e}_2 \wedge \mathbf{e}_1 = -\mathbf{e}_1 \wedge \mathbf{e}_2$, entonces

$$\begin{aligned} \mathbf{e}_1\mathbf{e}_2 &= \mathbf{e}_1 \cdot \mathbf{e}_2 + \mathbf{e}_1 \wedge \mathbf{e}_2 = 0 + \mathbf{e}_1 \wedge \mathbf{e}_2 = \mathbf{e}_1 \wedge \mathbf{e}_2, \\ \mathbf{e}_2\mathbf{e}_1 &= \mathbf{e}_2 \wedge \mathbf{e}_1 = -\mathbf{e}_1 \wedge \mathbf{e}_2 = -\mathbf{e}_1\mathbf{e}_2. \end{aligned} \tag{2}$$

Por otro lado $e_1 \cdot e_1 = 1$ y $e_1 \wedge e_1 = 0$ (pues no generan áreas), por lo que

$$(e_1)^2 = e_1 e_1 = e_1 \cdot e_1 + e_1 \wedge e_1 = 1 + 0 = 1, \tag{3}$$

y lo mismo para e_2. Sustituyendo (2) y (3) en la ecuación (1)

$$\begin{aligned} \mathbf{ab} &= a_1 b_1 e_1 e_1 + a_1 b_2 e_1 e_2 + a_2 b_1 e_2 e_1 + a_2 b_2 e_2 e_2 \\ &= (a_1 b_1)1 + a_1 b_2 (e_1 \wedge e_2) + a_2 b_1 (-e_1 \wedge e_2) + (a_2 b_2)1 \\ &= a_1 b_1 + a_2 b_2 + a_1 b_2 e_1 e_2 - a_2 b_1 e_1 e_2 \\ &= a_1 b_1 + a_2 b_2 + (a_1 b_2 - a_2 b_1) e_1 e_2 \end{aligned} \tag{4}$$

En la ecuación (4) se puede observar que $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2$ y que $\mathbf{a} \wedge \mathbf{b} = (a_1 b_2 - a_2 b_1) e_1 e_2$. Nótese que, en general, dado un espacio vectorial (R^2) y un producto como en (4), cualquier elemento que resulte puede escribirse como una combinación lineal de

$$(1, e_1, e_2, e_1 e_2),$$

que es la base de un nuevo espacio vectorial que se denota por $C(R^2)$ y se denomina álgebra de Clifford de R^2.

Si se desea construir $C(R^3)$, los vectores canónicos tendrán dimensión de 3 y se forma de todas las posibles sumas y productos de los elementos de la base

$$(1, e_1, e_2, e_3, e_1 e_2, e_1 e_3, e_2 e_3, e_1 e_2 e_3).$$

Esta base contiene ocho elementos: un *escalar*, tres *vectores*, tres *bivectores* y un *trivector*.

1.2 El Algebra de Clifford de R^n.

Partiendo de los resultados anteriores, se presentará una definición más general del álgebra de Clifford de R^n [5].

Sea R^n el espacio vectorial Euclidiano sobre los números reales, con una base $\{e_1, e_2, ..., e_n\}$ tal que

$$(e_i \mid e_j) = \delta_{ij}.$$

Introduciremos en este espacio un producto $\mathbf{uv}$ de vectores que pertenecen a R^n, los cuales cumplen con las propiedades asociativa y distributiva con respecto a la adición y que satisface la condición

$$\mathbf{uv} + \mathbf{vu} = 2(\mathbf{u} \mid \mathbf{v})\mathbf{1},$$

donde $(\mathbf{u} \mid \mathbf{v})$ es el producto escalar Euclidiano en R^n, y $\mathbf{1}$ es la identidad.

El resultado de todas las posibles sumas y productos de los vectores $\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n\}$ es llamado Álgebra de *Clifford* de R^n que se denota por $C(R^n)$. El producto $\mathbf{uv}$ es llamado *producto geométrico* de $\mathbf{u}$ y $\mathbf{v}$. Note que en particular

$$\mathbf{e}_i\mathbf{e}_j = -\mathbf{e}_j\mathbf{e}_i, \quad i > j$$
$$(\mathbf{e}_i)^2 = 1,$$
$$\mathbf{v}^2 = (\mathbf{v} \mid \mathbf{v}).$$

El Álgebra de Clifford $C(R^n)$ es por sí misma un espacio vectorial de dimensión $\sum_{p=0}^{n} \binom{n}{p} = 2^n$ y con base

$$(1, \mathbf{e}_1, \mathbf{e}_2, .., \mathbf{e}_n, \mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, .., \mathbf{e}_1\mathbf{e}_n, ..., \mathbf{e}_1\mathbf{e}_2 \cdots \mathbf{e}_n).$$

Un elemento A que pertenece a $C(R^n)$ se escribe de la forma

$$A = a_0 + a_{11}\mathbf{e}_1 + \cdots + a_{1i}\mathbf{e}_{1n} + a_{21}\mathbf{e}_1\mathbf{e}_2 + \cdots + a_{2i}\mathbf{e}_1\mathbf{e}_n + \cdots + a_{di}\mathbf{e}_1\mathbf{e}_2 \cdots \mathbf{e}_n,$$

donde $d = 2^n - 1$ e $i = \binom{n}{p}$ para todo número real a_{pi}. Consecuentemente el espacio vectorial $C(R^n)$ puede descomponerse en $n + 1$ subespacios:

$$C(R^n) = C_0 \oplus C_1 \oplus \cdots \oplus C_n \tag{5}$$

cada uno de dimensión $\binom{n}{p}$.

Los elementos A del álgebra de Clifford son llamados *multivectores*, los elementos de C_p son llamados p-vectores. En particular, los 0-vectores son números reales y C_0 tiene dimensión 1. C_1 tiene una base $\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n\}$, son 1-vectores o simplemente vectores y con $\dim(C_1) = n$. C_2 tiene una base $\{\mathbf{e}_1\mathbf{e}_2, ..., \mathbf{e}_1\mathbf{e}_n, \mathbf{e}_2\mathbf{e}_3, ..., \mathbf{e}_2\mathbf{e}_n, ..., \mathbf{e}_{n-1}\mathbf{e}_n\}$, son bivectores y $\dim(C_2) = \binom{n}{2}$. Finalmente, C_n tiene una base $\{\mathbf{e}_1\mathbf{e}_2 \cdots \mathbf{e}_n\}$, con

$\dim(\mathcal{C}_n) = 1$, se denominan n-vectores y también son llamados *pseudoescalares* [10] y suelen ser denotados por I_n. Obsérvese que un n-vector arbitrario A_n puede ser escrito como:

$$A_n = \lambda \mathbf{e}_1\mathbf{e}_2 \cdots \mathbf{e}_n = \lambda I_n,$$

donde λ es un número real.

En lo sucesivo, un multivector o un p-vector se denotará solamente con mayúsculas, por ejemplo A, sin nigún tipo de adorno[1] . Cuando utilicemos I_n nos estaremos refiriendo al pseudoescalar de $C(R^n)$ o n-vector.

Tomando en cuenta la descomposición de $C(R^n)$ en sumas de subespacios $\mathcal{C}_p$, $0 \leq p \leq n$, dado en (5), podemos extraer la parte de grado p de un multivector A por medio de una proyección $\langle\rangle\colon C(R^n) \to C(R^n)$, definiendola como sigue. Si

$$A = A_0 + A_1 + \cdots + A_n, \tag{6}$$

con $A_p \in \mathcal{C}_p$, entonces $\langle A\rangle_p = A_p$, y decimos que A_p es la parte p-vectorial de A, y que A_p es de grado p o el *grado* de A_p es p. En particular, (6) es equivalente a

$$A = \langle A\rangle_0 + \langle A\rangle_1 + \cdots + \langle A\rangle_n = \sum_{p=0}^{n} \langle A\rangle_p. \tag{7}$$

Si un multivector tiene la forma $A = \langle A\rangle_r$ para un entero positivo r, se dice que A es *homogéneo* de grado r.

La *magnitud* o *módulo* de A está definida por la ecuación

$$|A| = \left\langle A^\dagger A\right\rangle_0^{1/2}. \tag{8}$$

Donde $A^\dagger$ se denomina el *reverso* de A y se define como sigue. Si $A_p \in \mathcal{C}_p$, entonces

$$A_p = \mathbf{v}_1\mathbf{v}_2 \cdots \mathbf{v}_p,$$

se define al *reverso* de A_p como

$$A_p^\dagger = \mathbf{v}_p \cdots \mathbf{v}_2\mathbf{v}_1.$$

[1] Observe que los 1-vectores son vectores comunes y serán denotados como se especificó en un principio.

El reverso es distributivo [11, 5], por lo que

$$A^\dagger = A_0^\dagger + A_1^\dagger + \cdots + A_n^\dagger.$$

De (7) tenemos

$$|A|^2 = \langle A^\dagger A \rangle_0 = |\langle A \rangle_0|^2 + |\langle A \rangle_1|^2 + \cdots + |\langle A \rangle_n|^2 = \sum_{p=0}^{n} |\langle A \rangle_p|^2 .$$

Ahora, si el inverso de A existe [18], éste será denotado por A^{-1} o $1/A$. Y se define por la ecuación $AA^{-1} = 1$. En particular, para un $A_p = v_1 v_2 \cdots v_p$, su inverso estará dado por

$$A_p^{-1} = \frac{A_p^\dagger}{|A_p|^2} = \frac{A_p^\dagger}{v_1^2 v_2^2 \cdots v_p^2}. \tag{9}$$

Para que (9) exista, debe de cumplir con la siguiente condición:

$$AA^\dagger \in R$$

que es conocida con el nombre de *condición de Plüker*.

El producto exterior e interior definidos en la sección anterior para vectores, puede generalizarse tambien a multivectores. Estas operaciones son importantes porque permiten incrementar o disminuir el grado de un multivector. Esto es, establecen la relación entre todos los subespacios C_p dados en (5).

Sean $A_p \in C_p$ y $B_q \in C_q$ dos r-vectores de grado p y q respectivamente, se define su producto interno (escalar) como:

$$A_p \cdot B_q = \begin{cases} \langle A_p B_q \rangle_{|p-q|} & \text{si } p, q > 0 \\ 0 & \text{si } p = 0 \text{ o } q = 0, \end{cases} \tag{10}$$

y su producto exterior como

$$A_p \wedge B_q = \langle A_p B_q \rangle_{p+q}. \tag{11}$$

El producto $A_p \cdot B_q$ es el término de menor grado en $A_p B_q$ y $A_p \wedge B_q$ es el término de mayor grado. Con las definiciones anteriores, aplicándolas en multivectores, tenemos la siguiente definición:

Sea $A, B \in C(R^n)$ entonces

$$A \cdot B = \sum_{p,q>0} \langle A \rangle_p \cdot \langle B \rangle_q ,$$
$$A \wedge B = \sum_{p,q} \langle A \rangle_p \wedge \langle B \rangle_q .$$

De Acuerdo a (10) y (11), el producto de A_p con un q-vector, si es interior reduce el grado de A_p en q unidades, si es exterior aumenta el grado en q unidades. El producto interno y externo son los que reducen y aumentan el grado de los multivectores en el álgebra de Clifford y constituyen las operaciones básicas que relacionan a los diferentes subespacios de $C(R^n)$ definidos en la expresión (5).

Si igualamos a $\langle A \rangle_1 = \mathbf{a}$ y a $\langle B \rangle_1 = \mathbf{b}$ que se representaron en la ecuación (1), lógicamente los multivectores son de grado 1, ayudándonos por el resultado en la ecuación (4) tenenos que

$$A \cdot B = \sum_{p,q>0} \langle A \rangle_1 \cdot \langle B \rangle_1 = \langle \mathbf{ab} \rangle_0 = a_1 b_1 + a_2 b_2 = \mathbf{a} \cdot \mathbf{b}.$$

Y

$$A \wedge B = \sum_{p,q} \langle A \rangle_1 \wedge \langle B \rangle_1 = \langle \mathbf{ab} \rangle_2 = (a_1 b_2 - a_2 b_1)\mathbf{e}_1 \mathbf{e}_2 = \mathbf{a} \wedge \mathbf{b}.$$

Por último, el producto geométrico es:

$$AB = \langle AB \rangle_0 + \langle AB \rangle_2 .$$

Si multiplicamos un vector $\mathbf{v}$ por un p-vector A_p, podemos escribir que

$$\mathbf{v} A_p = \mathbf{v} \cdot A_p + \mathbf{v} \wedge A_p , \tag{12}$$

relación que no es válida para dos r-vectores en general [11].

Si B_p es un p-vector que puede escribirse como el producto exterior de p vectores:

$$B_p = \mathbf{v}_1 \wedge \mathbf{v}_2 \wedge \cdots \wedge \mathbf{v}_p$$

entnces B_p se denomina como *p-blade* o simplemente *blade*[2] .

[2] No encontramos una traducción adecuada del término "*blade*" por lo que preferimos utilizarlo en inglés.

1.3 Proyecciones, Reflexiones y Rotaciones en R^3.

Regresemos nuevamente al caso de vectores en R^3 que se discutió en la sección 1.1. En el producto geométrico de dos vectores **a** y **b** se encuentra toda la información a cerca de la dirección de los dos vectores. Si descomponemos a los dos vectores en su forma simétrica (parte escalar) y antisimétrica (parte bivector) de acuerdo con su ecuación fundamental

$$\begin{aligned} \mathbf{ab} &= \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\wedge\mathbf{b} \\ \mathbf{ba} &= \mathbf{b}\cdot\mathbf{a} + \mathbf{b}\wedge\mathbf{a} = \mathbf{a}\cdot\mathbf{b} - \mathbf{a}\wedge\mathbf{b}. \end{aligned} \tag{13}$$

Observe, si hacemos la suma de los dos productos de (13) obtenemos

$$\begin{aligned} \mathbf{ab}+\mathbf{ba} &= \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\wedge\mathbf{b} + \mathbf{b}\cdot\mathbf{a} + \mathbf{b} \\ &= \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\wedge\mathbf{b} + \mathbf{a}\cdot\mathbf{b} - \mathbf{a}\wedge\mathbf{b} \\ &= 2(\mathbf{a}\cdot\mathbf{b}) \\ \mathbf{a}\cdot\mathbf{b} &= (\mathbf{ab}+\mathbf{ba})/2, \end{aligned} \tag{14}$$

y restando, resulta

$$\begin{aligned} \mathbf{ab}-\mathbf{ba} &= \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\wedge\mathbf{b} - \mathbf{b}\cdot\mathbf{a} - \mathbf{b}\wedge\mathbf{a} \\ &= \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\wedge\mathbf{b} - \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\wedge\mathbf{b} \\ &= 2(\mathbf{a}\wedge\mathbf{b}) \\ \mathbf{a}\wedge\mathbf{b} &= (\mathbf{ab}-\mathbf{ba})/2. \end{aligned} \tag{15}$$

Los vectores son colineales o paralelos, si $\mathbf{a}\wedge\mathbf{b} = 0$ y sustituyendo en la ecuación (15) obtenemos que $\mathbf{ab}-\mathbf{ba} = 0$, entonces $\mathbf{ab} = \mathbf{ba}$. Si son ortogonales, sabemos que $\mathbf{a}\cdot\mathbf{b} = 0$ y sustituyendo en (14) obteniendo $\mathbf{ab}+\mathbf{ba} = 0$, entonces $\mathbf{ab} = -\mathbf{ba}$.

1.3.1 Proyecciones.

Dados dos vectores **a**, **b**, a uno lo podemos descomponer en un vector paralelo al otro y ese mismo en uno ortogonal al otro. Por ejemplo, el vector **a** se puede descomponer en [11]

$$\mathbf{a} = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp}, \tag{16}$$

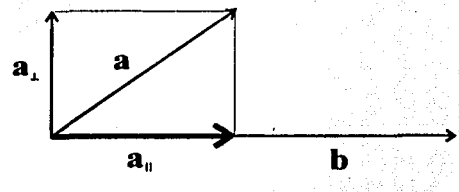

Figura 4: Proyección y rechazo de un vector sobre otro

donde $\mathbf{a}_{\|}$ es paralelo a $\mathbf{b}$ y $\mathbf{a}_{\perp}$ es ortogonal a $\mathbf{b}$, entonces

$$\begin{aligned}
\mathbf{a}_{\|}\mathbf{b} &= \mathbf{a}_{\|}\cdot\mathbf{b}+\mathbf{a}_{\|}\wedge\mathbf{b}=\mathbf{a}_{\|}\cdot\mathbf{b},\\
\mathbf{a}_{\perp}\mathbf{b} &= \mathbf{a}_{\perp}\cdot\mathbf{b}+\mathbf{a}_{\perp}\wedge\mathbf{b}=\mathbf{a}_{\perp}\wedge\mathbf{b},\\
\mathbf{a}\cdot\mathbf{b} &= (\mathbf{a}_{\|}+\mathbf{a}_{\perp})\cdot\mathbf{b}=\mathbf{a}_{\|}\cdot\mathbf{b}+\mathbf{a}_{\perp}\cdot\mathbf{b}=\mathbf{a}_{\|}\cdot\mathbf{b}=\mathbf{a}_{\|}\mathbf{b}=\mathbf{b}\mathbf{a}_{\|},\\
\mathbf{a}\wedge\mathbf{b} &= (\mathbf{a}_{\|}+\mathbf{a}_{\perp})\wedge\mathbf{b}=\mathbf{a}_{\|}\wedge\mathbf{b}+\mathbf{a}_{\perp}\wedge\mathbf{b}=\mathbf{a}_{\perp}\wedge\mathbf{b}=\mathbf{a}_{\perp}\mathbf{b}=-\mathbf{b}\mathbf{a}_{\perp},
\end{aligned} \tag{17}$$

como se ilustra en la Figura 4. Despejando los dos últimos resultados en (17) tenemos

$$\begin{aligned}
\mathbf{a}_{\|} &= \mathbf{a}\cdot\mathbf{b}\mathbf{b}^{-1}\\
\mathbf{a}_{\perp} &= \mathbf{a}\wedge\mathbf{b}\mathbf{b}^{-1}
\end{aligned} \tag{18}$$

En (18) puede haber una confución en el producto que debe realizarse primero. Para evitar esto se sigue la siguiente jerarquía: producto exterior, producto escalar y producto geométrico. Es decir, el producto geométrico es el último en evaluarse.

La ecuación (18) es válida para cualquier k-blade B_k, ya que de (12) se tiene:

$$\begin{aligned}
\mathbf{a}_{\|} &= \mathbf{a}\cdot B_k B_k^{-1}\\
\mathbf{a}_{\perp} &= \mathbf{a}\wedge B_k B_k^{-1}
\end{aligned} \tag{19}$$

En general, para un plano definido por B_2, $\mathbf{a}_{\|}$ y $\mathbf{a}_{\perp}$ (véase Fig. 5) son vectores paralelos y ortogonales al plano, que reciben los nombres de *proyección* y *rechazo* respectivamente.

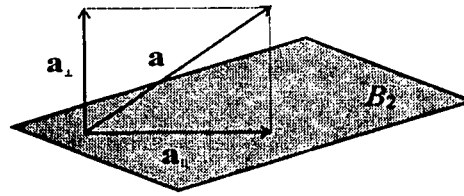

Figura 5: Proyección y rechazo de un vector sobre un plano

1.3.2 Reflexiones.

Una reflexión de un vector (x, y, z) sobre el plano xy o bien a lo largo del eje z, está dada por la transformación [9, 20]:

$$T\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \\ y \\ -z \end{pmatrix}. \tag{20}$$

Véase Figura 6.

En el álgebra de Clifford, se puede representar esta transformación por medio de productos geométricos. La *reflexión* de un vector $\mathbf{x}$ a lo largo de un vector $\mathbf{u}$ esta dada por [2, 11, 12]

$$\mathrm{U}(\mathbf{x}) = -\mathbf{u}^{-1}\mathbf{x}\mathbf{u} \tag{21}$$

Para el ejemplo de la transformación en (20), sabemos que la solución es que la componente en z es negativa, para resolverlo se tiene que

$$\begin{aligned} \mathbf{x} &= x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3, \\ \mathbf{u} &= z\mathbf{e}_3 \qquad\qquad \mathrm{y} \\ \mathbf{u}^{-1} &= \frac{z\mathbf{e}_3}{z^2} = \frac{\mathbf{e}_3}{z}. \end{aligned} \tag{22}$$

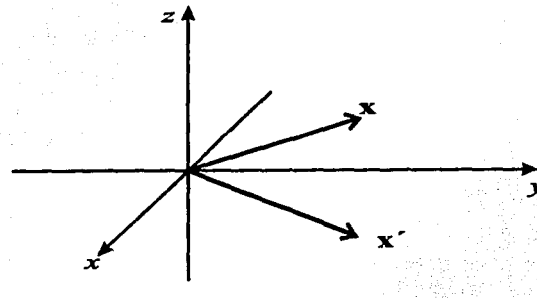

Figura 6: Transformación Lineal de un vector en el espacio.

