UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

DEPARTAMENTO DE SISTEMA OPERATIVO Y DESARROLLO DE TECNOLOGÍAS LIBRES

Facultad 10

QTOCTAVIZ: MÓDULO PARA LA VISUALIZACIÓN CIENTÍFICA INTEGRADO AL ASISTENTE MATEMÁTICO OCTAVE.

Trabajo presentado en opción al título de

Ingeniero en Ciencias Informáticas

Autores:

Miguel Angel Chávez Alfonso

Jorge Luis Hernández Cruz

Tutor:

MSc. Alexeis Companioni Guerra

Ciudad de La Habana

Mayo de 2010

"Así como los objetos más fáciles de ver no son los demasiado grandes ni los demasiado pequeños, también las ideas más fáciles en matemáticas no son las demasiado complejas ni las demasiado simples."

Bertrand Arthur William Russell

Declaración de autoría

Declaramos ser los autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firman la presente a los _____ días del mes de _____ de 2010.

Miguel Angel Chavez Alfonso

Jorge Luis Hernández Cruz

MSc. Alexeis Companioni Guerra

Agradecimientos

La culminación del presente trabajo, no hubiese sido posible sin la contribución de un grupo de personas, amigos todos, que brindaron parte de su tiempo en pos de auxiliarnos en nuestro afán de confeccionar, lo que constituye el punto cumbre de nuestra vida de estudiante. Nos gustaría infinitamente mencionarlos a todos, pero fueron incontables los compañeros que de una forma u otra aportaron a nuestra labor, ya fuera con sus conocimientos o su apoyo moral. No quisiéramos dejar pasar por alto la ocasión, para hacerles llegar nuestro más sincero agradecimiento. La más profunda y especial gratitud a nuestros padres, por habernos guiado por el camino correcto y ser nuestro sostén, nuestra inspiración, nuestra vida. Agradecimientos especiales también a nuestro tutor y amigo, que ha constituido un pilar importante en la materialización de este sueño, quien con su abnegación, esmero y paciencia ha sido un ejemplo para nosotros. Gracias a Michaell González Jorrin por su orientación en la metodología de investigación científica del trabajo. A Ernesto Delgado Clavero por su colaboración en la ingeniería del sistema. A Sandy Díaz Ramos por su colaboración en los ejercicios propuestos. A nuestro gran amigo Carlos Alberto García por su apoyo moral y confiar en todo momento en nosotros. A Alexander Martínez Fajardo por haber contribuido a un mejor funcionamiento de nuestras herramientas de trabajo. A nuestros amigos de la vieja escuela Manuel Cabrera Murillo, Orlando Merayo Maceda, Carlos Rafael Cabrera Cedeño, Javier Menéndez Rizo, Isleam Balceiro Rodríguez, Reinier Rivera Díaz, Adnier Rosello Carrazana, Yunier Soler Franco, Ángel Antonio Ravelo Batista, Camilo Trujillo Pacheco, Edilberto Alcolea, Yoanni Navarro, Victor Rosendo Junquera, Hansel Laza, Adalberto Ladron, por los buenos y malos momentos que pasamos juntos. A nuestras amistades más cercanas Daliana Montada, Heydi Alonso, Mirelis Torres. A los profesores que ayudaron en nuestra formación y en especial a nuestros profes y amigos Yanko Hernández, Raciel Garrido, Yaumara Arce, Geykel Diaz En fin, para no olvidar a nadie . . .

A todos, muchas gracias.

Dedicatoria

Miguel Angel Chavez Alfonso:

A mi madre Idalmis Alfonso por estar siempre presente cuando la necesito, por brindarme su amor y cariño todos estos años. Por confiar siempre en mi y por lo mucho que la quiero.

A mi padre Angel Chávez por brindarme su apoyo en todo momento y hacer posible que este momento ocurra. Por su confianza y esfuerzo todos estos años.

A mis abuelitas por su cariño todos estos años.

A mi hermanita Noemí Chávez y a mi sobrina Aliansi Hernández por su eterno amor, y sus más sinceros deseos de mi bienestar en todo momento.

A mi prima Yamilka Rodríguez por ser para mi como una hermana.

