Mathematica Conceptos Generales y Aplicación.

David Fallas Valverde
Universidad de Costa Rica, ECCI,
San José, Costa Rica
David.fallas@ecci.ucr.ac.cr

and

Cristina Agüero Víquez
Universidad de Costa Rica, ECCI,
San José, Costa Rica
Cristina.aguero@ecci.ucr.ac.cr

Abstract

Beginning in the 70s mathematics and computing have been together in a level of development, application and research. As we know computing was born for the need of human to make long and complex mathematical calculations. Once a calculation could be described in steps, those could have been implemented in a computer and resolved the problem. As technology has increased in both fields, they have tended to separate, not because of his differences, as for the wide range of research and development that have maid. In this article we are presenting a platform whit almost 20 years of research, created for development and application of complex mathematical concepts. This tool is mathematica, one application that developed a symbolic language for the resolved and application of mathematical process. In this article we raised an introduction to the symbolic language of mathematica along whit a general review, then describe the compatibility of the platform with other languages and finally a look at the last version of mathematica, to find out what are the tools that they provided to operate the infinite world of computing and mathematics

Keywords: *Mathematica, symbolic language, compatibility.*

Resumen

A partir de los años 70 la matemática y la computación se han acompañado a nivel de desarrollo investigación y aplicación. Como sabemos la computación nace de la necesidad de los seres humanos de realizar complejos y largos cálculos matemáticos. Una vez que un cálculo podía describirse en pasos, estos podían ser seguidos por una computadora y solucionar así el problema. Conforme ha incrementado la tecnología en ambos campos, estos han tendido a separarse no por que exista una brecha entre ellos, sino más bien por la extensa gama de nichos de investigación y desarrollo que han abarcado. En este artículo presentamos una plataforma con casi 20 años de investigación creada para el desarrollo y aplicación de complejos conceptos matemáticos. Esta herramienta es Mathematica, esta aplicación desarrolló un lenguaje simbólico para el trabajo, la investigación y aplicación de procesos matemáticos. En este artículo planteamos una introducción al lenguaje simbólico de Mathematica junto con un análisis superficial del mismo, posteriormente describiremos la compatibilidad de la plataforma con otros lenguajes y finalmente una revisión a la última versión de Mathematica para conocer cuales son las herramientas que propone para la explotación del infinito mundo de la computación y la matemática.

Palabras clave*: mathematica, lenguaje simbólico, compatibilidad.*

1 Introducción

En el año 1986 Stephen Wolfram junto con su equipo de trabajo iniciaron la construcción de lo que resultó ser un poderoso y versátil sistema de álgebra computacional [2]. Desde mediados de los años 60 se contaba con la existencia de paquetes específicos para tareas matemáticas, sin embargo no existía un solo sistema que pudiera manejar las diversas técnicas de matemática computacional de una forma coherente y unificada de allí se genera el concepto de matemática [1].

Esto se logró bajo la implementación de un lenguaje simbólico con la capacidad de manipular a la gran variedad de objetos necesarios para lograr la generalidad requerida en un sistema computacional con un mínimo número de primitivas básicas [2].

Cuando matemática 1.0 fue lanzado en 1988 por la compañía Wolfram Research, fue aclamado por la comunidad técnica como una de las principales revoluciones intelectuales del momento; en un principio el impacto del nuevo sistema se sintió mayormente en la matemática y la física, y fue con el paso de los años como su importancia a crecido así como el rango de áreas que cubre [1,2].

En el plano técnico, mathematica es ampliamente considerado como una gran hazaña de la ingeniería de software. Es uno de los mayores programas de aplicación desarrollados, y cuenta con una amplia gama de nuevos algoritmos originales e importantes innovaciones que han sido desarrolladas y mejoradas a lo largo de sus 13 versiones lanzadas al mercado desde 1988 hasta la versión 6.0 en el presente año [1,2].

Para cubrir los objetivos del artículo se comienza tratando de las generalidades del lenguaje así como una serie de ejemplos básicos que permiten comprender como se da la interacción con el programa; mas adelante se contempla el impacto del lenguaje simbólico en mathematica, así como la creación de diferentes documentos de trabajo, luego de establecer una base se trata el tema del uso del programa en combinación otros lenguajes de computación, haciendo un énfasis en Java para la creación de programas que contemplen estas dos tecnologías. Para finalizar, se establecen las innovaciones que se presentan en la mas resiente versión, seguido de las conclusiones a partir de la investigación realizada.

A continuación se presentan las generalidades importantes de conocer con respecto a matemática.

2. Generalidades del Lenguaje.

El programa se encarga de manejar los aspectos mecánicos de las matemáticas, permitiendo de esta forma a los usuarios concentrarse en las implicaciones que conllevan los trabajos, lo cual aísla para el usuario la forma de resolver los problemas presentando solo respuestas, trayendo de esta forma mayor eficacia y reduciendo en gran medida el tiempo y el esfuerzo del usuario brindando información concisa pertinente y exacta que es de gran complejidad y muy útil para resolver toda una gama de problemas presentes en la sociedad actual [1].

En la actualidad los usuarios de matemática divididos en organizaciones comerciales o gubernamentales e instituciones académicas varían en utilidades que van desde las más comunes como la ingeniería, física, informática, hasta otras de gran complejidad como biología, aplicaciones científicas, entre muchas otras.