Siguiendo los pasos de la función (21) y utilizando (22) tenemos

$$\begin{aligned} -\mathbf{u}^{-1}\mathbf{x} &= \frac{xe_1e_3 + ye_2e_3 - z}{z} \\ -\mathbf{u}^{-1}\mathbf{x}\mathbf{u} &= \frac{xze_1 + yze_2 - z^2e_3}{z} = xe_1 + ye_2 - ze_3, \end{aligned} \tag{23}$$

que es el resultado esperado.

En el caso de que el vector u sea unitario, la ecuación (21) se convierte en

$$\mathbf{U}(\mathbf{x}) = -\mathbf{u}\mathbf{x}\mathbf{u}. \tag{24}$$

Si descomponemos a $\mathbf{x}$ en $\mathbf{x} = \mathbf{x}_{\|} + \mathbf{x}_{\perp}$ y usando (17) vemos que

$$\begin{aligned} \mathbf{u}\mathbf{x}\mathbf{u} &= \mathbf{u}(\mathbf{x}_{\|} + \mathbf{x}_{\perp})\mathbf{u} \\ &= \mathbf{u}(\mathbf{x}_{\|} + \mathbf{x}_{\perp})\mathbf{u} \\ &= \mathbf{u}\mathbf{x}_{\|}\mathbf{u} + \mathbf{u}\mathbf{x}_{\perp}\mathbf{u} \\ &= \mathbf{u}\mathbf{u}\mathbf{x}_{\|} - \mathbf{u}\mathbf{u}\mathbf{x}_{\perp} \\ &= \mathbf{x}_{\|} - \mathbf{x}_{\perp}, \end{aligned}$$

donde se utilizó el hecho de que $\mathbf{u}\mathbf{u} = \mathbf{u}^2 = 1$. Con estos resultados podemos ver que

$$\mathbf{U}(\mathbf{x}) = -\mathbf{u}\mathbf{x}\mathbf{u} = \mathbf{x}_{\perp} - \mathbf{x}_{\|}. \tag{25}$$

Véase la Figura 7. Supongamos que ahora se nos proporciona el plano de reflexión por medio de un 2-blade B_2. Para poder encontrar un vector normal a dicho plano,

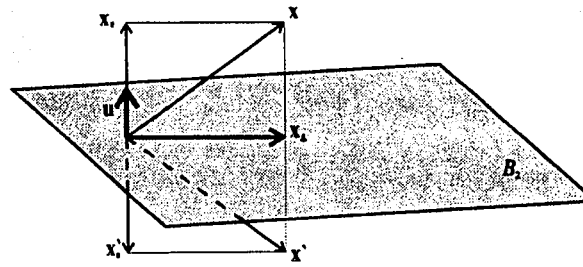

Figura 7: Reflexión de un vector sobre un plano.

sólo es cuestión de multiplicarlo por el reverso del pseudoescalar, de R^3 en este caso,

$$\mathbf{u} = B_2 I_3^{\dagger}$$

y se dice que **u** es el *dual* de B_2.

La operación de dualidad puede definirse de una forma general

$$\tilde{B}_{n-k} = B_k I_n^{\dagger}, \tag{26}$$

donde $\tilde{B}_{n-k}$ es el *dual* del k-blade, en general, el dual de un k-blade en R^n es un $(n-k)$-blade. Para nuestro ejemplo, el plano por el cual se refleja **x** es el xy, el cual representaremos como el producto de $\mathbf{e}_1\mathbf{e}_2$ para que el vector **u** sea unitario. Entonces

$$\mathbf{u} = \tilde{B}_{3-2} = B_2 I_3^{\dagger} = \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}_2\mathbf{e}_1 = \mathbf{e}_3,$$

sutituyéndolo en (25) y haciendo operaciones tenemos que

$$\begin{aligned} \mathrm{U}(\mathbf{x}) &= -\mathbf{u}\mathbf{x}\mathbf{u} \\ &= -\mathbf{e}_3(x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3)\mathbf{e}_3 \\ &= (x\mathbf{e}_1\mathbf{e}_3 + y\mathbf{e}_2\mathbf{e}_3 - z)\mathbf{e}_3 \\ &= x\mathbf{e}_1 + y\mathbf{e}_2 - z\mathbf{e}_3 \end{aligned}$$

Por último, se nos pueden proporcionar los vectores que forman el plano por el cual se refleja el vector **x**. Para ello sólo les aplicamos el producto exterior y tendremos como resultado a B_2 y realizamos los pasos anteriores.

1.3.3 Rotaciones.

Primero, reescribiremos una forma muy conveniente del producto geométrico de 2 vectores. Se tenía que

$$\begin{aligned}\mathbf{ab} &= a_1b_1 + a_2b_2 + (a_1b_2 - a_2b_1)\mathbf{e}_1\mathbf{e}_2 \\ &= a_1b_1 + a_2b_2 + (a_1b_2 - a_2b_1)I_2,\end{aligned}$$

donde $I_2 = \mathbf{e}_1\mathbf{e}_2$, que lo podemos representar de la siguiente forma

$$\begin{aligned}\mathbf{ab} &= |\mathbf{a}||\mathbf{b}|\cos\theta + I_2|\mathbf{a}||\mathbf{b}|\sin\theta \\ &= |\mathbf{a}||\mathbf{b}|(\cos\theta + I_2\sin\theta) = |\mathbf{a}||\mathbf{b}|\,e^{I_2\theta}.\end{aligned} \tag{27}$$

Ahora, si al vector resultante en (24) le aplicamos otra reflexión a lo largo del vector unitario $\mathbf{v}$, tendríamos que

$$\mathrm{V}(\mathrm{U}(\mathbf{x})) = -\mathbf{v}(-\mathbf{uxu})\mathbf{v} = \mathbf{vuxuv} = \mathrm{VU}(\mathbf{x}). \tag{28}$$

Si decimos que $\mathrm{R} = \mathrm{V}(\mathrm{U})$ entonces $\mathrm{R} = \mathbf{uv}$ por lo tanto

$$\mathrm{R}(\mathbf{x}) = \mathbf{vuxuv} = \mathrm{R}^\dagger\mathbf{x}\mathrm{R}, \tag{29}$$

note que $\mathrm{R}^\dagger\mathrm{R} = 1$ ya que $|\mathrm{R}| = 1$. Veamos cuál es el efecto que tiene R sobre el vector $\mathbf{x}$. Los vectores $\mathbf{u}$ y $\mathbf{v}$ forman un plano, el cual llamaremos A, entonces al descomponer $\mathbf{x}$ tenemos, por (19), que

$$\begin{aligned}\mathbf{x}_\parallel &= \mathbf{x}\cdot AA^{-1} \\ \mathbf{x}_\perp &= \mathbf{x}\wedge AA^{-1}.\end{aligned}$$

Si sumamos $\mathbf{x}_\parallel A$ con su conmutado tenemos que

$$\begin{aligned}\mathbf{x}_\parallel A + A\mathbf{x}_\parallel &= \mathbf{x}\cdot A + A\cdot\mathbf{x} \\ &= \mathbf{x}\cdot(\mathbf{u}\wedge\mathbf{v}) + (\mathbf{u}\wedge\mathbf{v})\cdot\mathbf{x} \\ &= [\mathbf{x}\cdot(\mathbf{uv}-\mathbf{vu}) + (\mathbf{uv}-\mathbf{vu})\cdot\mathbf{x}]/2 \\ &= [\mathbf{x}\cdot\mathbf{uv} - \mathbf{x}\cdot\mathbf{vu} + \mathbf{uv}\cdot\mathbf{x} - \mathbf{vu}\cdot\mathbf{x}]/2 \\ &= [\mathbf{x}\cdot\mathbf{uv} - \mathbf{x}\cdot\mathbf{vu} - \mathbf{x}\cdot\mathbf{uv} + \mathbf{x}\cdot\mathbf{vu}]/2 = 0 \\ \mathbf{x}_\parallel A &= -A\mathbf{x}_\parallel.\end{aligned} \tag{30}$$

Ahora, si se resta de $\mathbf{x}_\perp A$ con su conmutado tenemos

$$\begin{aligned} \mathbf{x}_\perp A - A\mathbf{x}_\perp &= \mathbf{x}\wedge A - A\wedge\mathbf{x} \\ &= \mathbf{x}\wedge(\mathbf{u}\wedge\mathbf{v}) - (\mathbf{u}\wedge\mathbf{v})\wedge\mathbf{x} \\ &= \mathbf{x}\wedge\mathbf{u}\wedge\mathbf{v} + \mathbf{u}\wedge\mathbf{x}\wedge\mathbf{v} \\ &= \mathbf{x}\wedge\mathbf{u}\wedge\mathbf{v} - \mathbf{x}\wedge\mathbf{u}\wedge\mathbf{v} = 0 \\ \mathbf{x}_\perp A &= A\mathbf{x}_\perp. \end{aligned} \tag{31}$$

Puesto que $R = uv = u\cdot v + u\wedge v$ y $R^\dagger = vu = u\cdot v - u\wedge v$. Utilizando la ecuación (27) y que $\mathbf{u},\mathbf{v}\in C(R^2)$, las ecuaciones que acabamos de poner quedan como

$$\begin{aligned} R = \mathbf{uv} &= \cos\theta + I_2\sin\theta = e^{I_2\theta} \\ R^\dagger = \mathbf{vu} &= \cos\theta - I_2\sin\theta = e^{-I_2\theta}. \end{aligned} \tag{32}$$

Con los resultados obtenidos en (30) y (31), la ecuación (29) puede escribirse como

$$\begin{aligned} R(\mathbf{x}) &= R^\dagger\mathbf{x}R \\ &= (\cos\theta - I_2\sin\theta)(\mathbf{x}_\parallel + \mathbf{x}_\perp)(\cos\theta + I_2\sin\theta) \\ &= (\cos\theta - I_2\sin\theta)\mathbf{x}_\parallel(\cos\theta + I_2\sin\theta) + (\cos\theta - I_2\sin\theta)\mathbf{x}_\perp(\cos\theta + I_2\sin\theta) \\ &= (\mathbf{x}_\parallel\cos\theta - I_2\mathbf{x}_\parallel\sin\theta)(\cos\theta + I_2\sin\theta) + (\mathbf{x}_\perp\cos\theta - I_2\mathbf{x}_\perp\sin\theta)(\cos\theta + I_2\sin\theta) \\ &= (\mathbf{x}_\parallel\cos\theta + \mathbf{x}_\parallel I_2\sin\theta)(\cos\theta + I_2\sin\theta) + (\mathbf{x}_\perp\cos\theta - \mathbf{x}_\perp I_2\sin\theta)(\cos\theta + I_2\sin\theta) \\ &= \mathbf{x}_\parallel(\cos\theta + I_2\sin\theta)(\cos\theta + I_2\sin\theta) + \mathbf{x}_\perp(\cos\theta - I_2\sin\theta)(\cos\theta + I_2\sin\theta) \\ &= \mathbf{x}_\perp + \mathbf{x}_\parallel(\cos 2\theta + I_2\sin 2\theta). \end{aligned}$$

Se utilizó el hecho de que $(\cos\theta + I_2\sin\theta)^n = \cos n\theta + I_2\sin n\theta$, y $I_2^2 = -1$. Note que la rotación del vector $\mathbf{x}$ se hace a lo largo del plano A únicamente, ya que el vector *rechazado* $\mathbf{x}_\perp$ es el mismo para $\mathbf{x}$ y $R(\mathbf{x})$. La rotación se lleva a cabo, sobre dos veces el ángulo que forman los vectores del plano A. Si queremos hacer rotar a un vector de un extremo del plano A al otro, es decir, del vector $\mathbf{u}$ al vector $\mathbf{v}$, lo que tenemos que hacer es dividir el ángulo entre dos [13]. O sea

$$\begin{aligned} R = \mathbf{uv} &= \cos\tfrac{\theta}{2} + I_2\sin\tfrac{\theta}{2} = e^{I_2\theta/2} \\ R^\dagger = \mathbf{vu} &= \cos\tfrac{\theta}{2} - I_2\sin\tfrac{\theta}{2} = e^{-I_2\theta/2}. \end{aligned} \tag{33}$$

Si los vectores $\mathbf{u}$ y $\mathbf{v}$ no se encuentran en $C(R^2)$ y no son unitarios, el pseudoescalar I_2 no es válido para ese caso. Por eso se debe de crear un nuevo bivector unitario

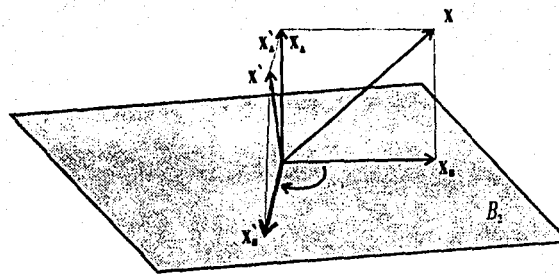

Figura 8: Rotación de un vector en relación a un plano.

que se comporte de la misma forma que el pseudoescalar I_2, la forma de hacerlo es la siguiente. Tomamos el plano A y lo hacemos unitario

$$\hat{A} = \frac{A}{|A|} \tag{34}$$

donde $|\hat{A}| = 1$, $\hat{A}^2 = -1$ y θ el ángulo entre los vectores u y v. Ahora bien, para no tener ese problema en (33), escribimos las ecuaciones de la siguiente forma

$$\begin{aligned} \mathrm{R} = \mathrm{uv} &= \cos\tfrac{\theta}{2} + \hat{A}\sin\tfrac{\theta}{2} = e^{\frac{1}{2}\hat{A}\theta} \\ \mathrm{R}^{\dagger} = \mathrm{vu} &= \cos\tfrac{\theta}{2} - \hat{A}\sin\tfrac{\theta}{2} = e^{-\frac{1}{2}\hat{A}\theta}. \end{aligned} \tag{35}$$

Es muy sencillo encontrar el ángulo entre los dos vectores que forman el plano A, utilizaremos la definición de magnitud del producto escalar para encontrar a θ. Utilizaremos a los dos vectores que forman a A,

$$\begin{aligned} \mathrm{u}\cdot\mathrm{v} &= |\mathrm{u}|\,|\mathrm{v}|\cos\theta \\ \theta &= \arccos\left(\tfrac{\mathrm{u}\cdot\mathrm{v}}{|\mathrm{u}||\mathrm{v}|}\right). \end{aligned} \tag{36}$$

Observe que este resultado es válido para dos vectores u y v dados en una dimensión arbitraria n. Pero para trabajar en $C(R^3)$, se puede resolver un problema teniendo como referencia al vector *normal* n y al ángulo θ. Para ésto sólo encontramos A utilizando la operación dual con repecto a n y se utiliza (35), véase Figura 8. Ahora veamos un ejemplo que nos ayudará a comprender mejor los resultados.

Sea $u = 2e_1, v = 3e_2$ y $x = 2e_1 + e_2 - e_3$, utilizando (35), y con un valor de $\theta = \pi/2$, tenemos

$$\begin{aligned} R^{\dagger}xR &= (\cos\pi/4 - \hat{A}\sin\pi/4)x(\cos\pi/4 + \hat{A}\sin\pi/4) \\ &= (\tfrac{1}{\sqrt{2}} - \hat{A}\tfrac{1}{\sqrt{2}})x(\tfrac{1}{\sqrt{2}} + \hat{A}\tfrac{1}{\sqrt{2}}). \end{aligned} \tag{37}$$

Ahora encontramos el valor de

$$\hat{A} = \frac{A}{|A|} = \frac{u \wedge v}{|u \wedge v|} = \frac{6e_1e_2}{6} = e_1e_2, \tag{38}$$

si sustituímos (38) en (37) y tambien x nos queda el producto

$$\begin{aligned} (\tfrac{1-e_1e_2}{\sqrt{2}})(2e_1 + e_2 - e_3) &= \tfrac{1}{\sqrt{2}}(2e_1 + e_2 - e_3 + 2e_2 - e_1 + e_1e_2e_3) \\ &= \tfrac{1}{\sqrt{2}}(e_1 + 3e_2 - e_3 + e_1e_2e_3), \end{aligned}$$

$$\begin{aligned} \tfrac{1}{\sqrt{2}}(e_1 + 3e_2 - e_3 + e_1e_2e_3)(\tfrac{1+e_1e_2}{\sqrt{2}}) &= \tfrac{1}{2}(e_1 + 3e_2 - e_3 + e_1e_2e_3 \\ &\quad + e_2 - 3e_1 - e_1e_2e_3 - e_3) \\ &= \tfrac{1}{2}(-2e_1 + 4e_2 - 2e_3) \\ &= -e_1 + 2e_2 - e_3. \end{aligned}$$

Como se podrá observar, el vector x se rotó 90 grados. Note que el plano en sí, es el plano xy.

1.4 Líneas, Planos e Hiperplanos.

La interpretación geométrica del producto exterior, ha sido explicada en los subtemas anteriores. En este subtema lo utilizaremos para describir a líneas, planos e hiperplanos de una forma sencilla y fácil de utilizar en dimensiones arbitrarias. Antes de continuar, es necesario dar un nombre para referirnos a líneas, planos e hiperplanos; se utilizará el nombre de *q-plano*, donde un *0-plano* es un punto,un *1-plano* es una línea, un *2-plano* es un plano, etc. [5].

Consideremos un subespacio de dimensión q del espacio euclidiano R^n. Sea $\{e_1, e_2, \ldots, e_q\}$ una base ortogonal de este subespacio. El q-blade formado por el producto geométrico de esta base es

$$B_q = e_1e_2\cdots e_q = e_1 \wedge e_2 \wedge \ldots \wedge e_q.$$

Ahora, un vector $\mathbf{x} \in R^q$ puede escribirse como:

$$\mathbf{x} = k_1\mathbf{e}_1 + k_2\mathbf{e}_2 + \cdots + k_q\mathbf{e}_q,$$

donde $k_1, k_2, \ldots, k_q$ son números reales. Tomando el producto exterior de $\mathbf{x}$ con B_q tenemos que

$$\mathbf{x} \wedge B_q = (k_1\mathbf{e}_1 + k_2\mathbf{e}_2 + \cdots + k_q\mathbf{e}_q) \wedge (\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \ldots \wedge \mathbf{e}_q) = 0,$$

consecuentemente

$$\mathbf{x} \wedge B_q = 0,$$

proporciona una ecuación no paramétrica del q-plano que pasa por el origen y que está representado por el q-blade B_q. Para un q-plano que pasa por un punto arbitario r, la ecuación es

$$(\mathbf{x} - r) \wedge B_q = 0. \tag{39}$$

Desde este punto de vista, un q-plano es la representación gemétrica de un q-blade. B_q da la orientación del q-plano, y los q-planos con direcciones iguales u opuestas se dicen paralelos el uno con el otro.