A Iracema mi inspiración, mi tesoro, por su amor y comprensión en estos años.

Jorge Luis Hernández Cruz:

A mi madre Ana Eloisa Cruz por ser mi faro y mi guía, por brindarme todo su amor y cariño, por confiar siempre en mi.

A mi padre Jorge Luis Hernández por su apoyo y confianza.

A mis abuelos por todo su amor y apoyo.

A toda mi familia por estar siempre a mi lado.

A mi padrino Avilio Rodríguez por toda su ayuda y consejos.

A mi novia Yuliet Legra por su amor, por su comprensión y por su paciencia para soportarme.

A mis amigos...

Resumen

A partir del creciente desarrollo experimentado por las Ciencias de la Informática desde las últimas décadas, surge la necesidad de contar con herramientas que ofrezcan un amplio abanico de opciones de acceso a la información, concediendo a los usuarios un elevado nivel de intervención-decisión a través de un gran número de objetos interactivos. Los asistentes matemáticos, como obra humana y científica, no escapan a este paradigma y de igual manera se trata que lleguen hasta los usuarios finales de la forma más amena y utilitaria posible. Los sistemas sobre los que se soporta Octave de modo particular para la visualización de sus resultados carecen, de manera general, de una eficiente interactividad lo que hace engorroso la gestión de las propiedades de las representaciones y obstaculiza en buena medida los proceso de enseñanza y/o aprendizaje de quienes buscan el conocimiento mediante su empleo.
El propósito del presente trabajo consiste en desarrollar un graficador interactivo para la Visualización Científica desde el ambiente de Octave que disponga de elevadas prestaciones para la representación de funciones en 2D y 3D, así como para la gestión completa de la información relativa a estas. De manera adicional, mediante el empleo de la nueva propuesta, se abstrae al usuario final de la necesidad de conocer engorrosos comandos para el tratamiento de las representaciones y este solo se concentra en la interacción con un popular sistema de ventanas y botones, que propicia fácil acceso a la totalidad de la información que puede gestionarse en el sistema.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE TÉRMINOS

script: Órdenes que por lo regular se almacena en un archivo de texto plano las cuales son pasadas a un intérprete para que las ejecute. No cumplen la definición de programa porque no son ejecutables por ellos mismos.

M-File: script con extensión .m que contiene varios comandos o la definición de funciones en lenguaje Octave.

Gráfica: Representación de datos, generalmente numéricos, mediante líneas, superficies o símbolos, para ver la relación que guardan entre sí. También puede ser un conjunto de puntos, que se plasman en coordenadas cartesianas, y sirven para analizar el comportamiento de un proceso, o un conjunto de elementos o signos que permiten la interpretación de un fenómeno.

Gráficador o sistema de graficación: Programa informático con el que pueden realizarse gráficos, dibujos y retocar fotografías. Permite crear, almacenar y modificar imágenes, utilizando puntos, líneas y figuras geométricas. También brinda la posibilidad de incorporar textos cortos y dibujos sencillos.

Diagrama de contorno: Un gráfico de contorno proporciona una vista en dos dimensiones del diagrama de superficie visto desde arriba, de forma parecida a un mapa topográfico en dos dimensiones. Las líneas de los gráficos de tipo Contorno conectan puntos interpolados de igual valor.

Diagrama de mallas: Colección de triángulos y vértices que aproximan una superficie en 3D.

Diagrama de superficie: Diagrama que representa las variables mediante áreas proporcionales de tramos y colores. Suelen ser circulares, semicirculares o rectangulares. Como

en un mapa topográfico, los colores y tramas de un gráfico de superficie señalan áreas que contienen el mismo rango de valores.

Licencia Privativa: Es el tipo de licencias que debe adquirir cualquier usuario para utilizar un programa privativo a través del pago de una cantidad determinada de dinero al autor del software en cuestión. Generalmente la licencia es sustentada por un documento o contrato que la casa distribuidora de software entrega al usuario para que este pueda sustentar ante las entidades de control la legalidad en la adquisición de los programas que utiliza. No permiten la distribución de copias sobre el Software que cubren, en otras palabras, si usted paga por una Licencia de Windows XP, usted tiene derecho a usar una copia del mismo, más sin embargo no puede hacer copias del instalador y mucho menos distribuirlas

Migración: Conjunto de actuaciones cuya finalidad es la sustitución de infraestructuras tecnológicas apoyadas en un software de una licencia por otra con funciones equivalentes por ejemplo la migración un programa de software propietario a otro de Software Libre.

Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente. Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

EIDMAT: Entorno de trabajo para el desempeño matemático que toma como base el asistente matemático Octave y extiende sus potencialidades mediante una interacción con los asistentes Maxima y R (con éste último aun no liberada).

Frontend: Software que provee de una interfaz más "amigable" a otra aplicación que si bien es la que realmente administra la información, su uso directo representa algún grado de dificultad para el usuario al que está destinada.

Instalación: Incorporar a la computadora un programa o dispositivo para ser utilizado.

Conexión o comunicación: Transferencia de datos desde un programa informático a otro.

Artefacto: Elementos de entrada y salida de las actividades. Productos tangibles del proyecto. Las entidades que el proyecto produce o usa para componer el producto final (modelos, documentos, código, ejecutables...).

API: Interfaz de Aplicación del Programa. Grupo de rutinas del sistema operativo o de una aplicación que definen cómo invocar cualquier servicio desde un programa.

Entorno de Desarrollo Integrado o IDE: Programa informático compuesto por un conjunto de herramientas de programación.

Introducción

Desde la prehistoria el hombre trató de reproducir en las paredes de las grutas las formas de los animales que había observado, logrando dar idea de sus movimientos y la forma de los cuerpos. La representación tridimensional comenzó con la denominada "perspectiva natural", practicada en la antigüedad clásica, recurriendo a métodos combinados para representar la problemática de las distancias a partir de distintos recursos para simular el concepto de lejanía de los objetos en el plano de representación [1].

Con el desarrollo de la ciencia a lo largo de la historia, el hombre siempre ha pretendido visualizar los diferentes fenómenos para así lograr un mejor entendimiento de estos, permitiendo mediante diferentes técnicas representar de forma gráfica sus comportamientos y efectos. Científicos, ingenieros, personal médico, analistas comerciales y otros con frecuencia necesitan analizar grandes cantidades de información o estudiar el comportamiento de ciertos procesos. Las simulaciones numéricas efectuadas en supercomputadores a menudo producen archivos de datos que contienen miles e incluso millones de valores de datos. De modo similar, cámaras vía satélite y otras fuentes acumulan grandes archivos de datos más rápido de lo que se pueden interpretar. El rastreo de estos grandes conjuntos de números para determinar tendencias y relaciones es un proceso tedioso e ineficaz. Pero si se convierten los datos a una forma visual, es frecuente que se perciban de inmediato las tendencias y los patrones. La representación de datos gráficamente como un medio para ganar conocimiento y comprensión de los datos es ocupada por la disciplina Visualización Científica de Datos. Existen muchas clases de conjuntos de datos y los esquemas de visualización efectivos dependen de las características de los datos. Una compilación de datos contiene valores escalares, vectores, tensores de orden superior o cualquier combinación de estos tipos de datos,

y los conjuntos de datos pueden ser bidimensionales o tridimensionales. La codificación de colores es sólo una manera de visualizar un conjunto de datos, las técnicas adicionales por su parte incluyen trazos, gráficas y diagramas de contorno, presentaciones de superficie y visualización de interiores de volumen; además, se combinan técnicas de procesamiento de imágenes con gráficas por computadora para crear muchas de las visualizaciones de datos. Las comunidades de matemáticos, científicos físicos y otros utilizan técnicas visuales para analizar funciones matemáticas y procesos o sólo con el propósito de crear representaciones gráficas interesantes[2].

Los graficadores empleados por los asistentes matemáticos constituyen una de las principales herramientas que utiliza la ciencia moderna para la Visualización Científica de Datos. En el caso particular de los asistentes matemáticos de esencia numérica el más conocido y empleado internacionalmente es Matlab[3], dada la cantidad de prestaciones que posee y el nivel de integración, extensibilidad y facilidad de uso que hoy incorpora. Si bien son innegables estos atractivos en dicha herramienta de elevado poder de cálculo, no es menos cierto que la misma posee altos precios de adquisición[4] que la hacen completamente inasequibles para universidades y/o centros de investigación o producción, fundamentalmente del tercer mundo.