Juegan además un papel significativo en el modelado de sofisticados sistemas financieros y de análisis general. Para ampliar la variedad de funcionalidades que posee hace uso de su propia herramienta de desarrollo de software, con un lenguaje altamente conocido y estudiado, lo cual permite no solo utilizar las funciones brindadas por los creadores sino implementar las propias de acuerdo a las diferentes necesidades [2].

Para dar a entender que tan variada es la propuesta por parte de mathematica iniciamos mostrando ejemplo basicos matematicos que permiten entender como es la utilización del programa. Primeramente, es importante saber que maneja una variedad enorme de funcione matemáticas, desde funciones, ecuaciones diferenciales, integración, derivación, calculo de limites, creación de gráficos hasta funciones mucho más complejas y especificas, la mayoría de ellas se pueden trabajar desde las paletas de mathematica o bien a través de comandos a continuación se presentan algunos ejemplos.

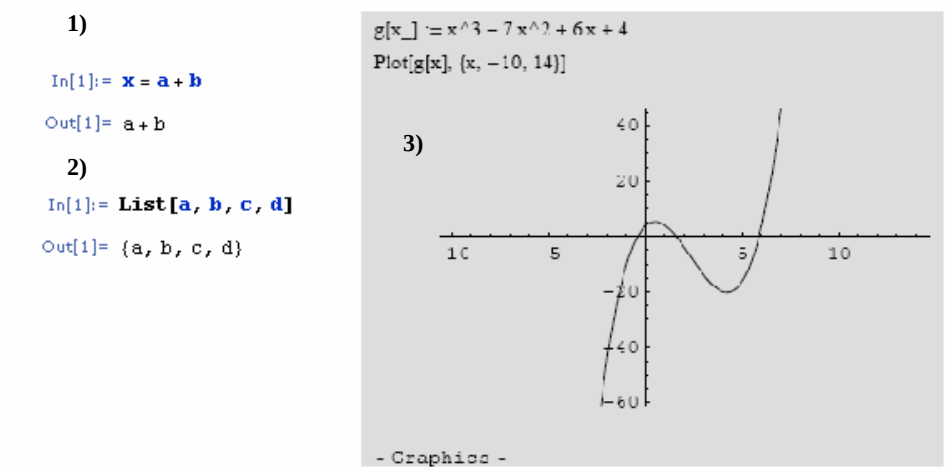

Figura1. Declaración de variables, declaración de listas, creación de gráficas.

Como se observa en la figura anterior el punto 1, indica como declarar variables, una vez declarada se reconoce a lo largo del programa, también se pueden declarar listas de elementos(punto 2) y la graficación como puede observarse se convierte en una aplicación rápida y simple(3), en la figura anterior primero se declara la función y luego se llama a graficar mediante el comando Plot. Así mismo observamos la resolución de ecuaciones simples a continuación.

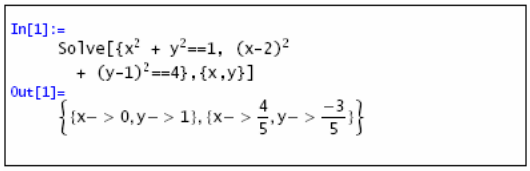

Figura2. Ecuaciones simples.

Otro caso interesante es el uso de matrices, ya que aparte de permitir aplicarle cualquier tipo común de operaciones el programa permite que el usuario trabaje de forma más amigable, ya que se le presentan de la forma como el esta acostumbrado a verlas. En la figura2 se presenta un ejemplo de cómo se crea la matriz y de que forma el programa lo muestra al usuario.

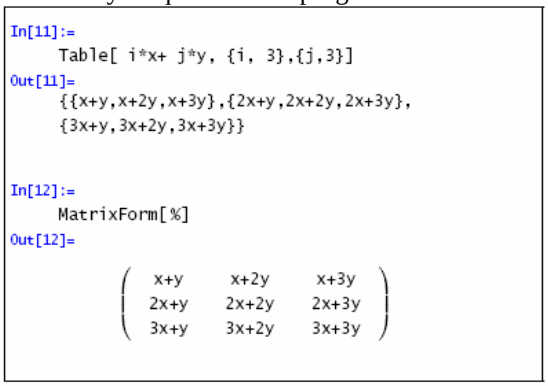

Figura3. Matrices.

Otro ejemplo importante es la utilización de las paletas con las que cuenta el programa que permiten omitir los comandos y en vez de esto darle clic a la opción deseada, a continuación se muestran algunas de ellas.

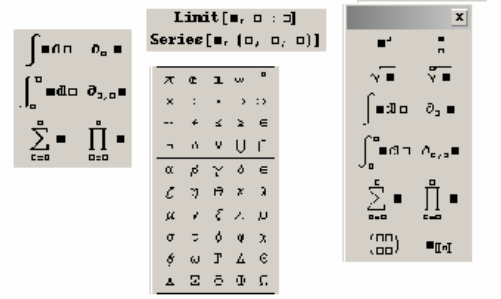

Figura4. Algunas paletas de funciones variadas.