1.4.1 Líneas.

Una línea puede representarse por dos tipos de ecuaciones: paramétricas y simétricas. Consideremos dos puntos $P_1 = (x_1, y_1, z_1)$ y $P_2 = (x_2, y_2, z_2)$, el vector B_q en la línea es:

$$B_1 = (x_2 - x_1)\mathbf{e}_1 + (y_2 - y_1)\mathbf{e}_2 + (z_2 - z_1)\mathbf{e}_3. \tag{40}$$

Sea $\mathbf{x} = (x, y, z)$ un punto cualquiera y $r = P_1$ un punto fijo. De acuerdo con (39) se tiene:

$$\mathbf{x} - r = (x - x_1)\mathbf{e}_1 + (y - y_1)\mathbf{e}_2 + (z - z_1)\mathbf{e}_3. \tag{41}$$

Haciendo el producto exterior de (40) y (41) el resultado es

$$\begin{aligned}(\mathbf{x} - r) \wedge B_q \; &= (x - x_1)(y_2 - y_1)\mathbf{e}_1\mathbf{e}_2 + (x - x_1)(z_2 - z_1)\mathbf{e}_1\mathbf{e}_3 \\ &\quad -(y - y_1)(x_2 - x_1)\mathbf{e}_1\mathbf{e}_2 + (y - y_1)(z_2 - z_1)\mathbf{e}_2\mathbf{e}_3 \\ &\quad -(z - z_1)(x_2 - x_1)\mathbf{e}_1\mathbf{e}_3 - (z - z_1)(y_2 - y_1)\mathbf{e}_2\mathbf{e}_3 = 0.\end{aligned}$$

Factorizando las $\mathbf{e}_i$ se obtiene

$$\begin{aligned}&[(x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)]\mathbf{e}_1\mathbf{e}_2\\&+[(x-x_1)(z_2-z_1)-(z-z_1)(x_2-x_1)]\mathbf{e}_1\mathbf{e}_3\\&+[(y-y_1)(z_2-z_1)-(z-z_1)(y_2-y_1)]\mathbf{e}_2\mathbf{e}_3=0.\end{aligned}\tag{42}$$

Para que la igualdad de (42) se cumpla debe suceder que

$$\begin{aligned}&(x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)=0\\&(x-x_1)(z_2-z_1)-(z-z_1)(x_2-x_1)=0\\&(y-y_1)(z_2-z_1)-(z-z_1)(y_2-y_1)=0.\end{aligned}\tag{43}$$

De (43) se obtiene

$$\begin{aligned}\frac{(x-x_1)}{(x_2-x_1)}&=\frac{(y-y_1)}{(y_2-y_1)},\\\frac{(x-x_1)}{(x_2-x_1)}&=\frac{(z-z_1)}{(z_2-z_1)},\\\frac{(y-y_1)}{(y_2-y_1)}&=\frac{(z-z_1)}{(z_2-z_1)}.\end{aligned}$$

Los últimos tres resultados nos proporcionan las ecuaciones simétricas de la recta [9, 20]

$$\frac{(x-x_1)}{(x_2-x_1)}=\frac{(y-y_1)}{(y_2-y_1)}=\frac{(z-z_1)}{(z_2-z_1)}=\alpha.\tag{44}$$

Despejando a las variables del punto $\mathbf{x}$ de (43) y sutituyendo los resultados por (44) resulta

$$\begin{aligned}x&=x_1+\alpha(x_2-x_1),\\y&=y_1+\alpha(y_2-y_1),\\z&=z_1+\alpha(z_2-z_1).\end{aligned}\tag{45}$$

Entonces las ecuaciones paramétricas son las de (45) y las ecuaciones simétricas son (44).

En particular, para una recta en R^2, con los puntos $P_1=(x_1,y_1)$ y $P_2=(x_2,y_2)$, con $\mathbf{x}=(x,y)$ y $r=P_1$, se tiene:

$$B_1=(x_2-x_1)\mathbf{e}_1+(y_2-y_1)\mathbf{e}_2\tag{46}$$

y

$$\mathbf{x}-r=(x-x_1)\mathbf{e}_1+(y-y_1)\mathbf{e}_2,\tag{47}$$

entonces el producto exterior de (47) con (46) es

$$\begin{aligned}(\mathbf{x}-r)\wedge B_1 &= (x-x_1)(y_2-y_1)\mathbf{e}_1\mathbf{e}_2-(y-y_1)(\mathbf{x}_2-x_1)\mathbf{e}_1\mathbf{e}_2\\ &= [(x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)]\mathbf{e}_1\mathbf{e}_2=0.\end{aligned}$$

De aquí se deduce que

$$(x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)=0,$$

y despejando a y obtenemos

$$y=\frac{(y_2-y_1)}{(x_2-x_1)}x-\frac{(y_2-y_1)}{(x_2-x_1)}x_1+y_1. \tag{48}$$

Que es la ecuación de la recta. Para ser representada en su forma más conocida hacemos:

$$m=\frac{(y_2-y_1)}{(x_2-x_1)} \tag{49}$$

y

$$b=-\frac{(y_2-y_1)}{(x_2-x_1)}x_1+y_1, \tag{50}$$

lo que da el resultado esperado.

1.4.2 Planos.

Como un ejemplo, encontraremos la ecuación del plano que pasa por el punto $(2,1,-1)$ y tiene como vector normal a $\mathbf{v}=-\mathbf{e}_1+\mathbf{e}_2+3\mathbf{e}_3$. Como $\mathbf{v}$ es perpendicular al plano deseado, su dual es el bivector B_2 que caracteriza este plano. Por lo tanto

$$\begin{aligned}B_2=\bar{v} &= (-\mathbf{e}_1+\mathbf{e}_2+3\mathbf{e}_3)(-\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3)\\ &= \mathbf{e}_2\mathbf{e}_3+\mathbf{e}_1\mathbf{e}_3-3\mathbf{e}_1\mathbf{e}_2.\end{aligned} \tag{51}$$

De $\mathbf{x}=(x,y,z)$ y $r=(2,1,-1)$ tenemos que

$$\mathbf{x}-r=(x-2)\mathbf{e}_1+(y-1)\mathbf{e}_2+(z+1)\mathbf{e}_3. \tag{52}$$

por lo que nuestra ecuación nos queda como

$$\begin{aligned}(\mathbf{x}-r)\wedge B_2 &= (x-2)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3-(y-1)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3-(3z+3)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\\ &= (x-y-3z-4)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3=0.\end{aligned}$$

de donde resulta $-x+y+3z+4=0$.

1.4.3 Hiperplanos.

Para hiperplanos, haremos un ejemplo numérico que nos proporcione un 2-plano en 4 dimensiones [15].

Encontrar la ecuación del 2-plano que tiene como normal a $\mathbf{v} = 3\mathbf{e}_1 - \mathbf{e}_2 + \mathbf{e}_3 + 2\mathbf{e}_4$ y que pasa por el punto $r = (-1, -4, 1, 1)$, con $\mathbf{x} = (x, y, z, w)$.

Observe que necesitamos un B_3 para poderlo resolver, o sea

$$\begin{aligned}
B_3 = \bar{v} &= (3\mathbf{e}_1 - \mathbf{e}_2 + \mathbf{e}_3 + 2\mathbf{e}_4)(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4) \\
&= 3\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4 + \mathbf{e}_1\mathbf{e}_3\mathbf{e}_4 + \mathbf{e}_1\mathbf{e}_2\mathbf{e}_4 - 2\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3, \\
\mathbf{x} - r &= (x+1)\mathbf{e}_1 + (y+4)\mathbf{e}_2 + (z-1)\mathbf{e}_3 + (w-1)\mathbf{e}_4, \\
(\mathbf{x} - r) \wedge B_3 &= (3x+3)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4 - (y+4)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4 \\
&\quad +(z-1)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4 + (2w-2)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4 \\
&= (3x - y + z + 2w - 4)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4 = 0,
\end{aligned}$$

por lo tanto la ecuación es $3x - y + z + 2w - 4 = 0$.

1.5 Relación Entre el Producto Vectorial y el Exterior.

El producto vectorial de dos vectores es otro vector. La característica del vector resultante es ser perpendicular a los vectores que fueron operados. No se puede aplicar a vectores arriba de la dimensión tres. En el álgebra de Clifford, como hemos visto, existe una ecuación llamada dual (ver página 15), que nos proporciona un objeto perpendicular a un dado q-blade.

El producto vectorial de dos vectores paralelos es cero, la magnitud del vector resultante es igual a

$$|\mathbf{a}|\,|\mathbf{b}| \cos\theta,$$

donde $\mathbf{a}$ y $\mathbf{b}$ son los vectores y θ el ángulo entre ellos.

Sabemos que el vector $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ es perpendicular al plano formado por $\mathbf{a}$ y $\mathbf{b}$. Por otro lado, el bivector que caracteriza a este plano es $B_2 = \mathbf{a} \wedge \mathbf{b}$ y un vector normal a este plano esta dado por

$$\bar{B}_2 = (\mathbf{a} \wedge \mathbf{b}) I_3^{\dagger}.$$

Debemos tener entonces que $\mathbf{c} = \tilde{B}_2$, esto es [7, 2]

$$\mathbf{a} \times \mathbf{b} = (\mathbf{a} \wedge \mathbf{b}) I_3^!,$$

o bien

$$\mathbf{a} \wedge \mathbf{b} = I_3 \mathbf{a} \times \mathbf{b}.$$

2 El Sistema *Mathematica.*

2.1 Introducción y Descripción.

Mathematica es un sistema general para realizar operaciones matemáticas [22]. Se puede utilizar de tres maneras:

1) Cálculo numérico. El programa *Mathematica* da la oportunidad de realizar cálculos numéricos con una gran precisión.

2) Cálculo simbólico. En el programa *Mathematica* se puede realizar cálculos simbólicos, como por ejemplo,

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3.$$

3) Generación de gráficos. *Mathematica* tiene la capacidad de realizar gráficas de funciones determinadas.

Mathematica es un programa en lenguaje *C* de aproximadamente 150,000 líneas. El código fuente fué escrito en una versión especial orientada a objetos de *C*, y ha sido compilado en la forma estándar de *C*.

Mathematica representa una síntesis de distintos formatos de paquetes:

* Formas numéricas interactivas de lenguajes como BASIC.

* Sistemas numéricos interactivos como MathCAD, MATLAB y TK!Solver.

* Sistemas algebráicos como Macsyma, Maple, Reduce, Schoonschip, Scratchpad y SMP.

* Interpretación de lenguajes gráficos como PostScript.

* Manipulación de listas numéricas y simbólicas de lenguajes como APL y LISP.

* Estructuras de programación de lenguajes como C y Pascal.

La base en una sesión en *Mathematica* es el llamado *kernel*, que contiene una cantidad extraordinaria de funciones predefinidas que ayuda para el cálculo científico. La sintaxis para utilizar estas funciones es, por lo general, con la primer letra del nombre en mayúscula. En la interfase con el usuario se identifican las entradas y las salidas de una operación por medio de etiquetas de entrada (*In*[*n*]) y salida (*Out*[*n*]), donde *n* es la operación *n*-ésima en la sesión. Una sesión dura desde que se activa hasta que se desactiva el *kernel*.

A continuación unos ejemplos de las operaciones que se hacen en *Mathematica* en una sesión típica.

La primera involucra sólo cálculos numéricos. Se solicita el valor de la función $\log(4\pi)$

```
In[1] := N[Log[4 Pi]]
Out[1] = 2.53102
```

la `N` es la función que indica que el resultado debe ser numérico. El mismo ejemplo pero con mayor precisión es

```
In[2] := N[Log[4 Pi], 20]
Out[2] = 2.5310242469629079297
```

Los cálculos simbólicos se realizan de la siguiente forma

```
In[3] := x^4/(x^2 - 1)
```

$$Out[3] = \frac{x^4}{-1 + x^2}$$

La integral de la función anterior puede obtenerse así

```
In[4] := Integrate[%, x]
```

$$Out[4] = x + \frac{x^3}{3} + \frac{\text{Log}[-1+x]}{2} - \frac{\text{Log}[1+x]}{2}$$

La función "%" se usa para denotar el resultado inmediato anterior, (se pudo haber utilizado la instrucción *Out*[3]), El segundo argumento del comando `Integrate` indica la variable de integración.

Como un ejemplo de la capacidad gráfica de *Mathematica*, se graficará la función *seno*(xy):

```
In[5] := Plot3D[Sin[x y], {x, 0, Pi}, {y, 0, Pi}]
Out[5] = -Graphics-
```

La gráfica resultante se presenta en la figura 9. En la función `Sin[x y]` hay un espacio entre x y y que se interpreta como un producto. Para evitar confusiones, la manera correcta es:

```
Plot3D[Sin[x * y], {x, 0, Pi}, {y, 0, Pi}]
```

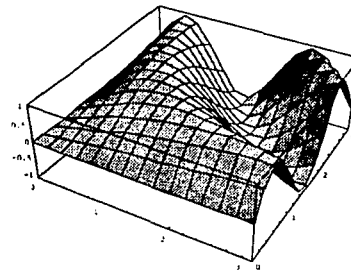

Figura 9: Gráfica del sin(xy)

Mathematica permite la definición de funciones propias del usuario que pueden utilizarse exclusivamente en una sesión. Por ejemplo,

$$In[6] := \texttt{f[n\_] := Table[Prime[i], \{i, n\}]} \tag{53}$$

La función `f[n_]` genera una *tabla* de los primeros n primos. La instrucción `":="` es conocida con el nombre de *regla retrasada*, es decir, no proporcionará un valor a `f[n]` hasta que sean evaluadas las funciones intrínsecas de *Mathematica* `Table[]` y `Prime[]` según sea el valor de n. Una vez definida, se puede emplear así

$$In[7] := \texttt{f[10]}$$
$$Out[7] = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}$$

Si se vuelve a pedir el valor de `f[10]`, las instrucciones de la función serán evaluadas de nuevo, esto quiere decir que `f[10]` no es una variable sino una función que se evaluará cada vez que se invoque. El término del lado derecho de una *regla retrasada* debe estar en una forma no evaluada, es decir, debe tener un valor indefinido que pertenece a los parametros del lado izquierdo de la regla.

A cualquier resultado, ya sea numérico, simbólico o gráfico, se le puede asignar una variable, por ejemplo, en forma comparativa

$$In[8] := \%$$
$$Out[8] = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}$$

$$In[9] := v = f[10]$$
$$Out[9] = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}$$

A las tablas generadas en las salidas 7 y 9 se les conocen con el nombre de *listas*, que se pueden mostrar, insertar o eliminar los datos de cualquier posición.

$$In[10] := \%7[[3]]$$
$$Out[10] = 5$$

$$In[11] := v[[3]]$$
$$Out[11] = 5$$

En la entrada 10 la instrucción "%7" se usa para designar a la salida número 7, y en la entrada 11 los paréntesis anidados [[i]] se usan para mostrar el i-ésimo elemento de una *lista* $\{\ldots\}$, en este caso, el tercer elemento. Como 5 está colocado en la tercera posición de la tabla, 5 es el elemento extraído. Obsérvese que en la entrada 9 se usó una *regla normal*, lo que implica que será evaluada en el momento de su definición.

En las entradas $In[10]$ e $In[11]$ vimos cómo mostrar el i-ésimo elemento de una lista; para insertar datos en una lista se utiliza el comando `Insert[]`, por ejemplo

$$In[12] := \texttt{Insert}[v, 10, 5]$$
$$Out[12] = \{2, 3, 5, 7, 10, 11, 13, 17, 19, 23, 29\}$$

Tambien se pudo haber hecho como

$$In[13] := \texttt{Insert}[v, 10, -5]$$
$$Out[13] = \{2, 3, 5, 7, 11, 13, 10, 17, 19, 23, 29\}$$

Observe que cuando la posición donde se debe de insertar el dato es un número negativo, cuenta desde el final hacia el principio de la lista. El comando para eliminar un elemento de la lista es llamado `Drop[]`, para entenderlo haremos un ejemplo

$$In[14] := \texttt{Drop}[v, \{5, 5\}]$$
$$Out[14] = \{2, 3, 5, 7, 13, 17, 19, 23, 29\}$$

Existen otras dos funciones que pueden insertar datos que son `Prepend[]` y `Append[]`, que insertan un elemento al principio y al final de una lista respectivemente. O sea

$$In[15] := \texttt{Prepend}[v, 1]$$
$$Out[15] = \{1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}$$

$$In[16] := \texttt{Append[v, 1]}$$
$$Out[16] = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 1\}$$

Observe que en las entradas 12, 13, 14, 15 y 16, el valor de v no fué modificado, sólo se fueron mostrando los resultados. Para que las modificaciones se conserven en la lista, la variable v debe igualarse a las funciones, o sea v = Append[v, 1], y ahora v será la lista de la salida 16.

Nuestra intención en esta sección fué sólo dar una introducción al sistema *Mathematica*, sin pretender ser exhaustivos. Las funciones preconstruidas son muchas y una referencia completa es, por supuesto, el manual [22].

2.2 El Lenguaje de Programación en *Mathematica*.

Mathematica tiene su propio lenguaje de programación que describiremos brevemente a continuación. Es importante mencionar primero que con el lenguaje de *Mathematica* hay tres formas diferentes de programar. La primera que analizaremos es llamada *funcional* [8], porque se crea una función que corresponde directamente a una sola instrucción. Otra forma de programar es por *procedimientos*; este tipo de programación se identifica porque utiliza la instrucción `Module[]`, esta instrucción nos sirve para agrupar un conjunto de instrucciones y generar *variables locales* que serán exclusivas del procedimiento sin ser afectadas por *variables externas* que se generan en la sesión. Es muy común utilizar instrucciones de programación conocidas como *for*, *if then else* [22], etc. Por último veremos la programación *recursiva*, que se caracteriza por estar definida para un valor n y tener una forma de terminar.

2.2.1 Programación Funcional.

Un ejemplo de este tipo de programación fué dado en (53) y como podrán ver está igualada a una intrucción llamada `Table[]`. Otro ejemplo es el definir una función para el logaritmo de un producto

$$\texttt{log[x\_ y\_] := log[x] + log[y]}$$

O bien, definir la regla de la derivada para una variable x^n

$$\texttt{d[x\_\^{}n\_] := n * x\^{}(n - 1)}$$

2.2.2 Programación por Procedimientos.

Para este tipo de programación. Describiremos las instrucciones básicas que intervienen en los procedimientos. Para hacer ciclos *Mathematica* tiene las instrucciones *For* y *Do*. La primera tiene la siguiente sintaxis

```
For[ inicio, condición, incremento, instrucciones]        (54)
```

La variable empieza a correr desde *inicio* y posteriormente se incrementa por *incremento* y las *instrucciones* se realizan mientras *condición* sea verdadero. Por ejemplo, podemos crear una función que nos proporcione la suma desde 1 hasta n, o sea

```
In[1] := sumatoria[n_] := Module[{i, j = 0},
    For[i = 1, i <= n, i++, j+= i];
  j ]
```

Que se puede utilizar así:

```
In[2] := sumatoria[10]
Out[2] = 55
```

Aquí se puede ver bien el trabajo de la instrucción `Module`; es decir, si se iguala una variable j al resultado de `sumatoria[n]` no se altera la función porque la variable j interna es exclusiva de la función y la variable j externa es una variable denominada *Global* (se definirá este concepto después), por ejemplo

```
In[3] := j = sumatoria[5]
Out[3] = 15
```

La otra instrucción para hacer ciclos es `Do`, que tiene como sintaxis

```
Do[ instrucciones, {variable,inicio,fin,incremento}]      (55)
```

donde *variable* controla el ciclo; `inicio` e `incremento` son el inicio y el incremento de ésta, y `fin` donde termina el ciclo. Si se omiten `inicio` e `incremento`, éstos tendrán el valor: `inicio=1` e `incremento=1`.

Para ejemplificar haremos el mismo ejemplo de la función `sumatoria[n_]`

```
In[4] := sumatoria[n_] := Module[{j = 0},
  Do[j += i, {i, n}];
  j ]
```

Para la variable *j* es el mismo caso que el anterior y la variable *i* es exclusiva de `Do`. Por ejemplo

```
In[5] := i = sumatoria[100]
Out[5] = 5050
```

La variable *i* no afecta a la función `sumatoria`, quiere decir que no se genera una doble variable, *i.e.*, la variable *i* en *In*[5] es una variable externa e *i* en *In*[4] es una variable local. Los *contextos* de las dos variables *i*'s son distintos, ésto es lo que hace que no se confundan las variables, pues la máquina interpreta internamente a la variable *i* en *In*[5] como *Global'i* y en *In*[4] como *Global'sumatoria'Do'i*, o sea, es como si hubieramos puesto

```
In[5] := Global'i = sumatoria[100]
```

Y la función `sumatoria` se hubiera escrito como

```
In[4] := sumatoria[n_] := Module[{j = 0},
  Do[j += Global'sumatori'Do'i,
    {Global'sumatoria'Do'i, n}];
  j ]
```

Es por ello que son distintas.

Como mencionamos en un principio, en *Mathematica* existen muchas funciones predefinidas que realizan operaciones matemáticas. La función `sumatoria[n_]`, por ejemplo, que ha sido creada, puede ser sustituída por la función interna `Sum`, cuya sintaxis es:

```
Sum[f, {i, imin, imax, di}, {j, jmin, jmax, dj}, ...]
```

Se evalúa la función f con respecto a las variables i y/o j (se puede omitir j), desde $i =$ `imin`, hasta `imax` con un incremento de `di` (se pueden omitir `imin` y `di`).