Las limitantes que imponen las licencias privativas a las que están sujetas un sinnúmero de herramientas y sistemas operativos, constituyen sobradas razones para pensar en una migración al software libre, como una opción que permitirá la búsqueda de aplicaciones homólogas que faciliten la realización de trabajos similares con buena calidad.

En el caso particular de Cuba, se gestan los primeros pasos para lograr una migración total hacia el software libre en la que juega un papel fundamental la facultad 10 de la Universidad de las Ciencias Informáticas[5], la cual brinda su apoyo en la búsqueda de novedades para la sustitución de los productos informáticos basados en software libre, necesarios hoy en diferentes empresas y organismos, aunque en algunos casos esto signifique el sacrificio de determinadas prestaciones hoy disponibles en las herramientas privadas.

Polémicas como esta, a nivel mundial, han constituido un motor impulsor del desarrollo de aplicaciones libres para el cálculo moderno siendo este el caso de Octave, que brinda un

conjunto de facilidades similares a Matlab y que está al alcance de un mayor número de usuarios.

Gracias al esfuerzo internacional, hoy en día Octave cuenta con varios ambientes de desarrollo integrados que brindan a los usuarios una mayor didáctica de trabajo incluso comparable con el ambiente de Matlab. El proyecto EIDMAT; desarrollado en la Universidad de las Ciencias Informáticas; ha implementado un frontend con las características antes mencionadas que posibilita a los usuarios de Octave contar con un ambiente de trabajo similar al de Matlab, en el cual se pretende la integración de varios asistentes matemáticos para disímiles usos[6]. El nivel de visualización científica que posibilita Octave en nuestros días se ubica muy por debajo del nivel de otras aplicaciones privativas. Octave utiliza fundamentalmente dos graficadores para la visualización de sus resultados Octaviz y Gnuplot; sin embargo estos poseen un ineficiente grado de interactividad. El Gnuplot constituye un graficador independiente que es utilizado con propósito general por otras aplicaciones en varios sistemas operativos[7]. El Octaviz por su parte constituye una aplicación implementada precisamente para la graficación desde Octave[8]. Las gráficas generadas desde Octave con estos dos graficadores, necesitan de conocimientos avanzados de los usuarios para realizar modificaciones de los elementos previamente visualizados y por lo general los usuarios que trabajan con este tipo de aplicaciones son matemáticos y estudiantes universitarios con conocimientos básicos de informática y les resulta engorroso interactuar con las gráficas. Algunas de las opciones más utilizadas por los usuarios en otras herramientas para la visualización de resultados; como pueden ser la graficación de varias funciones en una misma ventana; no funcionan de manera correcta en los sistemas de graficación de Octave o simplemente no existen. Por otra parte, la instalación de Octaviz como paquete adicional provoca que la versión instalada de Octave sufra trastornos en sus funciones básicas [1], ocasionando que algunas de estas no puedan ser utilizadas de forma correcta e imponiendo además incompatibilidades con algunos frontends existentes hoy para Octave.

[1]Esto se debe en lo fundamental a que es una aplicación desatendida por la comunidad

Problema científico

La ineficiente interactividad que poseen los sistemas para la Visualización Científica que actualmente utiliza Octave, impone un engorroso proceso de gestión de propiedades de las gráficas.

Idea a defender

El desarrollo de un graficador interactivo para la Visualización Científica mediante Octave, eliminará lo engorroso del proceso de gestión de propiedades de las gráficas.

Objeto de estudio

Sistemas para la Visualización Científica.

Campo de acción

Sistemas para la Visualización Científica empleados por el asistente matemático Octave.

Objetivo general

Desarrollar un sistema para la Visualización Científica, que permita la graficación interactiva en dos y tres dimensiones de elementos procesados por Octave.