Para ejemplificar el uso de las paletas de la figura 4 se presenta dos variantes al en el calculo de una integral, usando comandos o por la paleta. El caso uno de la figura 5, muestra el uso de la paleta mientras que el dos lo hace por medio de comando, claramente en una aplicación grande con cálculos complejos puede resultar mas cómodo para el usuario tener la primera opción a disposición.

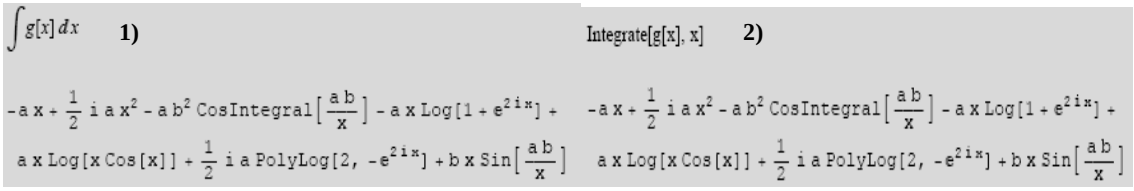

Figura5. Integrales ejecutadas de dos formas.

Todo lo presentado anterior es prueba de la facilidad de uso que presenta el programa acompañado de una extensa gama de funcionalidades que abarcan diferentes áreas de la ciencia y la matemática. Lo que permite que sea utilizado por todo tipo de usuarios desde los principiantes hasta los mas avanzados.

Para entender la forma correcta de utilizar *Mathematica* es preciso conocer la manera en que trabaja y las diferentes formas de cómo interactuar con el. El programa consta de dos partes, la primera llamada núcleo (kernel), es la que tiene a cargo ejecutar todos los comandos y realizar los cálculos involucrados, la segunda corresponde a la interfaz de usuario cuya función es permitir al usuario generar documentos interactivos en los que se incluyen los comandos a evaluar por el núcleo. Este tipo de documentos se conoce como *notebooks* [4].

La forma como interactúan estas dos partes la dirige el usuario, por lo que el núcleo no realiza ninguna acción hasta que el usuario lo indique, luego se carga el núcleo, y a partir de este punto se está en condiciones de comunicarse con el programa. Una entrada proporcionada por el usuario va de la mano con una salida que devuelve el programa a lo largo de una sesión de trabajo [4].

Para analizar más a fondo la manera de comunicarse del usuario con el sistema es necesario referirse al lenguaje simbólico, para conocerlo y entender el porque se utiliza.

2.1. Lenguaje Simbólico en mathematica

"Es en el núcleo de *Mathematica* donde se maneja la idea fundacional de que todos los programas, datos, formulas, graficas, documentos pueden ser representados como expresiones simbólicas."[3] Este es precisamente el concepto unificador bajo el que funciona el paradigma de programación simbólica, que caracteriza en gran parte el poder singular del lenguaje y en si del sistema.

Como se menciono previamente este tipo de lenguaje permite manipular una muy amplia variedad de objetos que son necesarios en el ámbito computacional utilizando solo un pequeño número de primitivas básicas. El sistema maneja formulas matemáticas, listas, gráficos, procesadores de señales, imágenes, audio, por solo mencionar algunos, y a pesar de ser aplicaciones con funcionalidades tan diferentes *Mathematica* los maneja de manera uniforme, cada uno de ellos es una expresión [5].

Y es precisamente mediante el uso de expresiones que se da la interacción entre el sistema y el usuario para la obtención de los diferentes cálculos y resultados. Sobre la representación del lenguaje simbólico mediante expresiones se trata a continuación.

2.2. Representación simbólica por expresiones

A manera de ejemplo se pueden observar diferentes operaciones que puede ingresar un usuario y que el sistema interpreta como expresiones: el usuario ingresa $x+y+z$ y el sistema interpreta Plus[x,y,z], así mismo para representar una multiplicación se convierte a Times[x,y,z], una lista que para el usuario se representa de la forma *{a,b,c}* mathematica la maneja como List[a,b,c]. Esto muestra como operaciones relativamente diferentes entre si son manejadas de la misma forma por el programa [5].

Queda claro que la noción de las expresiones dentro del lenguaje es un principio fundamental en lo que a la unificación del sistema concierne. Se debe al hecho que todos los objetos son manejados bajo la misma estructura lo que hace posible que mathematica logre cubrir tantas áreas con un número relativamente pequeño de operaciones básicas [4].

Mathematica le brinda al usuario la posibilidad de crear sus propias estructuras, mediante el uso de expresiones. Según las necesidades especificas de los diferentes usuarios es posible crear funciones que encapsulen un conjunto de expresiones y generan como resultado un calculo que podría ser de uso común para el usuario, de esta forma se logra disminuir el tiempo de trabajo y evita además el uso de secuencias repetitivas [4].

A partir del ejemplo mencionado anteriormente se puede observar una característica del lenguaje y es que permite el uso de notación especial para muchos operadores comunes; tal es el caso de entradas como x+y+z, x^y, {x,y,z} entre muchas más. Esto lo hace definiendo una gramática que especifica como la entrada del usuario se va a convertir en una expresión que entiende el lenguaje, en otras palabras especifica como se deben agrupar piezas de un *input* [5].

A continuación información brindada por la fuente [6].