Otro elemento importante en un lenguaje de programación son las condicionales. *Mathematica* cuenta con `If`, `Condition`, entre otras. la primera tiene una sintaxis

```
If[condición, instrucciones1, instrucciones2]                    (56)
```

Si el resultado de condición es verdadero, se realiza `intrucciones1`, en caso contrario se realiza `instrucciones2`. Por ejemplo

```
In[6]:= raizcuadrada[x_] := If[x < 0,
          Print[Sqrt[-x], ''i''],
          Print[Sqrt[x]]     ]
```

que sirve para calcular la raíz cuadrada de un número negativo simplemente multiplicando el resultado por i, es decir

```
In[7]:= raizcuadrada[-9]
3I
```

Resultaría difícil encontrar funciones que no estén incluidas en el *kernel* de *Mathematica*.

Existen otras formas de programar con condiciones, por ejemplo, una función que nos dé la raíz cuadrada de un número positivo es

```
In[8]:= raizcuadrada[x_] := Sqrt[x] /; x >= 0
```

La condición "`/;`" significa que se realizará la operación que se encuentra a su izquierda siempre y cuando sea verdadera la condicción de su derecha. Así se pueden obtener dos salidas, dados número negativos y positivos

```
In[9]:= raizcuadrada[144]
Out[9] = 12

In[10]:= raizcuadrada[-9]
Out[10] = raizcuadrada[-9]
```

2.2.3 Programación Recursiva.

Otra forma de programar es por medio de los métodos recursivos, un ejemplo de ello es la función que genera el enésimo término de Fibonacci: 1,1,2,3,5,8,...,(n-2)+(n-1); o sea

$$In[11] := \texttt{fibonacci[0]} = \texttt{fibonacci[1]} = 1;$$
$$\texttt{fibonacci[n\_]} := \texttt{fibonacci[n} - 1\texttt{]} + \texttt{fibonacci[n} - 2\texttt{]}$$

Esta función se detiene cuando calcula `fibonacci[0]` o `fibonacci[1]`. No es muy eficiente esta forma ya que para $n = 6$, cuando calcula el valor de 4 lo hace para 6 y para 5. Hay una forma de evitar esa pérdida de doble cálculo y es como sigue

$$In[12] := \texttt{fibonacci[n\_]} := \texttt{fibonacci[n]} =$$
$$\texttt{fibonacci[n} - 1\texttt{]} + \texttt{fibonacci[n} - 2\texttt{]}$$

No hubo necesidad de definir la forma de término, pues fué definida en $In[11]$. Ahora el valor de 4 para $n = 6$ no se calculará dos veces ya que se conserva en `fibonacci[4]`. Esta función sólo maneja números enteros, si se pide un número fraccionario la función no se detendrá pues nunca encontrará el valor de 0 o 1. Entonces la función más eficiente de la serie de números de Fibonacci será

$$In[13] := \texttt{fibonacci[n\_Integer]} := \texttt{fibonacci[n]} =$$
$$\texttt{fibonacci[n} - 1\texttt{]} + \texttt{fibonacci[n} - 2\texttt{]}$$

Con esto nos aseguramos que la entrada siempre será un número entero.

2.3 Paquetes.

Hay dos formas de crear programas en *Mathematica*. Una de ellas es conocida como *cuaderno de trabajo* (*Notebook*), que es salvado con extensión *"*.MA"*; consiste en crear funciones en el editor de *Mathematica*, tambien se pueden generar gráficas, y ser salvados con un mismo nombre. La otra es creando un archivo *ASCII* y salvarlo con extención *"*.M"*; puede contener las mismas funciones que el *"*.MA"*. La diferencia entre un *"*.MA"* y un *"*.M"* es que el *"*.MA"* presenta las funciones en pantalla y deben de compilarse una por una y en un *"*.M"* no las presenta en pantalla y se compilan todas al mismo tiempo. Las gráficas que son salvadas en un *"*.MA"* al ser

compiladas esas funciones presentan la gráfica en pantalla, en cambio en un *"*.M"* las funciones que generan gráficas no pueden ser salvadas con valores específicos. En un *"*.MA"* se pueden hacer escritos con respecto a las funciones y qué es lo que hacen, así como el hacer presentaciones, ya que al ser recuperado el archivo presenta todo en pantalla; en cambio un *"*.M"* lo único que se puede hacer es poner comentarios entre los símbolos *(*comentario*)*, pero no son presentados en pantalla. A los archivos con extención *"*.M"* son conocidos como *Paquetes*.

La estructura de un *Paquete* en *Mathematica* está definida por los *Contextos*. Existen dos tipos de éstos, el *Contexto Global* y el *Contexto Privado*. El *Cotexto Global* es utilizado en las sesiones de *Mathematica*, indicando que cualquier variable o función que sea creada en la sesión se borrará su definición para cuando sea renovada, y si es parecida a una función de *Mathematica* ya implícita marcará una precaución con símbolos parecidos. Se puede ver que el *Contexto Global* es el nivel más cercano al usuario; es decir, lo que se escribe en un *Cuaderno de Trabajo* y en una sesión son de *Contexto Global*. Con respecto al *Contexto Privado* se tiene que todo lo que se encuentra en él es exclusivo del *Paquete* y que se crea en memoria las variables que se trabajan en él y desaparecen cuando deja de trabajar. El *Contexto Privado* es un nivel que el usuario no puede accesar; por ejemplo, existe una función en el *Contexto Privado* y el usuario escribe una función con el mismo nombre, pero en su sesión, no ocurre algún error porque la función en el *Contexto Privado* es exclusivamente de éste, y trabajar la función del *Contexto Privado* no se afecta por la función que creó el usuario [17].

Se presentará un esquema de un *Paquete* y lo analizaremos parte por parte:

```
1.  Begin Package["Paquete1"']
2.  Needs["Paquete2"']
3.  Paquete1::usage = "Paquete1.m es un paquete que no hace nada"
4.  Función1::usage = "Función1[n] no hace nada"
5.  Función2::usage = "Función2[n,m:17] tampoco hace nada"
6.  Begin["'Private"']
7.  protected = Unprotect[Sin, Cos]
```

```
8.   Aux[f_] := Do[lo que sea]
9.   statievar = 0
10.  Paquete1::badarg = "Precaución, llamaste a '1' con argumento '2"'
11.  Función1[n_] := n
12.  Función2[n_,m:17] := n m /; n < 5 || Message[Paquete1::badarg, Función2,
n]
13.  Sin/: Sin[x_]^2 := 1 - Cos[x]^2
14.  Protect[ Evaluate[protected] ]
15.  End[]
16.  Protect[ Función1, Función2 ]
17.  EndPackage[]
```

1. Comienza el paquete Paquete1 y tiene la capacidad de que todo lo que se maneje en él tiene un *Contexto Global*, a menos que se indique lo contrario.

2. Utilización de otros paquetes si es necesario.

3. Esta instrucción da la pauta para que Paquete1 sea utilizado en un *Contexto Global*, así como proporcionar información cada vez que se requiera. Esta información puede ser obtenida por medio de la función ?, por ejemplo su sintaxis sería

```
In[15] := ?Paquete1
Out[15] = Paquete1.m es un paquete que                    (57)
          no hace nada
```

4. Hace lo mismo que 3. pero para la Función1.

5. Hace lo mismo que 4. pero para la Función2.

6. Da comienzo al *Contexto Privado*.

7. Desactiva las protecciones de las funciones Sin y Cos para que se puedan redefinir en el paquete; esto es, las definiciones de las funciones que antes de ser invocado *Paquete1* son removidas para que trabajen con respecto a la definición del paquete. No solamente se puede hacer con Sin y Cos sino con cualquier función que así lo requiera el paquete.

8. Funciones auxiliares del paquete. Son funciones que el usuario nunca va a poder utilizar.

9. Definición de variables locales (estáticas).

10. Control de salidas de error por un mal manejo de funciones por parte del usuario.

11. Definición de la Función1.

12. Definición de la Función2 y utilización de un posible mensaje de error en la función Message[].

13. Definición de la salida del símbolo Sin que genera *Paquete1* dado su entrada Sin[x]^2. Sólo existe esta definición, por lo tanto, si entra otra no tiene control de salida.

14. Vuelve a proteger los símbolos desprotegidos para que vuelva a tener la definición que había en el *Contexto Global.*

15. Fin del *Contexto Privado.*

16. Protección de los símbolos (funciones) para que el usuario no los redefina como otras funciones creadas por él.

17. Fin del paquete.

Ahora, *Mathematica* tiene un subdirectorio llamado *PACKAGES* y es en ese subdirectorio donde se deben guardar los paquetes, ya que al ser invocados en una sesión el lugar donde los busca el sistema es en ese subdirectorio. Supongamos que Paquete1 está en ese subdirectorio, para invocarlo su sintaxis es

In[14] := << "Paquete1.m"

Y ahora sí se pueden utilizar a Función1 y a Función2 así como Sin[x]^2. Observe que (57) sólo puede ser utilizado después de haber invocado a Paquete1.

3 Paquete para Cálculos con Algebra de Clifford.

3.1 Esquema del Paquete

Las operaciones básicas de un paquete que realice cálculos en un álgebra de Clifford son el producto geométrico y el operador de grado. El algoritmo que calcula el primero de ellos esta basado en el siguiente razonamiento. Consideremos el álgebra de Clifford de R^n, $\mathcal{C}(R^n)$. Como un multivector se forma de combinaciones lineales de r-blades, el producto más simple que podemos realizar es entre dos blades A_r y B_q. Considerando la base canónica de R^n, cada blade es un múltiplo de

$$e_1^{m_1} e_2^{m_2} ... e_n^{m_n}, \tag{58}$$

donde m_i $(i = 1, .., n)$ toma el valor de 1 o 0, por lo que podemos establecer un isomorfismo entre el r-blade y la eneada $(m_1, m_2, ..., m_n)$. El producto de elementos de la forma (58) es

$$(e_1^{m_1} e_2^{m_2} ... e_n^{m_n})(e_1^{r_1} e_2^{r_2} ... e_n^{r_n}) = (-1)^s e_1^{m_1+r_1} e_2^{m_2+r_2} ... e_n^{m_n+r_n}, \tag{59}$$

donde

$$s = \sum_{1 \le i < j \le n} r_i m_j. \tag{60}$$

Para que el producto de A_r y B_q sea también un blade, la suma $m_i + r_i$ debe hacerse módulo 2 [5].

Como un ejemplo, consideremos $\mathcal{C}(R^4)$ y el producto de los blades

$$A_2 = 7e_1e_3 \quad \text{y} \quad B_3 = ce_1e_2e_4,$$

que, de acuerdo con el isomorfismo señalado, éstos pueden verse como

$$A_2 = 7e_1^1e_2^0e_3^1e_4^0 \quad \text{y} \quad B_3 = ce_1^1e_2^1e_3^0e_4^1.$$

De (59), el producto nos da

$$A_2B_3 = (-1)^2 7c e_1^{1+1} e_2^{0+1} e_3^{1+0} e_4^{0+1},$$

haciendo la suma módulo 2 en los exponentes de los vectores canónicos tendremos

$$\begin{aligned} A_2B_3 &= 7c\ \mathbf{e}_1^0\mathbf{e}_2^1\mathbf{e}_3^1\mathbf{e}_4^1 \\ &= 7c\ \mathbf{e}_2\mathbf{e}_3\mathbf{e}_4. \end{aligned}$$

Ahora, como cualquier multivector $A \in C(R^n)$ se puede descomponer en la suma de blades:

$$A = \sum_{i=0}^{n} \langle A \rangle_i = \sum_{i=0} \left(\lambda_i \prod_{k=1}^{n} \mathbf{e}_k^{m_{ik}} \right), \tag{61}$$

donde λ_i son escalares. Observe que m_0 es un vector $\mathbf{0}$ de dimensión n, el producto geométrico de dos multivectores, A y B, se puede expresar como:

$$\begin{aligned} AB &= \left(\sum_{i=0}^{n} \langle A \rangle_i \right) \left(\sum_{j=0}^{n} \langle B \rangle_j \right) \\ &= \left(\sum_{i=0} (\lambda_i \textstyle\prod_{k=1}^{n} \mathbf{e}_k^{m_{ik}}) \right) \left(\sum_{j=0} \left(\beta_j \textstyle\prod_{k=1}^{n} \mathbf{e}_k^{r_{jk}} \right) \right) \\ &= \sum_{i=0} \sum_{j=0} \left(\lambda_i \beta_j \textstyle\prod_{k=1}^{n} \mathbf{e}_k^{(m_i + r_j)_k} \right), \end{aligned}$$

que involucra sólo productos geométricos de r-blades y donde λ_i y β_j son escalares.

Siguiendo la idea de establecer un isomorfismo entre un r-blade y una eneada de unos y ceros, proponemos el siguiente algoritmo para el cálculo de un producto geométrico entre multivectores.

1. Sean A y $B \in C(R^n)$ dos multivectores, descomponer a A y B de la siguiente forma:

$$\begin{aligned} A &= \sum_{i=0} (\lambda_i \textstyle\prod_{k=1}^{n} \mathbf{e}_k^{m_{ik}}) \quad \text{y} \\ B &= \sum_{j=0} \left(\beta_j \textstyle\prod_{k=1}^{n} \mathbf{e}_k^{r_{jk}} \right), \end{aligned} \tag{62}$$

donde m_i y r_j son eneadas.

2. Realizar el producto geométrico AB quedando

$$\begin{aligned} AB &= \sum_{i=0} \sum_{j=0} \left((-1)^s \lambda_i \beta_j \textstyle\prod_{k=1}^{n} \mathbf{e}_k^{m_{ik} + r_{jk}} \right) \\ s &= \textstyle\sum_{1 \le p < q \le n} r_{jp} m_{iq} \end{aligned}$$

donde $m_{ik} + r_{jk} = (m_{ik} + r_{jk})\ Mod\ 2$.

Veamos un ejemplo preciso. Sea $A = a_1 + a_2\mathbf{e}_2\mathbf{e}_3$ y $B = b_1\mathbf{e}_1 + b_2\mathbf{e}_3 + b_3\mathbf{e}_1\mathbf{e}_2$ dos multivectores de $C(R^3)$, siguiendo el algoritmo tenemos que

Paso 1.

$$A = a_1 e_1^0 e_2^0 e_3^0 + a_2 e_1^0 e_2^1 e_3^1 \quad y$$
$$B = b_1 e_1^1 e_2^0 e_3^0 + b_2 e_1^0 e_2^0 e_3^1 + b_3 e_1^1 e_2^1 e_3^0.$$

Observe que para A sus enceadas son $m_0 = (0,0,0)$ y $m_1 = (0,1,1)$ esto es debido a (62).

Paso 2.

$$\begin{aligned} AB &= a_1 e_1^0 e_2^0 e_3^0 (b_1 e_1^1 e_2^0 e_3^0 + b_2 e_1^0 e_2^0 e_3^1 + b_3 e_1^1 e_2^1 e_3^0) \\ &+ a_2 e_1^0 e_2^1 e_3^1 (b_1 e_1^1 e_2^0 e_3^0 + b_2 e_1^0 e_2^0 e_3^1 + b_3 e_1^1 e_2^1 e_3^0) \\ &= (-1)^0 a_1 b_1 e_1^1 e_2^0 e_3^0 + (-1)^0 a_1 b_2 e_1^0 e_2^0 e_3^1 + (-1)^0 a_1 b_3 e_1^1 e_2^1 e_3^0 \\ &+ (-1)^2 a_2 b_1 e_1^1 e_2^1 e_3^1 + (-1)^0 a_2 b_2 e_1^0 e_2^1 e_3^0 + (-1)^3 a_2 b_3 e_1^1 e_2^0 e_3^1 \\ &= a_1 b_1 e_1 + a_1 b_2 e_3 + a_1 b_3 e_1 e_2 + a_2 b_1 e_1 e_2 e_3 + a_2 b_2 e_2 - a_2 b_3 e_1 e_3 \\ &= a_1 b_1 e_1 + a_2 b_2 e_2 + a_1 b_2 e_3 + a_1 b_3 e_1 e_2 - a_2 b_3 e_1 e_3 + a_2 b_1 e_1 e_2 e_3. \end{aligned}$$

Una vez definido el producto geométrico, el siguiente paso consiste en definir el operador de grado.

El siguiente algoritmo extrae la parte de grado r de un multivector $M \in C(R^n)$, esto es, $\langle M \rangle_r$.

1. Al multivector M se le descompone de la forma (61), como se muestra en (62) para A o B.
2. Sea p_j la suma de las componentes de las enceadas del multivector

$$p_j = \sum_{k=1}^{n} m_{jk}.$$

3. Se comparan todos los p_j con el valor de r.
4. Tomar todos los r-blades correspondientes a todas las comparaciones en el paso 3 que sean iguales a r.

Tomemos como ejemplo el siguiente multivector,

$$M = a_1 b_1 + a_2 b_2 + (a_1 b_2 - a_2 b_1)\, e_1 e_2$$

al que le extraeremos un el término de grado 2

Paso 1.

$$M = (a_1b_1 + a_2b_2)\,\mathbf{e}_1^0\mathbf{e}_2^0 + (a_1b_2 - a_2b_1)\,\mathbf{e}_1^1\mathbf{e}_2^1$$
$$m_0 = (0,0),\ m_1 = (1,1)$$

Paso 2.

$$p_0 = 0 + 0 = 0,\ p_1 = 1 + 1 = 2$$

Paso 3.

$$p_0 \neq r,\ p_1 = r$$

Paso 4.

$$(M)_2 = (a_1b_2 - a_2b_1)\mathbf{e}_1\mathbf{e}_2$$

Todas las operaciones en un álgebra de *Clifford* están definidas por medio del *producto geométrico* y el *operador de grado* (ver capítulo I), por lo que tenemos ya los ingredientes básicos.

No describiremos los detalles del algoritmo, que puede consultarse en el apéndice A. En cambio, por considerarlo más útil, describiremos todas las funciones que fueron construidas en el paquete, lo que es básicamente una guía del usuario.

3.2 Descripción de las Funciones Implícitas.

3.2.1 Comentario general

El paquete fué llamado **Clifford.** En una sesión con **'Clifford'** los vectores canónicos $\mathbf{e}_i$ se escribe de acuerdo con la sintaxis [22]:

```
e[i]                                                    (63)
```

En seguida mostramos 3 ejemplos de multivectores y su sintaxis correcta

```
a = a1 * e[1] + a2 * e[2]
A = a1 * b1 + a2 * b2 + (a1 * b2 - a2 * b1) e[1]e[2]
B = 17 * e[1] + a^2 * e[1] e[2] e[5].
```

Es necesario recalcar que el orden de los vectores canónicos en una sesión en *Mathematica*, es muy importante. Ya que si se escribe el producto de *In*[1] := e[2] * e[1], automáticamente el editor lo presentará como *Out*[1] = e[1] e[2]; que no es correcto, pues $\mathbf{e}_2\mathbf{e}_1 \neq \mathbf{e}_1\mathbf{e}_2$.

3.2.2 CrossProduct[v,w]

Alias: Cp.

Descripción: Calcula el producto cruz de los vectores v y w.

Argumentos: v y w son vectores.

Comentarios: Es auxiliada por las funciones Dual, Oup y HQ. Esta última determina si v y w son vectores.

3.2.3 DotProduct[v,w]

Alias: Dp.

Descripción: Calcula el producto punto de los vectores v y w.

Argumentos: v y w son vectores.

Comentarios: Se auxilia de las funciones Ip y HQ.

3.2.4 Dual[m,d]

Alias: Ninguno.

Descripción: Calcula el dual de un r-blade m que se encuentra en un espacio de dimensión d, resultando un (d-r)-blade.

Argumentos: m es un blade de grado r, y d es un entero positivo.

Comentarios: Utiliza las funciones Gp, Pseudoscalar y Turn que estan contenidas en el paquete.

3.2.5 GeometricCos[m,n]

Alias: GCos.

Descripción: Calcula la función `Cos[m]`, donde `m` es un multivector. El cálculo se basa en un desarrollo en serie de Taylor de la función hasta una potencia n, alrededor del cero.

Argumentos: `m` es un multivector y `n` es un entero positivo.

Comentarios: Si se omite `n`, por default, su valor será de 10.

3.2.6 `GeometricExp[m,n]`

Alias: `GExp`.