Objetivos específicos

- Caracterizar los graficadores más utilizados por Octave así como las potencialidades de la librería gráfica VTK[2].
- Desarrollar la conexión de Octave con la aplicación externa de graficación.
- Implementar la graficación de funciones en 2D y 3D, la exportación de gráficas en diferentes formatos y la gestión de la información relativa a las gráficas.

[2]Del inglés Visualization ToolKit

- Validar el eficiente funcionamiento de la propuesta.

Resultados esperados

- Obtención de una aplicación independiente y de elevadas prestaciones integrada a Octave que posibilite la graficación en 2D y 3D así como la gestión completa de la información relativa a las gráficas.

Tareas de investigación

1. Estudio y sintetización de las características de los programas que utiliza octave para graficar y las librerías relacionadas enfatizando en la VTK.

2. Elaboración de todos los *.m y los *.oct que permiten la comunicación desde el Octave con la nueva propuesta.

3. Implementación de los métodos que permiten, dados los datos procesados por Octave, visualizar cada tipo de gráfica.

4. Desarrollo de las opciones de modificación relacionadas con las propiedades de las gráficas visualizadas y exportación de estas a diferentes formatos.

5. Definición de los casos de prueba que permitan valorar la eficiente interactividad de la nueva propuesta.

Métodos Científicos

A lo largo del presente trabajo se sintetizan los conceptos fundamentales sobre las herramientas y tecnologías utilizadas; luego de un exhaustivo análisis bibliográfico; poniéndose de manifiesto el método teórico Analítico-Sintético. De manera adicional, se presentan las características relacionadas con la interactividad, deduciéndose que a partir de estas un sistema informático posee una eficiente interactividad; se manifiesta así el método teórico Inductivo-Deductivo en su parte deductiva.

Estructura del documento

El presente documento se estructura en introducción, cuatro capítulos, conclusiones y un grupo de anexos, donde se expone y da cumplimiento de forma progresiva y elocuente a la totalidad de los objetivos propuestos:

- Capítulo 1. Fundamentación teórica: Son analizados los principales conceptos que se utilizarán en el trabajo. Se expone además, un análisis de las principales características de la Visualización Científica vinculando esta terminología a los asistentes matemáticos. Se realiza una síntesis sobre los graficadores más notables que utiliza Octave destacando sus potencialidades y debilidades. Además son analizadas las características relevantes de las tecnologías a usar en la nueva propuesta.

- Capítulo 2. Descripción y análisis de la solución propuesta: Se realiza una síntesis comparativa de las mejores prácticas en cuanto a la conexión que permite Octave con los graficadores que utiliza tomando algunos de estos elementos para el óptimo funcionamiento y extensibilidad de la nueva propuesta. Además, se caracteriza la estructura y funcionamiento que posee la aplicación desarrollada y los elementos que la componen.

- Capítulo 3. Desarrollo ágil de la aplicación Qtoctaviz: En este capítulo se exponen los elementos referidos a los artefactos generados al aplicar la metodología SXP en el desarrollo de la propuesta.

- Capítulo 4. Validación de la propuesta: En este capítulo se presenta la solución de dos ejercicios estableciendo una comparación entre los resultados alcanzados mediante Gnuplot, Matlab y la nueva propuesta. Adicionalmente se plasman los casos de pruebas o test de aceptación a las que fue sometida la aplicación en cada una de las iteraciones.

Capítulo 1

Fundamentación teórica

En este capítulo se hace referencia al concepto de Visualización Científica vinculado a la graficación mediante los asistentes matemáticos. Se analizan las características de los principales graficadores utilizados actualmente por Octave, así como sus principales potencialidades y debilidades. Se caracteriza la librería gráfica VTK que constituye un elemento fundamental en la solución propuesta. Por último se argumenta el empleo de las herramientas y lenguajes empleados en la solución desarrollada.