Después de comprender la forma en que se manejan las expresiones es necesario conocer como comunicarse con el programa, para enviarle las operaciones requeridas, para este fin mathematica cuenta con documentos los que se pueden crear de tres formas:

1. Cuadernos de notas (*notebooks*), son documentos que permiten anidar texto, gráficos, sonidos y animaciones, ofreciendo la posibilidad de ejecutar el documento de forma interactiva.
2. Cuadernos de visualización, deben de ser abiertos con la utilidad *MatReader*, no pueden ser ejecutados.
3. Programas ejecutables, en entorno Mathematica Aplicaciones: Ingeniería, Ciencias Físicas, Ciencias Químicas, Ciencia de los Materiales, Ciencias Matemáticas, Ciencias de la Computación, Ciencias de la Salud, Negocios y Finanzas.

Es necesario como base para el manejo de mathematica el conocimiento básico de como funcionan los *notebooks* a partir de expresiones par poder iniciar el aprendizaje y la interacción con el programa.

2.3. *Notebooks* para la edición de documentos

Una alternativa interesante brinda mathematica y es la existencia de una serie de editores de texto específicos, cuya función es permitirle al usuario visualizar documentos de trabajo del programa, esto con el propósito de que el mismo pueda editarlos, leerlos o más aún modificarlos según sus necedades si tener que adquirir la versión completa del programa [4].

Estos editores no contienen el núcleo de mathamatica por lo tanto no le es posible al usuario realizar operaciones con ellos, simplemente es una herramienta que permite crear nueva funcionalidades. La compañía Wolfram Research cuenta con un editor gratuito, que puede ser descargado de la página de Internet de la empresa. El editor se llama MathReader y se puede encontrar gratuitamente en la siguiente dirección www.wolfram.com/products.

Otra de las aplicaciones que incorpora mathematica es la posibilidad de crear documentos de trabajo tales como artículos. Para ello, se incorporan opciones de formato tales como la elección del estilo de presentación, la fuente para el texto, el tamaño de la letra, el estilo de letra, la justificación del texto entre muchas otras [4].

Mathematica tiene disponible junto con el programa una ayuda muy completa y clara que se encuentra en el mismo programa, contiene las opciones *Help Browser* o *Master Index*, donde se puede acceder a abundante documentación, donde se pueden encontrar ejemplos, definiciones, enlaces entre otros más. Además de tener la posibilidad de acceder a la página de los creadores que cuenta con una amplia documentación, manuales y noticias de relevancia referentes al tema, sin mencionar la cantidad de enlaces de Internet que contienen información para miles usuarios del sistema [4].

Una característica que hace a *matemática* una de las principales plataformas para el desarrollo con componentes matemáticos es la compatibilidad que la misma brinda, con otros lenguajes de programación, por medio de la interfaz de comunicación *mathlink***.**

3. MathLink: Conexión Entre *Matemática* y Otros Programas.

Mathlink es una herramienta que permite a otros programas comunicarse con matemática y viceversa. Es posible entonces llamar funciones de programas desde *matemática* o llamar funciones de *mathematica* desde un programa externo, hacer intercambio de información y dar una diferente interfaz a una aplicación de *matemática*. [7]

Es común que existan aplicaciones para manejar a la operación con agentes. Con esta interfaz es posible crear programas que sean compatibles con *matemática* desde el diseño de su arquitectura interna, por lo tanto podrían conectarse ya sea por medio comunicación interna en una aplicación de escritorio, como en un aplicación distribuida que esté montada sobre una red de computadoras. [7]

Tener programas que sean compatibles de forma transparente con matemática, permite utilizar todas las expresiones que pertenecen a esta plataforma y emplearlas par el procesamiento de bloques

específicos de datos, procesamientos gráficos o cualquier otra necesidad que sea más eficiente y efectiva en *matemática.* [7].

La interfaz de comunicación de matemática es bastante amplia, actualmente términos como compatibilidad y portabilidad son los que determinan la funcionalidad de un sistema (generalmente), claro está acompañados por velocidad, tamaño y otros atributos que pueden asociarse a una aplicación. Mathematica ha sido reconocido por muchos académicos y estudiantes como uno de los programas más completos para desarrollo de algoritmos para solucionar modelos matemáticos complejos. Por medio de *mathlink, Mathematica* se sitúa en un lugar prestigioso pues abarca conexiones con lenguajes como c, c++, java, .Net de manera interna y con una interfaz especialmente diseñada para cada una de esas plataformas.

El análisis de imágenes médicas en nuestros días implica matemática avanzada, comentan los autores A. van Almsick, M. ter Haar Romeny y H. Bennink (grupo de Análisis de imágenes Biomédicas de la universidad de Eindhoven en Holanda) en el inicio de su artículo *Mathematical Prototyping in Medical Image Análisis* . En la introducción ellos se preguntan cuales son las características deseadas en un sistema para el desarrollo ágil de prototipos en imágenes médicas y de una extensa lista con un total de 16 requerimientos entre los cuales están: extensa manipulación simbólica, rendimiento gráfico rápido y de alta calidad, fácil de aprender, independencia de plataforma, soporte web y facilidad de crear una interfaz gráfica concluyen que *Mathematica* es la plataforma que cumple de forma más completa los requerimientos de esa lista.[13]

Lo anterior nos permite concluir que la compatibilidad en *Mathematica*, es un atributo necesario para que su efectividad pueda llevarse a niveles mayores en combinación con otros lenguajes de programación. Por ejemplo Java es un lenguaje que actualmente se ha posicionado de una manera exitosa en la industria de la programación. Como lo conocemos Java es un lenguaje que permite una manipulación limpia, clara, ágil, portable y poderosa del paradigma de programación por objetos. A continuación explicaremos un más específicamente la interacción de *Matemática y* Java por medio de la interfaz *J/Link.*

4. *Mathematica* y Java.