Descripción: Calcula la serie de potencia de la función Exponencial del multivector `m` hasta una potencia `n`, al rededor del punto cero.

Argumentos: `m` es un multivector y `n` es un entero.

Comentarios: Si se omite `n`, por default, su valor será de 10.

3.2.7 `GeometricPower[m,n]`

Alias: `GPower`.

Descripción: Eleva el multivector `m` a una potencia `n`.

Argumentos: `m` es un multivector y `n` es un entero.

Comentarios: Realiza el Producto Geométrico del multivector `m`, `n`-veces.

3.2.8 `GeometricProduct[m1,m2,...]`

Alias: `Gp`.

Descripción: Calcula el producto geométrico de los multivectores `m1,m2,...`

Argumentos: `m1,m2,...`son multivectores cualesquiera.

Comentarios: Sigue el algoritmo correspondiente al producto geométrico descrito en la sección anterior. Está respalada por funciones privadas que son: `maxdim` y `ntuple`.

3.2.9 `GeometricProductSeries[sym,m,n]`

Alias: `GSeries`.

Descripción: Calcula la serie de potencias de la función `sym` con respecto al multivector `m`, hasta una potencia `n`.

Argumentos: `sym` es una función definida de *Mathematica*, `m` es un multivector y `n` es un entero positivo.

Comentarios: `sym` puede ser cualquier función que se pueda representar en una serie de potencias al rededor del punto cero. Y si omitimos el valor de `n`, por default, su valor será 10.

3.2.10 `GeometricSin[m,n]`

Alias: `GSin`.

Descripción: Calcula la serie de potencia de la función Seno del multivector `m` hasta una potencia `n`, al rededor del punto cero.

Argumentos: `m` es un multivector y `n` es un entero positivo.

Comentarios: Si se omite `n`, por default, su valor será de 10.

3.2.11 `GeometricTan[m,n]`

Alias: `GTan`.

Descripción: Calcula la serie de potencia de la función Tangente del multivector `m` hasta una potencia `n`, al rededor del punto cero.

Argumentos: `m` es un multivector y `n` es un entero positivo.

Comentarios: Si se omite n, por default, su valor será de 10.

3.2.12 `Grade[m,r]`

Alias: Ninguno.

Descripción: Extrae la parte de grado `r` del multivector `m`.

Argumentos: `m` es multivector y `r` es un entero mayor o igual a 0.

Comentarios: Sigue el algoritmo correspondiente al operador de grado descrito en la sección anterior. Se auxilia de la función privada **grados**.

3.2.13 `HomogeneousQ[b,r]`

Alias: HQ.

Descripción: Regresa el valor de `True` si el blade es de grado `r` y `False` en caso contrario.

Argumentos: `b` es un blade y `r` es un entero mayor o igual a 0.

Comentarios: Es auxiliada por la función `Grade`. Con esta función se puede saber si un multivector es, o no, homogéneo de grado `r`.

3.2.14 `Im[q]`

Alias: Ninguno.

Descripción: Extrae la parte vectorial de un cuaterno `q`.

Argumentos: `q` es un cuaterno.

Comentarios: Presenta el resultado en forma de un vector. Es auxiliada por las funciones `maxdim`, `coef` y `transform` que son privadas. Se tuvo que desproteger de su definición original.

3.2.15 InnerProduct[m1,m2,...]

Alias: Ip.

Descripción: Realiza el producto interno de los multivectores m1,m2,...

Argumentos: m1,m2,... son multivectores.

Comentarios: Utiliza las funciones Gp y Grade.

3.2.16 Magnitude[m]

Alias: Ninguno.

Descripción: Calcula la magnitud de un multivector m.

Argumentos: m es un multivector.

Comentarios: Utiliza las funciones Gp, Turn y Grade.

3.2.17 MultivectorInverse[m]

Alias: Mvi.

Descripción: Calcula el inverso de un multivector, si es que existe.

Argumentos: m es un multivector.

Comentarios: Es auxiliada por las funciones Gp, Grade y Turn para saber si es invertible o no. Utiliza las funciones Turn y Magnitude.

3.2.18 OuterProduct[m1,m2,...]

Alias: Oup.

Descripción: Realiza el producto exterior de los multivectores m1,m2,...

Argumentos: m1,m2,... son multivectores.

Comentarios: Utiliza las funciones Gp y Grade.

3.2.19 `Projection[v,b]`

Alias: Ninguno.

Descripción: Calcula la proyección del vector v sobre el espacio generado por el blade b.

Argumentos: v es un vector y b es un r-blade.

Comentarios: Es auxiliada por las funciones `Gp`, `Ip` y `Mvi`.

3.2.20 `Pseudoescalar[n]`

Alias: Ninguno.

Descripción: Genera el pseudoescalar de dimensión n.

Argumentos: n es un entero positivo.

Comentarios: Ninguno.

3.2.21 `QuaternionConjugate[q]`

Alias: `Qc`.

Descripción: Calcula el conjugado del cuaterno q.

Argumentos: q es un cuaterno.

Comentarios: Se auxilia de la función `Turn` y de las funciones privadas `transform` y `untransform`.

3.2.22 `QuaternionInverse[q]`

Alias: `Qi`.

Descripción: Calcula el inverso del cuaterno q.

Argumentos: q es un cuaterno.

Comentarios: Auxiliada por las funciones `Mvi`, `transform` y `untransform`.

3.2.23 `QuaternionMagnitude[q]`

Alias: `Qm`.

Descripción: Calcula magnitud del cuaterno `q`.

Argumentos: `q` es un cuaterno.

Comentarios: Auxiliada por las funciones `Magnitude`, `transform` y `untransform`.

3.2.24 `QuaternionProduct[q1,q2,...]`

Alias: `Qp`.

Descripción: Calcula el producto de los cuaternos `q1`, `q2`, ...

Argumentos: `q1`, `q2`,... son cuaternos

Comentarios: Es auxiliada por las funciones `Gp`, `transform` y `untransform`.

3.2.25 `Re[q]`

Alias: Ninguno.

Descripción: Extrae la parte real del cuaterno `q`.

Argumentos: `q` es un cuaterno.

Comentarios: Es auxiliada por las funciones `Grade` y `transfor`. Fue necesario desproteger a la función de su definición original.

3.2.26 `Reflection[v,w,x]`

Alias: Ninguno.

Descripción: Calcula la reflexión especular de un vector `v` por el plano formado por los vectores `w` y `x`.

Argumentos: `v`,`w`,`x` son vectores.

Comentarios: Las funciones que intervienen son `Gp`, `Oup`, `Dual` y `HomogeneousQ`. Esta última determina si las entradas son vectores.

3.2.27 `Rejection[v,b]`

Alias: Ninguno.

Descripción: Calcula el vector rechazado de un vector `v` sobre el espacio formado por el blade `b`.

Argumentos: `v` es un vector y `b` un r-blade.

Comentarios: Es auxiliada por las funciones `Gp`, `Oup` y `Mvi`.

3.2.28 `Rotation[v,w,x,theta]`

Alias: Ninguno.

Descripción: Calcula la rotación de `v`, por un ángulo `theta`, sobre el plano definido por $\mathtt{w} \wedge \mathtt{x}$.

Argumentos: `v,w,x` son vectores y `theta` es el ángulo de rotación en grados.

Comentarios: Se auxilia de las funciones `Ip` (si es necesario), `Oup`, `Gp`, `Magnitude`, `Turn` y `HQ`. Esta última, para saber si las entradas son vectores. De los parámetros, `theta` se puede omitir y el ángulo de rotación en este caso es el formado por los vectores `w` y `x`.

3.2.29 `ToBasis[v]`

Alias: `Tb`.

Descripción: Transforma un vector de la forma estándar (lista) a la forma que maneja el paquete (63).

Argumentos: `v` es un vector en forma de lista (forma estándar en Mathematica).

Comentarios: Ninguno.

3.2.30 ToVector[v,d]

Alias: Tv.

Descripción: Transforma el vector de la forma que maneja el paquete a la forma estándar.

Argumentos: v es un vector y d es un entero positivo.

Comentarios: Es auxiliada por la función HQ. El parámetro d se puede omitir, y en tal caso la dimensión de la lista resultante será la del valor más grande de los vectores canónicos.

3.2.31 Turn[m]

Alias: Ninguno.

Descripción: Calcula el reverso del multivector m.

Argumentos: m es un multivector.

Comentarios: Es auxiliada por la función Gp.

4 Aplicaciones

En este capítulo mostraremos la utilidad del álgebra de Clifford para cálculos que involucran cantidades algebraicas y geométricas. Como veremos, objetos matemáticos tales como vectores, números complejos y cuaternos son sólo casos particulares de multivectores. La exposición se desarrollará a través de aplicaciones simples a la cristalografía geométrica en dimensiones arbitrarias, que ha cobrado relevancia desde la aparición de materiales tales como las estructuras inconmensuradas y los cuasicristales que pueden interpretarse como estructuras periódicas en 4,5 ó 6 dimensiones [14]. Se mostrará también cómo el cálculo con números complejos y cuaternos puede hacerse dentro del mismo paquete.

Debemos mencionar que no es el objetivo de este capítulo revisar los conceptos básicos de la cristalografía así como el cálculo con cuaternos, de tal manera que asumiremos un conocimiento de estos campos.

4.1 Cristalografía en n dimensiones.

4.1.1 Redes en R^n

En esta sección daremos algunos conceptos relacionados con *redes*. Primero veremos los casos más sencillos que son en R^2 y en R^3 para luego definir a R^n.

Redes en R^2 R^2 denota el espacio euclidiano de 2 dimensiones. Sea $\{\mathbf{a}, \mathbf{b}\} \in R^2$ una base linealmente independiente. La Red L generada por la base $\{\mathbf{a}, \mathbf{b}\}$ está definida por

$$L = \left\{ P \in R^2 | P = h\mathbf{a} + k\mathbf{b}; h, k \in Z \right\}$$

donde Z es el conjunto de todos los enteros.

Redes en R^3 En el espacio euclidiano tridimensional, la Red L está generada por la base linealmente independiente $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\} \in R^3$ se define como

$$L = \left\{ P \in R^3 | P = h\mathbf{a} + k\mathbf{b} + l\mathbf{c}; h, k, l \in Z \right\}.$$

Redes en R^n Similarmente, en un espacio euclidiano n-dimensional R^n la Red L es generada por la base $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\} \in R^n$ es definida por

$$L = \left\{ P \in R^2 | P = \sum_{i=1}^{n} h_i \mathbf{a}_i;\ h_i \in Z \right\}$$

4.1.2 Red recíproca

Dada una base $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ siendo ésta única, se dice que tiene una base *recíproca* [12, 6]

$$\left\{ \mathbf{a}^1, \mathbf{a}^2, ..., \mathbf{a}^n \right\}, \tag{64}$$

(donde los valores "1,2,...,n" de (64) no son exponentes sino *índices superiores*), que es linealmente independiente y con la propiedad de

$$\mathbf{a}_i \cdot \mathbf{a}^j = \delta_i^j$$

donde la *delta de Kronecker*

$$\delta_i^j = \begin{cases} 1 & si \quad i = j \\ 0 & si \quad i \neq j, \end{cases}$$

para cualquier $0 \leq i, j \leq n$ [4].

La Red generada por la base (64) se llama *Red recíproca* de L y se denota por L^*. Usando técnicas de álgebra de Clifford, puede hallarse una fórmula explícita para encontrar la Red recíproca L^* dada una Red L en cualquier dimensión, de la siguiente manera [11, 5].

$$\begin{aligned} \mathbf{a}^j (\mathbf{a}_1 \wedge ... \wedge \mathbf{a}_n) &= \mathbf{a}^j \cdot (\mathbf{a}_1 \wedge ... \wedge \mathbf{a}_n) + \mathbf{a}^j \wedge (\mathbf{a}_1 \wedge ... \wedge \mathbf{a}_n) \\ &= \textstyle\sum_{i=1}^{n} (-1)^{i-1} (\mathbf{a}^j \cdot \mathbf{a}_i) (\mathbf{a}_1 \wedge ... \hat{\mathbf{a}}_i ... \wedge \mathbf{a}_n) \\ &= \textstyle\sum_{i=1}^{n} (-1)^{i-1} \delta_i^j (\mathbf{a}_1 \wedge ... \hat{\mathbf{a}}_i ... \wedge \mathbf{a}_n) \\ &= (-1)^{j-1} (\mathbf{a}_1 \wedge ... \hat{\mathbf{a}}_j ... \wedge \mathbf{a}_n), \end{aligned}$$

donde $\hat{a}_j$ quiere decir que el j-ésimo elemento del producto exterior se excluye. Por lo tanto

$$a^j = (-1)^{j-1}(a_1 \wedge ... \hat{a}_j ... \wedge a_n)(a_1 \wedge ... \wedge a_n)^{-1}$$

Ejemplo: Encontrar L^* de la Red L generada por la base $\{a_1, a_2, a_3, a_4\}$, donde $a_1 = (1,0,0,0)$, $a_2 = (0,1,0,0)$, $a_3 = (0,0,1,0)$ y $a_4 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ [3].

Solución:

```
In[1] := << "Clifford.m"

In[2] := a11 = e[1];
         a12 = e[2];
         a13 = e[3];
         a14 = 1/2e[1] + 1/2e[2] + 1/2e[3] + 1/2e[4];

In[6] := ia1 = Mvi[Op[a11, a12, a13, a14]]
Out[6] = 2e[1]e[2]e[3]e[4]

In[7] := a21 = Gp[Op[a12, a13, a14], ia1];
         a22 = -Gp[Op[a11, a13, a14], ia1];
         a23 = Gp[Op[a11, a12, a14], ia1];
         a24 = -Gp[Op[a11, a12, a13], ia1];

In[11] := {a21, a22, a23, a24}
Out[11] = {e[1] - e[4], e[2] - e[4], e[3] - e[4], 2e[4]}
```

Observe que para los resultados $a^1 = a21$, $a^2 = a22$, $a^3 = a23$ y $a^4 = a24$. En forma de vectores, la base recíproca sera:

```
In[12] := ToVector[a21]
Out[12] = {1, 0, 0, -1}

In[13] := ToVector[a22]
Out[13] = {0, 1, 0, -1}

In[14] := ToVector[a23]
Out[14] = {0, 0, 1, -1}
```

$$In[15] := \texttt{ToVector[a24]}$$
$$Out[15] = \{0, 0, 0, 2\}.$$

Esto es, $\mathbf{a}^1 = \{1, 0, 0, -1\}$, $\mathbf{a}^2 = \{0, 1, 0, -1\}$, $\mathbf{a}^3 = \{0, 0, 1, -1\}$ y $\mathbf{a}^4 = \{0, 0, 0, 2\}$. Puede verificarse fácilmente que $\mathbf{a}_i \cdot \mathbf{a}^j = \delta_i^j$.

4.1.3 Celdas unidad y primitiva

Consideremos una red L generada por la base $B = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$, el poliedro de n dimensiones (que, genericamente, se denomina paraleletopo) formado por B se denomina *celda unidad* de L. En el caso $n = 2$ la celda unidad es un paralelogramo y cuando $n = 3$ la celda unidad es un paralelepípedo.

Algunas veces es conveniente referir los vectores de la red a una base $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ tal que no todos los puntos de la red tienen coordenadas enteras sino coordenadas racionales. En este caso la celda es llamada *no primitiva*. En el caso usual cuando todos los vectores de la red tienen coordenadas enteras, la celda es llamada *primitiva*.

Dada una celda unidad generada por B, el volumen V de esta celda puede calcularse muy fácilmente con la ayuda del producto exterior, y esta dado por

$$V = |\mathbf{a}_1 \wedge \mathbf{a}_2 \wedge ... \wedge \mathbf{a}_n|.$$

Como un ejemplo, consideremos la red en 6 dimensiones generada por la base

$$\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_6\},$$

donde

$$\mathbf{a}_1 = (-1, -1, 0, 0, 0, 0),$$
$$\mathbf{a}_2 = (1, -1, 0, 0, 0, 0),$$
$$\mathbf{a}_3 = (0, 1, -1, 0, 0, 0),$$
$$\mathbf{a}_4 = (0, 0, 1, -1, 0, 0),$$
$$\mathbf{a}_5 = (0, 0, 0, 1, -1, 0),$$
$$\mathbf{a}_6 = (0, 0, 0, 0, 1, -1),$$

que generan una red cúbica de 6 dimensiones centrada en las caras [14]. El volumen

de la celda unidad de esta base es calculado como sigue:

```
In[16]:= a1 = -e[1] - e[2];
          a2 = e[1] - e[2];
          a3 = e[2] - e[3];
          a4 = e[3] - e[4];
          a5 = e[4] - e[5];
          a6 = e[5] - e[6];

In[22]:= Magnitude[Oup[a1, a2, a3, a4, a5, a6]]
Out[22]= 2
```

Con lo que obtenemos que el volumen de la celda unidad de una red cúbica de 6 dimensiones centrada en las caras es igual a 2.

4.1.4 Planos

La descripción de p-planos en n dimensiones se discutió en el capítulo 1. Mostraremos aquí cómo se implementan para la descripción de planos en redes de dimensión arbitraria [5].

Redes Planas e Hiperplanas El número de grados de libertad de un p-plano en un espacio n-dimensional está dado por

$$f = (n-p)(p+1)\,. \tag{65}$$

Los grados de libertad de un p-plano en un espacio n-dimensional que contiene a un r-plano son

$$f = (n-p)(p-r)\,.$$

Lineas en R^2 Una Red en R^2 contiene un 1-plano que pasa por dos puntos. Si los puntos de la Red por donde pasa el 1-plano son q y r entonces el 1-plano está dado por la ecuación

$$(\mathrm{p}-\mathrm{q}) \wedge (\mathrm{q}-\mathrm{r}) = 0 \tag{66}$$

donde p es un vector fijo, (66) se puede escribir de la forma

$$(\mathbf{p} - \mathbf{q}) \wedge \mathbf{v} = 0$$

donde el vector v tiene coordenadas enteras con respecto a la base de la Red.

Planos en R^3 Consideraremos el caso de un plano en R^3. Como vimos en el capítulo I, la ecuación de un plano puede escribirse como

$$(P - P_0) \wedge B = 0$$

donde B es un bivector de coordenadas enteras formadas por la base de la Red.

(n-1)-plano en R^n En forma análoga a los anteriores, se puede decir que el (n-1)-plano en R^n tiene las siguientes características:

1) La ecuación del hiperplano que pasa por la Red es

$$(P - P_0) \wedge M = 0$$

donde M es un (n-1)-vector que está definido por coordenadas enteras que corresponden a la base de la Red. P_0 es un punto que pertenece a la Red y por la cual pasa el hiperplano.

2) Puede describirse por las intersecciones con los ejes coordenados (índices de Miller), siempre y cuando el hiperplano no pase por el origen.

3) Los grados de libertad del hiperplano es

$$f = (n - p)(p + 1) = n$$

p-planos en R^n En el caso más general, para un p-plano ($p \neq n - 1$) en R^n, tiene las siguientes características:

1) Los grados de libertad del p-plano

$$f = (n - p)(p + 1) \neq n$$

2) La ecuación del p-plano es

$$(P - P_0) \wedge M = 0$$

donde P_0 es un punto de la Red y M es un p-vector que está formado por coordenadas enteras con respecto a la base de la Red.

4.2 Números complejos

En el capítulo I mencionamos que una base del espacio vectorial del álgebra de Clifford de R^2 es

$$\{1, e_1, e_2, e_1e_2\}$$

de manera que un multivector $A \in C(R^2)$ se escribe como

$$A = k_0 + k_1e_1 + k_2e_2 + k_3e_1e_2,$$

donde $k_i \in R$. Podemos descomponer A en la suma de dos multivectores: $A = A_+ + A_-$, donde A_+ contiene los elementos de grado par y A_- los de grado impar. Es decir,

$$A_+ = k_0 + k_3e_1e_2$$
$$A_- = k_1e_1 + k_2e_2.$$

Los elementos de grado par A_+ forman a su vez un álgebra, por lo que a este conjunto se le denomina *subálgebra par* de $C(R^2)$, y se denota por $C_+(R^2)$. Si definimos

$$i = e_1e_2,$$

entonces, un elemento de $C_+(R^2)$ se puede escribir como

$$A_+ = z = k_0 + k_3i.$$

Observe que $i^2 = (e_1e_2)(e_1e_2) = -(e_1e_2e_2e_1) = -1$ e i es por si mismo un generador de rotación. Con esto, vemos que la subálgebra par de $C(R^2)$ es equivalente al álgebra de los *números complejos.*

Veamos como el paquete permite la manipulación de números complejos.