MathLink es el protocolo desarrollado por Wolfram Research's para el intercambio de datos y llamados funciones desde y hacia *Mathetica* y otros programas. El propósito de la creación de *J/Link* es poder ocultar muchas de las funciones y particularidades del protocolo de *mathLink*, para que el usuario de Java pueda utilizar de manera transparente a *mathematica* como una extensión del lenguaje y viceversa. Esto convierte a Java el lenguaje más indicado para realizar una aplicación junto *mathematica.*[8]

J/Link provee la opción al usuario de realizar una interfaz basada completamente en componentes Java desde la interfaz de programación de *mathematica.* Algunas de las funciones generales que brinda esta extensión son:

- LLamar métodos Java desde *Mathematica*.
- Escribir programas Java que utilizan servicios de *Mathematica*.
- Crear presentaciones alternativas para *Mathematica*.
- Crear cuadros de dialogo y otros elementos de interfaz de usuario para programas de Mathematica.
- Escribir applets que utilizan núcleos de *Mathematica* en el lado del cliente o del servidor.
- Escrbir servlets que ponen los servicios de Mathematica a disposición de clients HTTP.

[8]

Para iniciar la descripción iniciaremos por describir de forma general algunos de los elementos necesarios para llamar programas de java desde un programa en *Mathematica* y haremos una descripción general del procedimiento básico para utilizarlos. Es importante recordar que *J/Link* le permite al usuario de *Mahtematica* utilizar Java de manera tan transparente que no es necesario para el programador conocer Java a profundidad ni implementar algún programa en ese leguaje. Esto por que esta herramienta permite la creación de Objetos de Java directamente desde *Mathematica.*

A continuación iniciaremos con los pasos básicos para poder utilizar Java en un programa de *Mathematica,* incluiremos los comandos necesarios del lenguaje (en *Mahtmatica*), para que estos pasos sean describan de la manera más exacta posible el procedimiento a seguir. Estas subsecciones serán interesantes pues nos ayudaran a comprobar si existe esa verdadera transparencia en la utilización de Java para esta plataforma.

4.1 Cargar el paquete *J/Link* e inicializar el *runtime* de Java. [9]

Para cargar el paquete *J/link* en simplemente debe escribirse el siguiente comando:

In[1]:= Needs[“Jlink”]

Una ves que se ha cargado el paquete, el siguiente paso es inicializar el *runtime* (maquina virtual) de Java e instalarlo en *Mathematica*. Este paso se realiza con los siguientes comandos:

InstallJava[ ]: inicializa la maquina virtual de Java y la prepara para ser utilizada en *Mathematica.* Crea una *comand Line* el cual se utiliza para la inicialización del *runtime* de Java.
JavaLink[ ]: retorna el objeto que se utilize para la interacción con Java.

Cuando se utiliza el comando *InstalJava,* la salida de esa entrada para *Mathematica* es el Java *LinkObject.* Además de eso es importante mencionar que una de las opciones para ese comando es el *comandLine* esta opción se utiliza para indicar al *Mathmatica* el lugar exacto en el cual se encuentra la maquina virtual de Java. Claro está que hay un *comandLine* que es creado por omisión en *Mathematica* el cual corresponde al versión del *runtime* de Java asociada a *Mathematica 4.2* y posteriores y posiblemente los usuarios y/o programadores no se vean en la necesidad de modificarlo.
Sin embargo, si se deseara utilizar una versión distinta de Java a la asignada por defecto, pude utilizar la opción anteriormente mencionada y la sintaxis es la siguiente:

InstallJava[CommandLine -> "d:\\full\\path\\to\\java.exe"]

Una ves que hemos inicializado la maquina virtual de Java estamos preparados para comenzar a utilizar las clases que necesitemos, a continuación la descripción de cómo cargar las clases Java que requieramos en un programa *Mathematica.*

4.2 Cargar las clases de Java. [9]
La manipulación de las clases Java es llevada a cabo por medio del lo comando:

In[1]:= urlClass = LoadJavaClass["java.net.URL"]

Y el output (salida) correspondiente para este comando es:

Out[1]= JavaClass[java.net.URL]

Cuando se utilice el comando *LoadJavaClass* la salida correspondiente del mismo es un objeto que permite utilizar la clases y cada uno de sus componentes. Es importante especificar que esta es la única forma en la que podemos referenciar clases de Java, si utilizamos simplemente el nombre de la clase no sería posible acceder a ninguno de sus métodos.
Ahora una ves que la clase ha sido cargada en *Mathematica*, el programador tiene la opción de crear objetos o instancias de esa clase he invocar sus métodos o utilizar cualquiera de sus campos; puede acceder a sus métodos estáticos y tiene acceso a todos los métodos constructores y públicos de la clase que ha sido cargada.

Algunos otros comandos que existen para la manipulación de las clases de Java son los siguientes:
StaticsVisible: colocado el valor en *true* hace accesibles los métodos y campos estáticos de la clase solo por su nombre, sin necesitar un contexto especial.
AllowShortContext: establecido en *false* esta opción hace accesibles a los métodos y campos estáticos de la clase únicamente en el contexto de la propia clase.
UseTypeChecking: si se establece en *false* se elimina el chequeo de tipos en las definiciones de los llamados a funciones Java.

Ahora que podemos cargar nuestras clases Java en *Mathemati*ca, es importante, antes de entrar en la sección de la creación física de objetos y la funcionalidad del acceso a campos e invocación a métodos, hacer referencia a la manipulación que realiza *Matemática* de los tipos de datos, y como realiza la conversión entre los tipos de datos de Java y los que utiliza internamente.
[calling java from]

4.3 Conversión de tipos entre Java y *Mathematica*.

Una ves que Java ejecuta algún procedimiento o función y retorna un valor a *Matemática,* inmediatamente esta plataforma trasforma ese valor de Java en un expresión que pertenezca a su lenguaje. A continuación se presentará una tabla del mapeo de tipos entre Java y *Matemática.* [9]

Tipo Java	Tipo *Mahtematica*
byte, char, short, int, long	Integer
Byte, Character, Short, Integer, Long, BigInteger	Integer
Flota, double	Real
Flota, Double, BigDecimal	Real
boolean	True o false
String	String
array	List
controlled by user (see Complex Numbers)	Complex
Object	JavaOject
Expr	any expression
null	Null

Tabla 1. Mapeo de tipos Java-Mathematica. [9]

EL mapeo de tipos es un factor importante cuando se desarrolla una aplicación en diferentes plataformas, una ves que conocemos como se realiza podemos ahorrar tiempo y recursos pues existe una mayor claridad en los momentos de almacenar o asignar los resultados de algún método así mismo como en la definición de laguna función. Seguiremos ahora con nuestro siguiente paso en la utilización de Java en *Mathemática*¸ la creación de instancias de clases su manipulación y uso.[9]

4.4 Creación de Objetos Java.

La creación de Ojetos de Java se administra por medio del comando *JavaNew*, el cual recibe como primer parámetro la clase del Objeto que se va a crear. Este nombre de la clase puede ser especificado como un *string* con el nombre completo de clase o con la salida del método anteriormente mencionado *LoadJavaClass* y de forma continua los demás parámetros del método constructor de la Clase, la sintaxis sería la siguiente: [9]

In[1]:= frm = JavaNew["java.awt.Frame"]

Lo cual produciría la siguiente salida en la consola de *Mathematica:*

Out[1]= <<JavaObject[java.awt.Frame]>>

La expresión de salida es definida en *Mathematica* como un JavaObject, este tipo de expresiones se utilizan para encapsular el objeto Java y su utilización. Estas expresiones utilizan los símbolos “<<”, “>>” para indicar la que le objeto es accesible pero que no podemos entrar en ni tomar atributos y utilizarlos según queramos. Las expresiones JavaObject son específicamente diseñadas para ser utilizadas con *J/Link*, en métodos que las reciben como parámetro. Por ejemplo si deseamos conocer el nombre de un objeto no podemos escribir First[OBJ], sino más bien hacerlo por el método de *J/Link* ***ClassName[obj]***.

Existen también otras formas que provee *J/Link* para crear y manipular objetos Java, estas son: *MakeJavaObject y MakeJavaExpresion.* Sin embargo, estas no son las opciones estándar para la manipulación creación de objetos y se usan para fines específicos. La primera se utiliza para crear un objeto cuyo valor sea usan expresión anteriormente definida y el segundo para representar una expresión de *Mathematica* como un objeto de Java, es decir encapsular la expresión con en un objeto. Estas otras opciones no serán cubiertas en este trabajo, pues no están dentro de nuestros objetivos.

El ultimo paso que necesitamos conocer ahora es como acceder a los métodos y campos que están en los objetos que podemos crear. La última subsección de esta parte de nuestro escrito explicara de forma general como es que se pueden realizar tales accesos.

4.5 Acceso a Métodos de Clases y Campos.

La sintaxis utilizada en *Mathematica* para trabajar e interactuar con los objetos es muy similar a la que se utiliza en Java. A continuación se resumirán las principales expresiones sintácticas del lenguaje Java y su

respectiva escritura en *Mathematica* por medio de una tabla en la cual se compara la escritura en *Mathematica* y la escritura en Java. [9]

Expresión Sintáctica	Escritura en Java	Escritura en *Mathematica*
Constructores	MyClass obj = new MyClass(args);	obj = JavaNew["MyClass", args];
Métodos	obj.methodName(args);	obj@methodName[args]
Campos	obj.fieldName = 1; value = obj.fieldName;	obj@fieldName = 1; value = obj@fieldName;
Métodos Estáticos	MyClass.staticMethod(args);	MyClass`staticMethod[args];
Campos Estáticos	MyClass.staticField = 1; value = MyClass.staticField;	MyClass`staticField = 1; value = MyClass`staticField;