Ejemplo: Primero hacemos la identificación $i = e_1e_2$

```
In[23]:= trans[x_] := x /. i_ -> e[1]e[2];
```

Como simple verificación, calcularemos $\mathbf{i}^2$

$$In[24] := \texttt{Gp[trans[i], trans[i]]}$$
$$Out[24] = -1$$

Ahora, definamos dos números complejos v y w

$$In[25] := \texttt{v = a + b i;}$$
$$In[26] := \texttt{w = c + d i;}$$

al realizar el producto entre ellos, el resultado quedará en términos de $\mathbf{e_1e_2}$, así que el resultado lo tenemos que modificar para que quede en términos de i

$$In[27] := \texttt{Gp[trans[v], trans[w]] /. e[1]e[2]- > i}$$
$$Out[27] = \texttt{a c - b d + b c i + a d i}$$

Por último encontraremos el conjugado de v

$$In[28] := \texttt{Turn[trans[v]] /. e[1]e[2]- > i}$$
$$Out[28] = \texttt{a - b i}$$

4.3 Cuaternos

Retomando el procedimiento seguido en el caso de los números complejos, desarrollaremos ahora el álgebra de Clifford de R^3 denotada por $C(R^3)$. El álgebra de Clifford $C(R^3)$ es un espacio vectorial de dimensión 8 y tiene como base

$$\{1, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, \mathbf{e}_2\mathbf{e}_3, \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\}.$$

Un multivector $A \in C(R^3)$ se escribe como

$$A = k_0 + k_1\mathbf{e}_1 + k_2\mathbf{e}_2 + k_3\mathbf{e}_3 + k_4\mathbf{e}_1\mathbf{e}_2 + k_5\mathbf{e}_1\mathbf{e}_3 + k_6\mathbf{e}_2\mathbf{e}_3 + k_7\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3,$$

el cual, podemos expresarlo como $A = A_+ + A_-$, donde

$$A_+ = k_0 + k_4\mathbf{e}_1\mathbf{e}_2 + k_5\mathbf{e}_1\mathbf{e}_3 + k_6\mathbf{e}_2\mathbf{e}_3$$
$$A_- = k_1\mathbf{e}_1 + k_2\mathbf{e}_2 + k_3\mathbf{e}_3 + k_7\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3.$$

Los elementos de grado par A_+ forman una subálgebra $C_+(R^3)$ de $C(R^3)$, equivalente al álgebra de los cuaternos. Esto puede verse con facilidad si hacemos las siguientes identidades [11]

$$\begin{aligned} \mathbf{i} &= -\mathbf{e}_2\mathbf{e}_3, \\ \mathbf{j} &= \mathbf{e}_1\mathbf{e}_3, \\ \mathbf{k} &= -\mathbf{e}_1\mathbf{e}_2, \end{aligned} \tag{67}$$

las cuales permiten las siguientes propiedades [1]

$$\begin{aligned} &\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1, \\ &\mathbf{ij} = \mathbf{k},\ \mathbf{ki} = \mathbf{j},\ \mathbf{jk} = \mathbf{i}, \\ &\mathbf{ijk} = -1, \end{aligned} \tag{68}$$

que son las famosas relaciones de Hamilton. Con las identidades dadas en (67), un elemento de $C_+(R^3)$ puede escribirse como

$$Q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}.$$

Debido a las propiedades (68), Q es un cuaterno.

Ejemplo: El paquete tiene funciones definidas para los cuaternos. Primero veremos las propiedades de **i**, **j** y **k**

```
In[29] := Qp[i, i] == Qp[j, j] == Qp[k, k] == -1
Out[29] := True
```

```
In[30] := Qp[i, j, k]
Out[30] = -1
```

```
In[31] := Qp[i, j]
Out[31] = k
```

```
In[32] := Qp[k, i]
Out[32] = j
```

```
In[33] := Qp[j, k]
Out[33] = i
```

Ahora definimos un cuaterno

$$In[34] := \mathtt{q = a + b\,i + c\,j + d\,k;}$$

encontramos su inverso

$$In[35] := \mathtt{Qi[q]}$$
$$Out[35] = \frac{\mathtt{a - b\,i - c\,j - d\,k}}{\mathtt{a^2+b^2+c^2+d^2}}$$

la magnitud del cuaterno

$$In[36] := \mathtt{Qm[q]}$$
$$Out[36] = \mathtt{Sqrt[a^2+b^2+c^2+d^2]}$$

el conjugado

$$In[37] := \mathtt{Qc[q]}$$
$$Out[37] = \mathtt{a - b\,i - c\,j - d\,k}$$

extraemos la parte real

$$In[38] := \mathtt{Re[q]}$$
$$Out[38] = \mathtt{a}$$

y por último, la parte imaginaria

$$In[39] := \mathtt{Im[q]}$$
$$Out[39] = \mathtt{\{b, c, d\}}$$

4.4 Rotaciones

Este tema se vió ampliamente en el capítulo I, en esta sección veremos cómo utilizar el paquete para el cálculo de una rotación. La rotación de un vector $\mathbf{x}$ sobre un plano caracterizado por un bivector $\mathbf{u} \wedge \mathbf{v}$, esta dada por

$$\mathrm{R}(\mathbf{x}) = \mathrm{R}^{\dagger}\mathbf{x}\mathrm{R},$$

donde

$$\mathrm{R} = \cos(\theta/2) + \frac{\mathbf{u} \wedge \mathbf{v}}{|\mathbf{u} \wedge \mathbf{v}|} \sin(\theta/2),$$

y θ es igual al doble del ángulo formado por los vectores $\mathbf{u}$ y $\mathbf{v}$.

Ejemplo: Encontrar la rotación del vector $\mathbf{x} = (1, 1, 1)$ en el plano formado por los vectores $\mathbf{u} = \mathbf{e}_1$ y $\mathbf{v} = \mathbf{e}_2$

$$In[40] := \mathtt{x = e[1] + e[2] + e[3];}$$

Como los vectores u y v forman un ángulo de 90°, entonces

$$In[41] := \mathtt{theta = 90 * Pi/180;}$$
$$In[42] := \mathtt{plano = Oup[e[1], e[2]]/Magnitude[Oup[e[1], e[2]]];}$$
$$In[43] := \mathtt{r = Cos[theta/2] + plano * Sin[theta/2];}$$

aplicando el producto geométrico adecuado tenemos

$$In[44] := \mathtt{Gp[Turn[r], x, r]}$$
$$Out[44] = -\mathtt{e[1] + e[2] + e[3]}$$

Estos pasos son los que sigue la función implícita `Rotation`, teniendo la siguiente sintaxis

$$In[45] := \mathtt{Rotation[x, e[1], e[2]]}$$
$$Out[45] = -\mathtt{e[1] + e[2] + e[3]}$$

Ahora, si rotamos el mismo vector 180° sobre el mismo plano.

$$In[46] := \mathtt{Rotation[x, e[1], e[2], 180]}$$
$$Out[46] = -\mathtt{e[1] - e[2] + e[3]}$$

Conclusiones

En este trabajo se describe un paquete construido para el sistema *Mathematica* que consta de 30 funciones implícitas, que permiten realizar operaciones (numéricas y simbólicas) en un álgebra de Clifford de un espacio vectorial de dimensión arbitraria (finita). Se incluye (Capítulo 3) una descripción de todas las funciones implícitas y (Capítulo 4) aplicaciones a vectores, números complejos, cuaternos y algunas operaciones geométricas, las cuales sirven de ejemplo para la utilización del paquete. El paquete, por obvias razones, fué llamado **Clifford**.

Dada la importancia que el álgebra de Clifford está tomando recientemente como un lenguaje natural para la Física, y la versatilidad del sistema *Mathematica*, el paquete puede resultar una herramienta muy útil y fácil de usar. Además, está estructurado de tal manera que permite su ampliación y perfeccionamiento.

En cuanto a la eficiencia del programa podemos mencionar que en un reporte que estudiaba los diversos sistemas que existen en el mercado para matemática computacional [19] se presentaba un problema para resolverlo con álgebra de Clifford. Se incluyen los programas que solucionan el problema en los diversos sistemas, incluyendo en *Mathematica*. En una computadora personal 486 a 66 Mhz, el problema se resuelve en 0.36s. Si resolvemos el problema usando **Clifford** el tiempo de cómputo es de 8.89s. La comparación no puede ser directa. El programa presentado en la referencia [19] está diseñado para resolver un problema específico y no pueden obtenerse a partir de él todas las operaciones de un álgebra de Clifford. Nuestro paquete sin embargo, es un programa general, con él podemos resolver una gran cantidad de problemas relacionados con esta álgebra. Esta generalidad, lleva consigo un aumento en el tiempo de cómputo.

Por otro lado, en la referencia [16] se describe un programa comercial, llamado **Clical**. Las diferencias con el nuestro son varias. Ambas requieren de un sistema coordenado, pero **Clical** utiliza una representación matricial de los multivectores, apoyándose en un isomorfismo para representar los vectores canónicos en matrices. Por otro lado, **Clical** realiza únicamente cálculos numéricos. En el caso de **Clifford**,

no se requiere de una representación matricial y los cálculos pueden hacerse, ademas, simbólicamente. Ambos incluyen funciones asociadas a multivectores, las cuales se pueden resolver con series de Taylor. El programa **Clical** esta, por otro lado, enfocado a problemas de la ecuación de Dirac, Electromagnetismo y Relatividad especial, lo cual no está implementado en nuestro paquete, aunque las bases están establecidas y con una buena manipulación de **Clifford** se pueden establecer.

Debemos mencionar que el paquete fué diseñado teniendo en mente las aplicaciones del álgebra de Clifford a la Cristalografía geométrica, lo que implica que hay un sistema coordenado intrínseco. Esto puede representar una limitación puesto que la formulación más general del álgebra de Clifford no requiere un sistema coordenado. Sin embargo, y como se mostró en este trabajo, hay muchas aplicaciones para el paquete en su estado actual. El trabajo aquí realizado deja abierta la posibilidad de mejorarlo y ampliarlo con el objetivo principal de articular el cálculo con diversos objetos matemáticos en un sólo algoritmo.

REFERENCIAS

1. **Aleksandrov, A. D., A. N. Kolmogorov, M. A. Laurentiev y otros**; *La matemática: su contenido, métodos y significado*; Vol.3; Alianza Editorial (Madrid, 1985).

2. **Baylis, W. E., Huschilt y Jiansu Wei**; *Why i?*; Am. J. Phys., 60 (1992) 788-792.

3. **Conway, J. H. y N. J. A. Sloane**; *Sphere Packings, Lattices and Groups*; Springer-Verlag (New York, 1988).

4. **Coxeter, H. S. M.**; *Fundamentos de Geometría*; LIMUSA (México, 1988).

5. **Dávila, F., A. Gómez y J. L. Aragón**; *Notas sobre Algebra de Clifford*; Trabajo sin publicar.

6. **Engel, P.**; *Geometric Cristallography*; D. Reidel Publishin Company (The Netherlands, 1986).

7. **Garrett, A.**; *An advertisement for Clifford algebra*; Physics Word (Septiembre del992); 13-14.

8. **Gaylord, R. J., S. N. Kaming y P. R. Wellin**; *Introduction to Programing with Mathematica*; Springer-Verlag (New York, 1993).

9. **Grossmann, S. I.**; *Algebra Lineal con Aplicaciones*; McGraw Hill (México, 1992).

10. **Gull, S., A. Lasenby y C. Doran**; *Imaginary Numbers Are Not Real - The Geometric Algebra of Spacetime*; Foundations of Physics, **23** (1993) 1175-1189.

11. **Hestenes, D.**; *New Fundation For Classical Machanics*; D. Reidel Publishing Company (The Netherlands, 1987).

12. **Hestenes, D. y G. Sobczyk**; *Clifford Algebra to Geometric Calculus*; D. Reidel Publishing Company (The Netherlands, 1985).

13. **Jancewicz, B.**; *Multivector and Clifford Algebra in Electrodynamics*; World Scientific (Singapore, 1988).

14. **Janot, C.**; *Quasicrystals*; Oxford Science Publications (U.K., 1992).

15. **Lang, S.**; *Introducción al Algebra Lineal*; Addison-Wesley Iberoamericana (Wilmington Delaware, 1990).

16. **Lounesto, P.**; *Clifford Algebra Calculation with a Microcomputer*; en "Clifford Algebras and their Applications in Mathematical Physics", A. Micali. et al. (Editores), Kluwer Academic Publishers (The Netherlands, 1992), pp. 39-55.

17. **Maeder, R.**; *Programing in Mathematica*; Addison-Wesley Publishing Company (Redwood City, 1991), segunda edición.

18. **Nguyen, D. B.**; *Plücker's relations and the electromagnetic field*; Am. J. Phys., **60** (1992) 1145-1147.

19. **Simon, B.**; *Comparative CAS Reviews. Simbolic Math: Problems and Solutions*; Notices of the American Mathematical Society, **39** (1992) 701-706.

20. **Strang, G.**; *Algebra Lineal y sus Aplicaciones*; Addison-Wesley (Wilmington Delaware, 1986).

21. **Vold, T. G.**; *An Introduction to the Geometric Algebra with an Application in Rigid Body Mechanics*; Am. J. Phys., **61** (1993) 491-500.

22. **Wolfram, S.**; *Mathematica. A System for Doing Mathematics by Computer*; Addison-Wesley Publishing Company (Redwood City, 1991), second edition.

Apéndice A

En este apéndice se enlista el paquete **Clifford**, desarrollado en esta tesis, para la versión 2.0 de *Mathematica*.

```
(* Set up the Package Context. *)

(* :Title: Clifford Algebra of a Vector Space. *)

(* :Summary:
        This file contains declarations for calculations with Clifford
        algebra of a n-dimensional vector space. The Euclidean inner
        product is considered to construct the algebra. When loaded,
        vectors (and multivectors) must be given as linear combinations
        of a canonical (orthonormal basis) that must be denoted by
        e[1],e[2],..,e[n].
      Example: The vector {1,2,0,-1} should be written as
        e[1] + 2 e[2] - e[4].
        With the exception of the function Dual[m,n], it is not
        neccesary to define the dimension of the vector space, this
        is calculated automatically by the function maxdim[ ].  *)

(* :Author: Oscar G. Caballero. *)

(* :References: 1. D. Hestenes, 1987. New Foundations for Classical Mechanics.
        D. Reidel Publishing Co. Holland
              2. S. Gull, A. Lasenby and C. Doran, 1993. Imaginary Numbers
        are not Real- The Geometric Algebra of Spacetime. Foundations of
        Physics, Vol. 23, No. 9: 1175-1201. *)


BeginPackage["Clifford`"]
```

```
(* Usage message for the exported function and the Context itself *)

Clifford::usage = "Clifford.m is a package to resolve operations with
Clifford Algebra."

GeometricProduct::usage = "GeometricProduct[m1,m2,...] calculates the Geometric
Product of multivectors m1,m2,... Alias = Gp."

Grade::usage = "Grade[m,r] gives the r-vector part of the multivector m."

HomogeneousQ::usage = "HomogeneousQ[x,r] gives True if x is a r-blade and False
otherwise. Alias = HQ."

Turn::usage = "Turn[m] gives the Reverse of the multivector m."

Magnitude::usage = "Magnitude[m] calculates the Magnitude of the multivector m."

Dual::usage = "Dual[m,n] calculates the Dual of the multivector m in a
n-dimensional space."

InnerProduct::usage = "InnerProduct[m1,m2,...] calculates the Inner Product of
multivectors m1,m2,... Alias = Ip."

OuterProduct::usage = "OuterProduct[m1,m2,...] calculates the Outer Product of
multivectors m1,m2,... Alias = Oup."

Rotation::usage = "Rotation[v,w,x,theta] Rotates the vector v by an angle theta
(in degrees), along the plane defined by w and x. The sense of the rotation
is from w to x. Default value of theta is the  angle between w and x."

MultivectorInverse::usage = "MultivectorInverse[m] gives the inverse of a
multivector m if this exist. Alias = Mvi."

Reflection::usage = "Reflection[v,w,x] reflects the vector v by the plane
formed by the vectors w and x."
```

Projection::usage = "Projection[v,w] calculate the Projection of the vector v on the subspace defined by the r-blade w."

Rejection::usage = "Rejection[v,w] calculate the Rejection of the vector v on the subspace defined by the r-blade w."

ToBasis::usage = "ToBasis[x] Transform the vector x from {a,b,...} to the standar form used in this Package: ae[1]+be[2]+...."

ToVector::usage ="ToVector[x,n] transform the n-dimensional vector x from ae[1]+be[2]+... to the standar Mathematica form {a,b,...}. The defaul value of n is the highest of all e[i]'s."

CrossProduct::usage = "CrossProduct[u,v] gives the Cross Product of the three-dimensional vectors u and v. Alias = Cp."

DotProduct::usage = "DotProduct[u,v] gives the Inner Product of vectors u and v. Alias = Dp."

QuaternionProduct::usage = "QuaternionProduct[q1,q2,...] gives the product of quaternions q1,q2,... Alias = Qp."

QuaternionInverse::usage = "QuaternionInverse[q] finds the inverse of a quaternion q. Alias = Qi."

QuaternionMagnitude::usage = "QuaternionMagnitude[q] gives the magnitude of a quaternion q. Alias = Qm."

QuaternionConjugate::usage = "QuaternionConjugate[q] gives the conjugated of a quaternion q. Alias = Qc."

Pseudoescalar::usage = "Pseudoscalar[n] gives the n-dimensional pseudoscalar."