Tabla 2. Sintaxis *Mathematica*

Se puede concluir de la tabla anterior que efectivamente la sintaxis en ambos leguajes es muy similar, los únicas diferencias son más que todo a nivel de símbolos para elementos como el operador de "acceso" a un objeto (“.” en Java y “@” en *Mahtematica*) y el símbolo para los métodos estáticos. Para mayor claridad a continuación se presenta un ejemplo simple de interacción con *J/Link*

```
In[1] := date = JavaNew["java.util.Date"] Out[1]=«JavaObject[java.util.Date] »
In[2]:=LoadJavaClass["java.text.DateFormat"];
      dateFormatter = DateFormat`getInstance[]
Out[3]:=«JavaObject[java.text.SimpleDateFormat] »
In[4]:= dateFormatter@format[date]
Out[4]=10/9/00 4:56 AM
```

En este punto ya podemos decir que hemos repasado los pasos generales necesarios para incluir clases y objetos Java en un programa de *Mathematica*, inicializarlos y interactuar con ellos*;* con esto se concluye nuestra segunda sección de este trabajo. En la última sección queremos de forma general introducir a las nuevas propuestas de tecnología que *Mathematica* está ofreciendo al mercado y cuales son las herramientas que implementan esas propuestas.

5. Innovación en *Mathematica : Mathematica* 6.0.

Mahtemaica 6.0 es la última versión comercial que se ha comercializado por parte de Wolfram´s Research, esta versión está revolucionando todas las herramientas que esta plataforma había ofrecido a hasta sus versiones anteriores. Gracias a su arquitectura y el diseño de su lenguaje simbólico, con su última versión *Mathematica* logra crear una nueva experiencia de interacción con la computación y la matemática. Gracias a sus investigaciones, el equipo Wolfram´s research ha desarrollado un nuevo sistema de que permite realizar interfaces que hacen verdaderamente dinámica la interacción con los modelos matemáticos realizados en *Mahtematica.*

En esta última versión han hecho un agregado de más de mil nuevas funciones además de optimizar las construidas anteriormente [10]. En este artículo nos enfocaremos en una herramienta con una aplicación muy grande en campos muy diversos. Esta es la función de *Manipulate*, con este comando el programador pude crear aplicaciones con un rango amplio de funciones interactivas las cuales son muy innovadoras para el usuario final de la aplicación.

5.1 Comando *Manipulate*.

El comando *Manipulate* es un componente muy poderoso para la creación de interfaces interactivas. Por medio de el se pueden controlar los parámetros que modifican el comportamiento de una expresión definida en *Mathemtica* y mostrarlo gráficamente y en tiempo real al cambio de los parámetros. Para cada parámetro en la expresión Manipulate genera un control (slider)[11].El formato de una expresión *Mnipulate* es el siguiente (en su forma más simple):

Manipulate[expr, {u, umin, umax}][12]

En ese comando la palabra **expr** puede ser sustituida por cualquier función definida en *Mathematica*, **u** es la variable que se desea manipular y **umin, umax** son el valor mínimo y máximo que tendrá u. Ahora veremos un ejmplo que con un expresión definida para poder visualizar el resultado que produce el comando.

In[1]=Manipulate[Plot[Sin[n x], {x, 0, 2 Pi}], {n, 1, 20, Appearance -> "Labeled"}]

[11]

Out[1]=

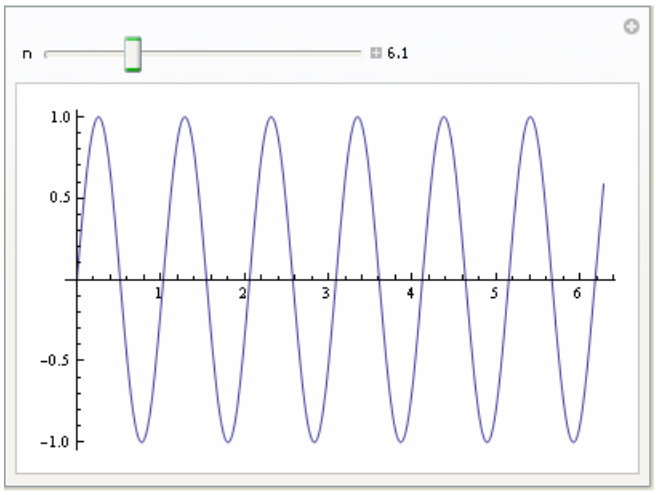

Figura 3. Resultado de Comando *Manipulate.[11]*

Analizando el ejemplo anterior podemos ver que es muy sencillo realizar una interfaz gráfica para ver el comportamiento de la función Sin(nx). El "slider", que es el termino asignado a la barra que muestra el cambio en la variable n, indica el valor actual de n conforme ese cambia la interfaz se modifica dinámicamente en tiempo real para reflejar el cambio en la variable.

Este es el ejemplo que usaremos para este artículo, sin embargo pueden encontrarse numerosos ejemplos más con funciones mucho más complejas e interesantes, no obstante no muchas más líneas de código, en la siguiente dirección url: http://demonstrations.wolfram.com/**.** Por medio de este sitio en internet, *Wolfram´s Research* ha iniciado un programa llamado "*Wolfram´s Demostration Proyect*" con el cual ponen en la *world wide web* (www) muchos ejemplos de aplicaciones interactivas con figuras y representaciones de funciones que pueden ser utilizadas por jóvenes de colegio hasta estudiantes universitarios. Este proyecto es parte de las innovaciones que esta planteando *Mathematica* para esta nueva versión, la cual aumenta su alcance en cuanto a aplicabilidad de su software.

Es importante resaltar la facilidad con la que se puede lograr un resultado óptico he interactivo con simplemente una línea de código, *Mathematia* no solo tiene definidas muchas funciones matemáticas, sino que además tiene objetos para graficar figuras en dos y tres dimensiones lo cual nos permitiría tener una acceso muy simple a elementos que en otro lenguaje de programación no son tan fáciles de manejar. Si a esto le agregamos la posibilidad de dar al desarrollo de un programa en *Mathematica* una interfaz con componentes Java, estamos hablando de una herramienta multiplataforma muy poderosa con un rango de aplicación muy extenso.

6. Conclusiones

Desde su lanzamiento en 1988, *Mathematica* creó una revolución en el contexto tecnológico en el que se encontraba. Ahora con el lanzamiento de la nueva versión 6.0 lo esta logrando de nuevo no solo por la trayectoria que ha tenido alrededor de los años, sino por que está implementando herramientas que la están convirtiendo en una plataforma con soporte web a nivel mundial y con una capacidad respaldada de creación de algoritmos propios y eficientes, que facilitan de sobremanera la interacción con elementos del modelado matemático con características altamente complejas.

Mathematica desarrolló años atrás su propio lenguaje simbólico para la edición programas matemáticos que ahora se está espandiendo a aplicaciones con fines no únicamente matemáticos, sino que son de interés para la humanidad y que aportan desarrollo a todo el planeta. Por ejemplo lo podemos ver en artículo *Mathematical Prototyping in Medical Image Análisis*[13], donde sus autores concluyen que *Mathematica* es la herramienta más completa para lograr el desarrollo e investigación médica que ellos necesitaban.

En este artículo se han identificado elementos de esta plataforma que la sitúan como un de las mejores del mercado. Esta se ha preocupado por ampliar su rango de alcance en cuanto a las aplicaciones que en ella se desarrollan. Proyectos como J/Link o la interfaz de programación en .Net, no mencionada revisada en este artículo pero si existente, hacen que *Matemática* intervenir en muchos más ambientes que simplemente en el matemático como pudo podría catalogarse.

Creemos que es de vital importancia para el continuo desarrollo tecnológico que compañías de desarrollo como *Wolfram´s Research* inviertan en dar características aplicables y amigables a programas que tienen un contenido muy poderoso para el desarrollo intelectual de la población en general. Por medio de programas como "*Demostration Program*" estas nuevas herramientas de aprendizaje pueden difundirse a inclusive estudiantes que están en grados preuniversitarios, lo cual los va introduciendo de forma muy atractiva al mundo de computación y la matemática por medio del cual podrán a su vez diseñar nuevas herramientas que permitan a la medicina, biología, química y otros campos de estudio avanzar en proyectos intelectuales que beneficien a toda la comunidad a nivel mundial.

7. Bibliografía

- [1] Wolfram Research, History & Background, http://www.wolfram.com/products/mathematica/history.html, 2007
- [2] Mathematica, http://es.wikipedia.org/wiki/Mathematica, 22 oct 2007
- [3] Wolfram Mathematica. Documentation Center, http://reference.wolfram.com/mathematica/guide/Expressions.html, 2007
- [4] Eugenio Manuel Fedriani y Alfredo García Hernández, Guía rápida para el nuevo usuario de Mathematica 5.0, mayo 2004
- [5] Wolfram Mathematica. Documentation Center, http://reference.wolfram.com/mathematica/tutorial/ExpressionsOverview.html, 2007
- [6] Informacion Listas de Distribución de RedIRIS, http://www.rediris.es/list/info/mathema.es.html, 19/02/1997
- [7] Wolfram Mathematica. Documentation Center, http://reference.wolfram.com/mathematica/tutorial/HowMathLinkIsUsed.html,2007
- [8] Wolfram Mathematica. Documentation Center http://reference.wolfram.com/mathematica/JLink/tutorial/Introduction.html, 2007
- [9] Wolfram Mathematica. Documentation Center http://reference.wolfram.com/mathematica/JLink/tutorial/CallingJavaFromMathematica.html,2007
- [10] Wolfram Mathematica. Documentation Center. http://reference.wolfram.com/mathematica/guide/NewIn60AlphabeticalListing.html,2007
- [11] Wolfram Mathematica. Documentation Center http://reference.wolfram.com/mathematica/tutorial/IntroductionToManipulate.html
- [12] Wolfram Mathematica. Documentation Center http://reference.wolfram.com/mathematica/ref/Manipulate.html

[13] Markus A. van Almsick, Bart M. ter Haar Romeny, Edwin H. Bennink. "Mathematical Prototyping in Medical Image Analysis", Eindhoven University of Technology,Department of Biomedical Engineering, Biomedical Image Analysis Group.