GeometricPower::usage = "GeometricPower[m,n] calculates the Geometic Product of a multivector m, n-times. Alias = GPower."

```
GeometricProductSeries::usage = "GeometricProductSeries[sym,m,n] calculates the
series of the function sym, of a multivector m up to a power n. Default value of
n is 10. Alias = GSeries."

GeometricExp::usage = "GeometricExp[m,n] calculates the series of the function
Exp, of a multivector m up to a power n. Default value of n is 10.
Alias = GExp."

GeometricSin::usage = "GeometricSin[m,n] calculates the series of the function
Sin, of a multivector m up to a power n. Default value of n is 10.
Alias = GSin."

GeometricCos::usage = "GeometricCos[m,n] calculates the series of the function
Cos, of a multivector m up to a power n. Default value of n is 10.
Alias = GCos."

GeometricTan::usage = "GeometricTan[m,n] calculates the series of the function
Tan, of a multivector m up to a power n. Default value of n is 10.
Alias = GTan."


(* Alias Section *)

$PreRead = StringReplace[#,{"Gp"->"GeometricProduct",
        "Ip"->"InnerProduct",
        "Oup"->"OuterProduct",
        "Mvi"->"MultivectorInverse",
        "Tb"->"ToBasis",
        "Tv"->"ToVector",
        "HQ"->"HomogeneousQ",
        "Cp"->"CrossProduct",
        "Dp"->"DotProduct",
        "Qp"->"QuaternionProduct",
        "Qi"->"QuaternionInverse",
        "Qm"->"QuaternionMagnitude",
        "Qc"->"QuaternionConjugate",
```

```
        "GSeries"->"GeometricProductSeries",
        "GPower"->"GeometricPower",
        "GExp"->"GeometricExp",
        "GSin"->"GeometricSin",
        "GCos"->"GeometricCos",
        "GTan"->"GeometricTan"}]&


Begin["`Private`"]  (* Begin the Private Context *)


(* Unprotect functions Re and Im to define our rules *)
protected = Unprotect [Re, Im]


(* Error Messages *)
Clifford::messagevectors = "`1` function works only with vectors."
Clifford::messageblade = "`1` has no inverse."
Clifford::messagedim = "Three must be the maximum dimension."


(* Begin Geometric Product Section *)
GeometricProduct[m1_,m2_,m3__] := GeometricProduct[tmp[m1,m2],m3] /.
        tmp->GeometricProduct
GeometricProduct[m1_,m2_] := geoprod[Expand[m1],Expand[m2]] // Expand
geoprod[a_,y_] := a y /; FreeQ[a,Global`e[_?Positive]]
geoprod[x_,a_] := a x /; FreeQ[a,Global`e[_?Positive]]
geoprod[x_,y_] := Module[{p1=ntuple[x,Max[maxdim[x],maxdim[y]]],
        m,n,q=1,s,r={},p2=ntuple[y,Max[maxdim[x],maxdim[y]]]},
        s=Sum[p2[[m]]*p1[[n]],{m,Length[p1]-1},{n,m+1,Length[p2]}];
        r=Mod[p1+p2,2];
        Do[ If[r[[i]]==1,q *= Global`e[i]],{i,Length[r]} ];
        (-1)^s*q ]
geoprod[a_ x_,y_] := a geoprod[x,y] /; FreeQ[a,Global`e[_?Positive]]
geoprod[x_,a_ y_] := a geoprod[x,y] /; FreeQ[a,Global`e[_?Positive]]
```

```
geoprod[a_ x_,b_ y_] := a b geoprod[x,y] /;
        FreeQ[a,Global`e[_?Positive]] && FreeQ[b,Global`e[_?Positive]]
geoprod[x_Plus,y_Plus] := Distribute[tmp[x,y],Plus] /. tmp->geoprod
geoprod[x_,y_Plus] := Distribute[tmp[x,y],Plus] /. tmp->geoprod
geoprod[x_Plus,y_] := Distribute[tmp[x,y],Plus] /. tmp->geoprod
(* End of Geometric Product Section *)


(* Begin Grade Section *)
Grade[m_,r_?NumberQ] := gradehelp[Expand[m],r]
grados[a_] := 0 /; FreeQ[a,Global`e[_?Positive]]
grados[x_] := Plus @@ ntuple[x,Max[maxdim[x]]]
grados[a_ x_] := grados[x] /; FreeQ[a,Global`e[_?Positive]]
gradehelp[x_Plus,n_] := Distribute[tmp[x,n],Plus] /. tmp->gradehelp
gradehelp[x_,n_] := If[grados[x]==n,x,0]
(* End of Grade Section *)


(* Begin Inner Product Section *)
InnerProduct[m1_,m2_,m3__] := InnerProduct[tmp[m1,m2],m3] /. tmp->InnerProduct
InnerProduct[m1_,m2_] := innprod[Expand[m1],Expand[m2]] // Expand
innprod[a_,y_] := 0 /; FreeQ[a,Global`e[_?Positive]]
innprod[x_,a_] := 0 /; FreeQ[a,Global`e[_?Positive]]
innprod[x_,y_] := Module[{p=Plus @@ ntuple[x,Max[maxdim[x],maxdim[y]]],
        q=Plus @@ ntuple[y,Max[maxdim[x],maxdim[y]]]},
        Grade[GeometricProduct[x,y],Abs[p-q]] ]
innprod[a_ x_,y_] := a innprod[x,y] /; FreeQ[a,Global`e[_?Positive]]
innprod[x_,a_ y_] := a innprod[x,y] /; FreeQ[a,Global`e[_?Positive]]
innprod[a_ x_,b_ y_] := a b innprod[x,y] /;
        FreeQ[a,Global`e[_?Positive]] && FreeQ[b,Global`e[_?Positive]]
innprod[x_Plus,y_Plus] := Distribute[tmp[x,y],Plus] /. tmp->innprod
innprod[x_,y_Plus] := Distribute[tmp[x,y],Plus] /. tmp->innprod
innprod[x_Plus,y_] := Distribute[tmp[x,y],Plus] /. tmp->innprod
(* End of Inner Product Section *)
```

```
(* Begin Outer Product Section *)
OuterProduct[m1_,m2_,m3__] := OuterProduct[tmp[m1,m2],m3] /. tmp->OuterProduct
OuterProduct[m1_,m2_] := outprod[Expand[m1],Expand[m2]] // Expand
outprod[a_,y_] := a y /; FreeQ[a,Global`e[_?Positive]]
outprod[x_,a_] := a x /; FreeQ[a,Global`e[_?Positive]]
outprod[x_,y_] := Module[{p=Plus @@ ntuple[x,Max[maxdim[x],maxdim[y]]],
      q=Plus @@ ntuple[y,Max[maxdim[x],maxdim[y]]]},
      Grade[GeometricProduct[x,y],p+q] ]
outprod[a_ x_,y_] := a outprod[x,y] /; FreeQ[a,Global`e[_?Positive]]
outprod[x_,a_ y_] := a outprod[x,y] /; FreeQ[a,Global`e[_?Positive]]
outprod[a_ x_,b_ y_] := a b outprod[x,y] /;
      FreeQ[a,Global`e[_?Positive]] && FreeQ[b,Global`e[_?Positive]]
outprod[x_Plus,y_Plus] := Distribute[tmp[x,y],Plus] /. tmp->outprod
outprod[x_,y_Plus] := Distribute[tmp[x,y],Plus] /. tmp->outprod
outprod[x_Plus,y_] := Distribute[tmp[x,y],Plus] /. tmp->outprod
(* End of Outer Product Section *)


(* Begin Turn Section *)
Turn[m_] := backside[Expand[m]]
backside[a_] := a /; FreeQ[a,Global`e[_?Positive]]
backside[x_] := x /; Length[x]==1
backside[x_] := GeometricProduct @@ Global`e/@Reverse[maxdim[x]]
backside[a_ x_] := a backside[x] /; FreeQ[a,Global`e[_?Positive]]
backside[x_Plus] := Distribute[tmp[x],Plus] /. tmp->backside
(* End of Turn Section *)


(* Pseudoescalar function *)
Pseudoscalar[x_?Positive] := Apply[Times,Global`e/@Table[i,{i,x}]]


(* HomogeneousQ function *)
HomogeneousQ[x_,r_?NumberQ] := SameQ[Expand[x],Grade[x,r]]
```

```
(*  Magnitude function *)
Magnitude[v_]:= Sqrt[Grade[GeometricProduct[v,Turn[v]],0]]


(* Dual function *)
Dual[v_,x_?Positive] := GeometricProduct[v,Turn[Pseudoscalar[x]]]


(* Begin Rotation function *)
Rotation[v_,w_,x_,angle_:Automatic] := Module[{bivector,r,theta=angle*Pi/180},
      If[(!HomogeneousQ[v,1]) || (!HomogeneousQ[w,1]) || (!HomogeneousQ[x,1]),
        Message[Clifford::messagevectors,Rotation]; $Failed,
      bivector=OuterProduct[w,x];
        If[angle === Automatic,
          theta=InnerProduct[w,x]/(Magnitude[w]*Magnitude[x]);
        r=Sqrt[(1+theta)/2]+Sqrt[(1-theta)/2]*bivector/Magnitude[bivector],
        r=Cos[theta/2]+Sin[theta/2]*bivector/Magnitude[bivector]];
        GeometricProduct[Turn[r],v,r]  ] ]
(* End of Rotation *)


(* Begin MultivectorInverse function *)
MultivectorInverse[v_] := If[
      GeometricProduct[v,Turn[v]]==Grade[GeometricProduct[v,Turn[v]],0],
                  Turn[v] / Magnitude[v]^2,
                  Message[Clifford::messageblade,v]; $Failed ]
(* End of MultivectorInverse *)


(* Begin Reflection function *)
Reflection[v_,w_,x_] := Module[{u},
      If[(!HomogeneousQ[v,1]) || (!HomogeneousQ[w,1]) || (!HomogeneousQ[x,1]),
        Message[Clifford::messagevectors,Reflection]; $Failed,
        u=Dual[OuterProduct[w,x],3];
        GeometricProduct[-u,v,u] ] ]
(* End of Reflection *)
```

```
(* Projection function *)
Projection[v_,w_] := GeometricProduct[InnerProduct[v,w],MultivectorInverse[w]]


(* Rejection function *)
Rejection[v_,w_] := GeometricProduct[OuterProduct[v,w],MultivectorInverse[w]]


(* ToBasis function *)
ToBasis[x_?VectorQ] := Sum[x[[i]]*Global`e[i],{i,Length[x]}]


(* Begin  ToVector function *)
ToVector[x_,d_:Automatic] := Module[{dim=d,aux,v=Expand[x],list={}},
      aux=Flatten[maxdim[v]];
      If[HomogeneousQ[v,1],
            If[d === Automatic, dim=Max[aux]];
            list=Table[0,{dim}];
            If[Length[aux]==1, list[[ aux[[1]] ]]=coef[v],
              Do[list[[ aux[[i]] ]]=coef[v[[i]]],
                  {i,Length[aux]}] ];
            list,
        Message[Clifford::messagevectors,ToVector]; $Failed ] ]
(* End of ToVector *)


(* Re function *)
Re[m_] := Module[{m1=transform[Expand[m]]},Grade[m1,0]]


(* Begin CrossProduct function *)
CrossProduct[u_,v_] :=
  Module[{aux=maxdim[Expand[u]],aux1=maxdim[Expand[v]]},
      If[(Max[aux]<4) && (Max[aux1]<4),
            If[HomogeneousQ[u,1] && HomogeneousQ[v,1],
                  Dual[OuterProduct[u,v],3],
```

```
                    Message[Clifford::messagevectors,CrossProduct];$Failed],
            Message[Clifford::messagedim]; $Failed ] ]
(* End of CrossProduct *)


(* DotProduct function *)
DotProduct[u_,v_] := If[HomogeneousQ[u,1] && HomogeneousQ[v,1],
                    InnerProduct[u,v],
                    Message[Clifford::messagevectors,DotProduct]; $Failed ]


(* Begin Im function *)
Im[x_] := Module[{l={0,0,0},v=transform[Expand[x]]},
      l[[3]]=-Coefficient[v,Global`e[1]Global`e[2]];
      l[[2]]=Coefficient[v,Global`e[1]Global`e[3]];
      l[[1]]=-Coefficient[v,Global`e[2]Global`e[3]];
      l ]
(* End of Im *)


(* Begin QuaternionProduct function *)
QuaternionProduct[x:_..] := Module[{aux={x},q=0},
      Do[ p[i]=transform[Expand[aux[[i]]]], {i,Length[aux]} ];
      q=p[1];
      Do[ q=GeometricProduct[q,p[i+1]], {i,Length[aux]-1} ];
      untransform[q] ]
(* End of QuaternionProduct *)


(* QuaternionInverse function *)
QuaternionInverse[q_] := untransform[MultivectorInverse[transform[Expand[q]]]]


(* QuaternionMagnitude function *)
QuaternionMagnitude[q_] := untransform[Magnitude[transform[Expand[q]]]]
```

```
(* QuaternionConjugate function *)
QuaternionConjugate[q_] := untransform[Turn[transform[Expand[q]]]]


(* Begin Geometric Power Section *)
GeometricPower[m_,n_Integer] := MultivectorInverse[GeometricPower[m,-n]] /;
                                n < 0
GeometricPower[m_,0] := 1
GeometricPower[m_,n_Integer] := GeometricProduct[GeometricPower[m,n-1],m] /;
                                n >= 1
(* End of Geometric Power *)


(* Geometric Exp function *)
GeometricExp[m_,n_:10] := GeometricProductSeries[Exp,m,n]


(* Geometric Sin function *)
GeometricSin[m_,n_:10] := GeometricProductSeries[Sin,m,n]


(* Geometric Cos function *)
GeometricCos[m_,n_:10] := GeometricProductSeries[Cos,m,n]


(* Geometric Tan function *)
GeometricTan[m_,n_:10] := GeometricProductSeries[Tan,m,n]


(* Begin Geometric Product Series function *)
GeometricProductSeries[sym_Symbol,m_,n_:10] := Module[
        {s=Series[sym[x],{x,0,n}],res=0,a=1},
        Do[If[i != 0, a=GeometricProduct[a,m]];
          res += Coefficient[s,x,i]*a, {i,0,n}];
        res      ] /; IntegerQ[n] && Positive[n]
(* End of Geometric Product Series *)
```

```
(* Begin maxdim Section *)
maxdim[a_] := {0} /; FreeQ[a,Global`e[_?Positive]]
maxdim[x_] := Module[{l={}},If[Length[x]==1,l={x[[1]]},
      Do[ l=Append[l,x[[i,1]]], {i,Length[x]} ] ];
      l ]
maxdim[a_ x_] := maxdim[x] /; FreeQ[a,Global`e[_?Positive]]
maxdim[x_Plus] := Module[{l={}},
      Do[ l=Append[l,maxdim[ x[[i]] ] ], {i,Length[x]} ];
      l ]
(* End of maxdim Section *)


(* Begin ntuple function *)
ntuple[x_,dim_] := Module[{bin=Table[0,{dim}]},
      If[Length[x]==1,bin[[ x[[1]] ]]=1,
        Do[ bin[[ x[[i,1]] ]]=1, {i,Length[x]} ] ];
      bin ]
(* End of ntuple *)


(* transform function *)
transform[x_] := x //. {Global`i -> -Global`e[2]Global`e[3],
                        Global`j -> Global`e[1]Global`e[3],
                        Global`k -> -Global`e[1]Global`e[2]}


(* untransform function *)
untransform[x_] := x //. {Global`e[2]Global`e[3] -> -Global`i,
                          Global`e[1]Global`e[3] -> Global`j,
                          Global`e[1]Global`e[2] -> -Global`k}


(* Begin coef Section *)
coef[a_ x_] := a coef[x] /; FreeQ[a,Global`e[_?Positive]]
coef[x_] := 1 /; Magnitude[x]==1
(* End of coef Section *)
```

```
Protect[Evaluate[protected]] (* Restore protection of the functions *)


End[] (* End the Private Context *)


(* Protect exported symbols *)

Protect[ GeometricProduct, Grade, Turn, Magnitude, Dual, InnerProduct,
        OuterProduct, Rotation, MultivectorInverse, Reflection, HomogeneousQ,
        Projection, Rejection, ToBasis, ToVector, CrossProduct, DotProduct,
        QuaternionProduct, QuaternionInverse, QuaternionMagnitude,
        QuaternionConjugate, GeometricPower, GeometricProductSeries,
        GeometrucExp, GeometricSin, GeometricCos, GeometricTan ]


EndPackage[] (* End the Package Context *)
```

Apéndice B

Clifford Algebra

O. Caballero and J.L. Aragón*

Instituto de Física, Universidad Nacional Autónoma de México
Apartado Postal 20-364
01000 México, Distrito Federal, México

e-mail: **aragon@ifunam.ifisicacu.unam.mx**

Abstract

The Clifford algebra of a n-dimensional vector space is introduced. It provides a general language comprising vectors, complex numbers and quaternions. Grassmann algebra is also naturally contained in a Clifford algebra. The exposition contains the basic ideas and describes a package which can be a powerful tool since allows the manipulation of all these mathematical objects.

* Author to whom correspondence must be addressed.

Introduction

The set of vectors in R^n with the standar operations of addition of vectors and multiplication for real numbers is a *vector space*, over the field of real numbers, which we denote as V^n. It means that the addition of n-tuples of the form $(x_1,x_2,...,x_n)$, and multiplication by real numbers satisfy certain properties which are those of a vector space [Lang 1969]. The canonical basis of V^n is the set of n-dimensional vectors $\{\mathbf{e}_1,\mathbf{e}_2,...,\mathbf{e}_n\}$ where $<\mathbf{e}_i,\mathbf{e}_j>=\delta_{ij}$ and $<.>$ is a inner product in V^n. An element $\mathbf{v}$ of V^n is written as a linear combination of this canonical basis:

$$\mathbf{v}=x_1\mathbf{e}_1+x_2\mathbf{e}_2+\cdots+x_n\mathbf{e}_n.$$

It is said therefore that the n-tuple $(x_1,x_2,...,x_n)$ is the coordinate vector of $\mathbf{v}$ with respect to the canonical basis $\{\mathbf{e}_1,\mathbf{e}_2,...,\mathbf{e}_n\}$.

The vector space V^n has two operations defined on it: addition of vectors and multiplication of vectors by scalars. The multiplication by vectors between themselves is not defined. The algebraic structure which involves multiplication between vectors is called an *algebra*.

An algebra A is a vector space over a field F together with a binary multiplication $\mathbf{ab}$ in A such that [Jacobson 1984]:

$$\begin{aligned}&(\mathbf{a}+\mathbf{b})\mathbf{c}=\mathbf{ac}+\mathbf{bc}\\&\mathbf{a}(\mathbf{b}+\mathbf{c})=\mathbf{ab}+\mathbf{ac}\\&\alpha(\mathbf{ab})=(\alpha\mathbf{a})\mathbf{b}=\mathbf{a}(\alpha\mathbf{b}),\quad \mathbf{a},\mathbf{b},\mathbf{c}\in A,\ \alpha\in F.\end{aligned}$$

In the case of the vector space V^n the field F is the set of real numbers. All that we should need in order to construct an algebra from V^n, would be to define a product $\mathbf{ab}$ between vectors in V^n. One particular product in V^n can be defined as follows.

Let us consider the vector space V^n with the inner product $<\mathbf{a},\mathbf{b}>$ and an orthonormal basis $\{\mathbf{e}_1,\mathbf{e}_2,...,\mathbf{e}_n\}$. We construct an algebra form V^n by introducing a product $\mathbf{ab}$ of vectors in V^n which satisfies the condition

$$\mathbf{ab}+\mathbf{ba}=2<\mathbf{a},\mathbf{b}>\mathbf{1}, \tag{1}$$

where $\mathbf{1}$ is the identity of the algebra. The product so defined is associative

$$\mathbf{a}(\mathbf{bc})=(\mathbf{ab})\mathbf{c}, \tag{2}$$

and we are constructing and algebra A equipped with an identity $\mathbf{1}$.

With the vector space and the product (1), the resulting algebra of all possible sums and products of vectors in V^n is called the *Clifford algebra* of V^n and is denoted by $C(V^n)$. Note in particular that

$$\mathbf{a}^2 = <\mathbf{a},\mathbf{a}>$$
$$\mathbf{e}_i^2 = 1$$
$$\mathbf{e}_i\mathbf{e}_j = -\mathbf{e}_j\mathbf{e}_i, \quad i \neq j$$

The Clifford algebra $C(V^n)$ is itself a vector space of dimension $\sum_{p=0}^{n}\binom{n}{p} = 2^n$, with basis

$$\{\mathbf{1}, \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n, \mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, \ldots, \mathbf{e}_1\mathbf{e}_n, \ldots, \mathbf{e}_1\mathbf{e}_2\cdots\mathbf{e}_n\},$$

such that an element A in $C(V^n)$ is written as

$$A = a_0 + a_{11}\mathbf{e}_1 + \cdots + a_{1i}\mathbf{e}_n + a_{21}\mathbf{e}_1\mathbf{e}_2 + \cdots a_{2i}\mathbf{e}_1\mathbf{e}_n + \cdots + a_{di}\mathbf{e}_1\mathbf{e}_2\cdots\mathbf{e}_n \qquad (3)$$

where $d = 2^n - 1$ and $i = \binom{n}{p}$ for the real numbers a_{pi}. Consequently, the vector space $C(V^n)$ can be decomposed in $n+1$ subspaces as:

$$C(V^n) = C_0 \oplus C_1 \oplus \cdots \oplus C_n. \qquad (4)$$

Each subspace is of dimension $\binom{n}{p}$.

The elements A (Equation 3) of the Clifford algebra $C(V^n)$ are called *multivectors* , and those of C_p, *p-vectors*. In particular, *0*-vectors are real numbers and $\dim(C_0) = 1$. C_1 has the basis $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$, so *1*-vectors are simply vectors and $\dim(C_1) = n$. C_2 has the basis $\{\mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, \ldots, \mathbf{e}_1\mathbf{e}_n\}$ and their elements (*2*-vectors) are also called *bivectors*. Finally, C_n has as basis $\{\mathbf{e}_1\mathbf{e}_2\cdots\mathbf{e}_n\}$ and since $\dim(C_n)$=1, the n-vectors of $C(V^n)$ are also referred as *pseudoscalars* that shall be denoted by I_n.

In what follows, arbitrary multivectors or p-vectors will be denoted by non bold uppercase characters without ornamentation such as A. In the particular case of vectors

(1-vectors), these will be denoted as in the very beginning, by bolded lowercase characters such as **a**. I_n will be reserved to pseudoscalars or n-vectors of $C(V^n)$.

Bearing in mind the decomposition of the vector space $C(V^n)$ as the direct sum of the subspaces C_p, $0 < p < n$, given in (4), any multivector A can be written as

$$A = \langle A \rangle_0 + \langle A \rangle_1 + \cdots + \langle A \rangle_n, \tag{5}$$

where $\langle A \rangle_p$ denotes the p-vector part of A, and $\langle\ \rangle$ is called the *grade operator*. If a multivector is such that $A = \langle A \rangle_r$ for some positive integer r, then A is said to be *homogeneous* of *grade r*.

The *magnitude* or *modulus* of a multivector A is defined by the equation

$$|A| = \langle A^* A \rangle_0^{1/2},$$

where * denotes the operation *reverse* defined as

$$(\mathbf{e}_1 \mathbf{e}_2 \cdots \mathbf{e}_p)^* = \mathbf{e}_p \cdots \mathbf{e}_2 \mathbf{e}_1.$$

The operation reverse is distributive [Hestenes and Sobczyk 1985] so that the reverse of an arbitrary multivector A can be easily calculated.

If the inverse of a multivector A exists, this is denoted by A^{-1} or **1**/A, and is defined by the equation $AA^{-1} = \mathbf{1}$.

Together the geometric product of multivectors, two more basic operations can be defined in a Clifford algebra. For a p-vector $A \in C_p$ and a q-vector $B \in C_q$ the *inner product* $A \cdot B$ is defined by

$$A \cdot B = \begin{cases} \langle AB \rangle_{|p-q|} & \text{if} \quad p, q \succ 0 \\ 0 & \text{if} \quad p = 0 \text{ or } q = 0 \end{cases} \tag{6}$$

Analogously, the outer product $A \wedge B$ (also called *Grassmann's exterior product*) is defined by

$$A \wedge B = \langle AB \rangle_{p+q}. \tag{7}$$

Inner and outer products arise frequently in multivector calculations and a remarkable property has place wen dealing with vectors, namely that

$$\mathbf{ab} = \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\wedge\mathbf{b}. \tag{8}$$

In this case, the geometric product has as symmetric part the familiar inner (or scalar) product $\mathbf{a}\cdot\mathbf{b}$ and as anti symmetric (associative) part the outer product $\mathbf{a}\wedge\mathbf{b}$, which is a bivector determined by the vectors **a** and **b**.

Bivectors have an interesting geometric interpretation [Hestenes 1987]. Just as a vector describes an oriented line segment, with the direction of the vector represented the oriented line and the magnitude of the vector is equal to the length of the segment; so a bivector describes an oriented plane segment, with the direction of the bivector represented the oriented plane and the magnitude of the bivector measuring the area of the plane segment. The same interpretation is extended to high-order terms: $\mathbf{a}\wedge\mathbf{b}\wedge\mathbf{c}$ represents an oriented volume, and so on.

A Package for Clifford algebra calculations

All that we should need in order to construct a package for multivector calculations, would be to define the two basic operations: geometric product and grade operator. In the first case, a simple algorithm for the computation of the geometric product between multivectors can be devised by noticing that a general multivector (3) in $C(V^n)$ is formed by a linear combination of terms of the form

$$\mathbf{e}_1^{m_1}\mathbf{e}_2^{m_2}\cdots\mathbf{e}_n^{m_n}, \tag{9}$$

where $m_i = 1,0 \quad (i = 1,\ldots,n)$. Let us call *blades* to multivectors of the form (9). The geometric product of two of these blades is:

$$(\mathbf{e}_1^{m_1}\mathbf{e}_2^{m_2}\cdots\mathbf{e}_n^{m_n})(\mathbf{e}_1^{r_1}\mathbf{e}_2^{r_2}\cdots\mathbf{e}_n^{r_n}) = (-1)^s\mathbf{e}_1^{m_1+r_1}\mathbf{e}_2^{m_2+r_2}\cdots\mathbf{e}_n^{m_n+r_n}, \tag{10}$$

where the sum $m_i + r_i$ is evaluated modulus two, and

$$s = \sum_{1\le i<j\le n} r_i m_j.$$

Equation 9 enables us to establish an isomorphism between blades and n-tuples $(m_1, m_2, \ldots, m_n)$ that can be more easily manipulated as it will be shown.

If we denote the j-th basis vector $\mathbf{e}_j$ as `e[j]`, a blade such as $\mathbf{e}_1\mathbf{e}_3\mathbf{e}_4$ ($\mathbf{e}_1^1\mathbf{e}_2^0\mathbf{e}_3^1\mathbf{e}_4^1$ using the nomenclature of equation 9) is written as

```
e[1] e[3] e[4],
```

and can be represented simply by `(1,0,1,1)`. Care must be taken in preserving the canonical order of the expression, since for instance `e[1]e[3]` is a geometric product of two vectors and `e[1]e[3]≠e[3]e[1]`.

Let us consider a Clifford algebra of V^{dim}. The following code implements the transformation of a blade onto a n-tuple:

```
ntuple[x_] := Module[{i=1,bin=Table[0,{i,dim}],
                  If[Length[x]==1, bin[[ x[[1]] ]] =1,
                      For[ , i<=Length[x], i++,
                         bin[[ x[[i,1]] ]] = 1
                         ]
                      ];
                       bin
                ]
```

The geometric product between two blades (Equation 10) can therefore be evaluated with help of the function `geoprod`:

```
geoprod[x_,y_] := Module[{p1=ntuple[x],
                          p1=ntuple[y],m,n,q=1,s,r={} },
                          s=Sum[p2[[m]]*p1[[n]],
                                {m,Length[p1]-1},
                                {n,m+1,Length[p2]}
                                ];
                          r=Mod[p1+p2,2];
                          For[m=1,m<=Length[x],m++,
                               If[r[[m]]==1,q*=e[m]
                                 ]
                              ];
                           (-1)^s*q
                     ]
```

This last function evaluates the geometric product of two blades in a Clifford algebra $C(V^{dim})$. One step further consists in to calculate the geometric product of two arbitrary multivectors as (3). This can be achieved from `geoprod` by providing the transformation rules which contains the properties of the geometric product under multiplication of blades by real numbers and addition of blades. Here is the behavior under scalar multiplication:

```
geoprod[a_,x_] := a x  /; FreeQ[a,e[_?Positive]]
geoprod[x_,a_] := a x  /; FreeQ[a,e[_?Positive]]
geoprod[a_ x_,y_] := a geoprod[x,y]  /; FreeQ[a,e[_?Positive]]
geoprod[x_,a_ y_] := a geoprod[x,y]  /; FreeQ[a,e[_?Positive]]
```

```
geoprod[a_ x_,b_ y_] := ab geoprod[x,y]  /;
            FreeQ[a,e[?_Positive]] && FreeQ[b,e[_?Positive]],
```

and the distributivity of addition:

```
geoprod[x_Plus,y_Plus] :=
                Distribute[tm[x,y],Plus] /. tm -> geoprod
geoprod[x_,y_Plus]:=  Distribute[tm[x,y],Plus] /. tm -> geoprod
geoprod[x_Plus,y_]:=  Distribute[tm[x,y],Plus] /. tm -> geoprod
```

To complete the basic operations of a Clifford algebra, it remains to implement the grade operator. The following auxiliary function calculates the grade of a blade

```
gradblade[a_] := 0  /; FreeQ[a,e[_?Positive]]
gradblade[x_] := Plus @@ ntuple[x]
gradblade[a_ x_] := gradblade[x]  /; FreeQ[a,e[_?Positive]]
```

Now, `grade[x,n]` should extract the term of grade n from the multivector X. Firstly, we consider the case when the multivector X is a blade of grade r: $\langle X\rangle_n = 0$ if $r \neq n$ and $\langle X\rangle_n = X$ if $r=n$. The code reads:

```
grade[x_,n_?NumberQ] := Module[{m=gradblade[x]},If[m==n,x,0]]
```

For a general multivector X we have:

```
grade[x_Plus,n_?NumberQ] :=
                Distribute[tm[x,n],Plus] /. tm -> grade
```

The functions `geoprod` and `grade` enable us to construct all the operations which can be defined in a Clifford algebra, such as outer product (`outprod[v,w,..]`), inner product (`innprod[v,w,..]`), magnitude (`magnitude[v]`), inverse (`mvinverse[v]`), reverse (`turn[v]`), dual (`dual[v]`), and many others, all included in the package **Clifford.m**. This package works with general multivectors of the form (3), but particular cases can help to envisage the power of multivector calculus, as is made explicit in what follows.

Vectors

Given a Clifford algebra $C(V^n)$, n-dimensional vectors are 1-vectors and lie in the subspace C_1. A vector **a** is therefore

$$\mathbf{a} = \langle A\rangle_1,$$

where A is a general multivector. The inner product defined in (6) becomes now the standard "dot product" between vectors. The "cross product " $\mathbf{a}\times\mathbf{b}$ of vector calculus is defined only in V^3 and is closely related to the outer product of vectors $\mathbf{a}$ and $\mathbf{b}$. The vector $\mathbf{a}\times\mathbf{b}$ is perpendicular to $\mathbf{a}\wedge\mathbf{b}$ and with the same magnitude $|\mathbf{a}\wedge\mathbf{b}|=|\mathbf{a}\times\mathbf{b}|$. The explicit algebraic relation between them is [Hestenes 1987]:

$$\mathbf{a}\times\mathbf{b}=(-I_3)(\mathbf{a}\wedge\mathbf{b}), \tag{11}$$

where $I_3=\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$ is the pseudoscalar of $C(V^3)$. We may actually take this as a definition of the cross product.

From (11) it is easy to define the function `crossprod[v,w]` that gives the cross product between two three dimensional vectors v and w:

```
crossprod[v_,w_] := geoprod[-e[1]e[2]e[3],outprod[v,w]]
```

The associativity of the geometric product (2) allows algebraic manipulations typical of real numbers that are no possible in the Gibbs' vector algebra since the cross and dot products are not generally associative. For example

$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c})\neq(\mathbf{a}\times\mathbf{b})\times\mathbf{c}.$$

Even more, many products such as $\mathbf{a}\cdot(\mathbf{b}\cdot\mathbf{c})$ are not even defined. With Clifford algebra all products are not only well defined but associative making simpler many algebraic manipulations and allowing to define derivatives and integrals just as they are defined for real functions of real variables, provided that we are careful to maintain the order of the factors since geometric product is not commutative.

One specific example that shows the simplicity of some expressions if geometric product is used, is concerning rotations. Consider a vector v in V^n which is rotated by an angle θ *in* the oriented plane characterised by the bivector $\mathbf{a}\wedge\mathbf{b}$. After the rotation, the vector v is transformed into v', given by [Hestenes 1987]:

$$\mathbf{v}'=U^{\dagger}\mathbf{v}U, \tag{12}$$

where

$$U=\cos(\theta/2)+\frac{\mathbf{a}\wedge\mathbf{b}}{|\mathbf{a}\wedge\mathbf{b}|}\sin(\theta/2).$$

The direction of the rotation (clockwise or counterclockwise) is specified by the orientation of the bivector $\mathbf{a}\wedge\mathbf{b}$. Equation 12 gives the rotated vector v' with no matter

the dimension of the space in which it is embedded. No corresponding simple expression exists in vector algebra.

As a quite simple example, which can be easily visualised, consider the vector $\mathbf{v}=(1,1,1)$, to be rotated 90° maintaining invariant the plane xy. To characterise the plane xy, we can use $\mathbf{a}\wedge\mathbf{b}$ where $\mathbf{a}=(1,0,0)$ and $\mathbf{b}=(0,1,0)$, in which case we get a rotation counterclockwise and is easy to see that $\mathbf{v}'=(-1,1,1)$) but if we use $\mathbf{b}\wedge\mathbf{a}$ then the rotation is clockwise and $\mathbf{v}'=(1,-1,1)$. Here is this example solved with **Clifford**:

```
In[1]:= << Clifford.m
In[2]:= v = e[1]+e[2]+e[3];
In[3]:= plane = outprod[e[1],e[2]]/magnitude[outprod[e[1],e[2]]]
Out[3]= e[1]e[2]
```

The operator U is now defined (U^{+} is the reverse of U):

```
In[4]:= u = Cos[Pi/4] + plane Sin[Pi/4];
In[5]:= vprime = geoprod[turn[u],geoprod[v,u]]
Out[5]= -e[1]+e[2]+e[3]
```

So we get a counterclockwise rotation where the vector $\mathbf{v}=(1,1,1)$ becomes $\mathbf{v}'=(-1,1,1)$.

Complex numbers

Let us consider the Clifford algebra of the most simple space that has geometrical structure: the plane V^2. Taking the canonical basis $\{\mathbf{e}_1,\mathbf{e}_2\}$, a basis for the Clifford algebra $C(V^2)$ is $\{\ ,\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_1\mathbf{e}_2\}$ and a general multivector A in $C(V^2)$ has the form

$$A=k_0+k_1\mathbf{e}_1+k_2\mathbf{e}_2+k_3\mathbf{e}_1\mathbf{e}_2.$$

We can express A as the sum of two multivectors: $A=A_{+}+A_{-}$, where A_{+} contain even grade elements and A_{-} the odd grade elements. That is

$$A=A_{+}+A_{-};$$
$$A_{+}=k_0+k_3\mathbf{e}_1\mathbf{e}_2,$$
$$A_{-}=k_1\mathbf{e}_1+k_2\mathbf{e}_2.$$

We focus our attention into the vector space formed by linear combinations of even multivectors A_{+}. This is itself an algebra so that it is called the *even subalgebra* of $C(V^2)$, denoted by $C_{+}(V^2)$. By taking

$$i = e_1 e_2, \tag{13}$$

an element of $C_+(V^2)$ can be written as

$$z = k_0 + k_3 i.$$

Since $i^2 = (e_1 e_2)(e_1 e_2) = -(e_1 e_2 e_2 e_1) = -1$, and i is itself a generator of rotations [Hestenes 1987], we see that $C_+(V^2)$ is equivalent to the algebra of complex numbers.

The algebraic operations of complex numbers can therefore be worked out with **Clifford**. In order to get a more standard notation we define firstly the function

```
In[6]:= trans[x] := x /. i -> e[1]e[2]
```

which makes the identification (13). Here we have two complex numbers:

```
In[7]:= w = a + b i;
In[8]:= z = c + d i;
```

The product `(a + b i)(c + d i)` becomes

```
In[9]:= geoprod[trans[w],trans[z]] /. e[1]e[2] -> i
Out[9]= a c - b c + b c i + a d i
```

The equivalence between some built-in basic operations of complex numbers in *Mathematica* and those of a Clifford algebra defined the package is shown in the following table:

Built-in Objets	**Clifford.m** Objets
`Re[z]`	`grade[z,0]`
`Im[z]`	`geoprod[grade[z,2],-e[1]e[2]]`
`Conjugate[z]`	`turn[z]`
`Abs[z]`	`magnitude[z]`

The inverse of the complex number `w` can be calculated with `mvinverse`:

```
In[10]:= mvinverse[trans[w]] /. e[1]e[2] -> i
```

$$\text{Out[10]} = \frac{a - b\,i}{a^2 + b^2}$$

Quaternions

Following the same lines as in the previous section, we now develop the Clifford algebra of the three-dimensional space V^3, equipped with the canonical basis $\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$. The Clifford algebra $C(V^3)$ is an eighth-dimensional vector space and has the basis $\{\ ,\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,\mathbf{e}_1\mathbf{e}_2,\mathbf{e}_2\mathbf{e}_3,\mathbf{e}_1\mathbf{e}_3,\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\}$. A general multivector A in $C(V^3)$ is written as

$$A = k_0 + k_1\mathbf{e}_1 + k_2\mathbf{e}_2 + k_3\mathbf{e}_3 + k_4\mathbf{e}_1\mathbf{e}_2 + k_5\mathbf{e}_2\mathbf{e}_3 + k_6\mathbf{e}_1\mathbf{e}_3 + k_7\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3,$$

Which can be also expressed as $A = A_+ + A_-$, where

$$A_+ = k_0 + k_4\mathbf{e}_1\mathbf{e}_2 + k_5\mathbf{e}_2\mathbf{e}_3 + k_6\mathbf{e}_1\mathbf{e}_3$$
$$A_- = k_1\mathbf{e}_1 + k_2\mathbf{e}_2 + k_3\mathbf{e}_3 + k_7\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3.$$

The even-grade elements A_+ form a subalgebra $C_+(V^3)$ of $C(V^3)$, equivalent to the algebra of quaternions. This can be seen by making the identifications

$$\begin{aligned}\mathbf{i} &= -\mathbf{e}_2\mathbf{e}_3,\\ \mathbf{j} &= \mathbf{e}_1\mathbf{e}_3,\\ \mathbf{k} &= -\mathbf{e}_1\mathbf{e}_2,\end{aligned} \tag{14}$$

leading to the famous equations

$$\begin{aligned}&\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1,\\ &\mathbf{ijk} = -1.\end{aligned} \tag{15}$$

With the identifications given in (14), an element of $C_+(V^3)$ can be written now as

$$Q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k},$$

which, in view of the properties (15), is a quaternion.

The algebra of quaternions is therefore comprised in the same package. The basic operations of this algebra are carried out by the function already defined, such as **geoprod, mvinverse, magnitude** and **turn**. To simplify the operations, we have incorporated the definitions (14) and redefined some functions to work only with quaternions and complex numbers. The new functions begin with the letter **q**, namely, **qprod, qinverse, qmagnitude** and **qturn**. So, for instance the inverse of the quaternion **q=a + 3 i + 6 j - 10 k** is

```
In[11]:= q = a + 3 i + 6 j - 10 k;
```

```
In[12]:= qinverse[q]
Out[12]= (a - 3 i - 6 j + 10 k) / (145 + a^2)
```

Grassmann algebra

The outer product (**outprod**) defined in (7) is associative and the identity **1** in a Clifford algebra $C(V^n)$ is also the identity for the outer product. Consequently, the vector space V^n with the outer product is an algebra of dimension 2^n, which is called the *Grassmann algebra* of V^n. It does not depend on the inner product of the vector space, but just on the alternantion of the outer product.

Grassmann algebra has relevance in modern theoretical physics [Lasenby, Doran and Gull 1993] and has the right structure for the theory of determinants. That the structure of this algebra is contained in Clifford algebra allows to reformulate Grassmann calculus [Lasenby *et al.* 1993], and to give an extensive treatment of determinants [Hestenes and Sobczyk 1985], both in terms of Clifford algebra.

Summary

We have introduced the Clifford algebra of a vector space and developed a package for calculations within this algebra. The relevance of Clifford algebra in physics and mathematics lies in the fact that it provides a complete algebraic expression of geometric concepts such as directed lines, areas, volumes, etc. (For this reason, Clifford algebra is also referred as *Geometric algebra* [Hestenes and Sobczyk 1985]). Quantities such as vectors, complex numbers, quaternions (and others non discussed here, such as Pauli matrices and Dirac spinors [Hestenes, 1971]), are naturally contained in a Clifford algebra. It becomes therefore an efficient mathematical language in a vast domain of physics.

It should be said that our package **Clifford** was though for calculations in n-dimensional crystallography using Clifford algebra [Gómez, Aragón, Dávila and Terrones, 1993] and consequently a coordinate system is intrinsic. This may represent a limitation since the most general formulation of a Clifford algebra is independent of a coordinate system. Although, as it was shown in this work, the present state of the package has several applications and can motivate their improvement to articulate the calculus with diverse mathematical objects in the same algorithm.

Acknowledgements

This work is a part of a larger project with A. Gómez and F. Dávila on the applications of Clifford algebra in modern crystallography. Many ideas of them were portrayed here. Financial support from CONACYT through grant 3348-E9308 is acknowledged.

References

Gómez, A., J.L. Aragón, F. Dávila and H. Terrones. 1993. A Clifford algebra approach to n-dimensional crystallography. *Proc. XVI Congress and General Assembly of the International Union of Crystallography (Beijing, China)*. p. 341.

Hestenes, D., and G. Sobczyk. 1985. *Clifford Algebra to Geometric Calculus. A unified Language for Mathematics and Physics*. D. Reidel Publishing Co. Holland

Hestenes, D. 1987. *New Foundations for Classical Mechanics*. D. Reidel Publishing Co. Holland

Hestenes, D. 1971. Vectors, Spinors and Complex Numbers in Classical and Quantum Physics. *Amer. J. Phys.* 39: 1013-1027.

Jacobson, N., 1984. *Lectures in Abstract Algebra II. Linear Algebra*. Springer Graduate Text in Mathematics 31. Springer-Verlag, New York.

Lang, S., 1969. *Linear Algebra*. Addison-Wesley. New York.

Lasenby, A., C. Doran and S. Gull. 1993. Grassmann Calculus, Pseudoclassical Mechanics, and Geometric Algebra. J. Math. Phys. 34: 3683-3712.