1.1. Visualización Científica

La Visualización Científica constituye una ciencia emergente la cual desarrolla ideas fundamentales que conduzcan a herramientas generales para aplicaciones reales. Su surgimiento viene dado por el desarrollo que a alcanzado la informática y la aparición de los gráficos por computadoras. Como disciplina la Visualización Científica de Datos se ocupa de la representación interactiva y el análisis de los datos como un medio para ganar en conocimiento y comprensión de sistemas o procesos bajo estudio. Desde una perspectiva computacional, la Visualización Científica es parte de un área más amplia llamada Visualización la cual incluye investigaciones y desarrollos tecnológicos en computación gráfica, procesamiento de imágenes, computación de alto desempeño, entre otras áreas. El objeto de estudio de esta ciencia es la información contenida en los datos científicos o de ingeniería, su visualización y

otros problemas íntimamente relacionados tales como el manejo computacional o el análisis multivariado. Los enfoques desarrollados son generales y su objetivo es hacerlos aplicables a conjuntos de datos de cualquier tamaño, manteniendo entre tanto una gran interactividad. A menudo se desea contar con la posibilidad de disponer de la visualización de datos provenientes de alguna fuente en tiempo real y son varias las disciplinas que se benefician con las técnicas de Visualización Científica. Dentro de las disciplinas vinculadas a esta ciencia podemos citar las siguientes:

- ***Solución de problemas inversos:*** La necesidad de producir una reconstrucción basada en mediciones remotas provenientes de una cierta cantidad de sensores constituye un aspecto común de la mayoría de las modalidades de creación de imágenes. Tal reconstrucción requiere de la solución de un "problema inverso" involucrando algoritmos numéricos complejos y grandes sistemas de ecuaciones. El objetivo de la reconstrucción puede ser estructural, como la anatomía que se obtiene a partir de imágenes médicas con la utilización de Rayos X o Resonancia Magnética, pero el propósito de la solución del problema inverso también puede ser información funcional, tal como la actividad eléctrica o la conductividad.

- ***Dinámica de los fluidos computacional (DFC):*** El interés por la materia posee dos aristas. La primera arista consisten en el deseo de tener capacidades para modelar fenómenos físicos de los fluidos que no pueden ser simulados o medidos fácilmente mediante experimentos físicos; por ejemplo, los sistemas meteorológicos y los vehículos aeroespaciales hipersónicos. En segundo lugar, el deseo de tener capacidad para investigar sistemas físicos de fluidos de un modo menos costoso y más rápido que mediante procedimientos experimentales.

- ***Mecánica computacional (MC):*** Constituye un área de importancia fundamental dentro de la ciencia y la ingeniería computacional que se ocupa del uso de enfoques computacionales para caracterizar, predecir y simular eventos físicos y sistemas ingenieriles gobernados por las leyes de la mecánica.

- ***Solución de ecuaciones diferenciales parciales (EDPs):*** Si un modelo matemático involucra más de una variable independiente y si al menos una de las variables físicas de interés no es constante con respecto al espacio o el tiempo, entonces el modelo matemático corresponderá a una ecuación diferencial parcial por ejemplo para describir un flujo de fluido en general se requiere que las variables físicas de interés, digamos la presión, la densidad y la velocidad, sean dependientes del tiempo y de tres componentes espaciales.

- ***Grandes conjuntos de datos:*** La renderización interactiva de conjuntos de datos arbitrariamente grandes es un problema fundamental en computación gráfica y en Visualización Científica, y una capacidad crítica para muchas aplicaciones reales. La visualización interactiva de grandes conjuntos de datos plantea retos sustanciales.

- ***Sistemas de partículas:*** Constituyen un medio eficiente y preciso para la visualización de elementos finitos de orden superior. El desarrollo e implementación de estos métodos no solo incrementa la eficiencia del componente de simulación, sino que también proporciona interesantes oportunidades para la interacción con el proceso de visualización.

- ***Procesamiento de señales e imágenes:*** El procesamiento y el análisis de imágenes juega un importante papel en muchas aplicaciones de cálculo y representación científica. Los datos de imágenes provienen de una amplia variedad de fuentes, que incluyen los escáneres médicos, los instrumentos científicos y las cámaras digitales. Usualmente es necesario procesar esas imágenes antes que ellas puedan arrojar información útil.

- ***Visualización remota:*** Permite ver los datos desde lejos por ejemplo si los datos son muy complejos para ser visualizados con los recursos informáticos locales, se puede dar la posibilidad de realizar la renderización remotamente en un cluster y ver los resultados interactivamente en una laptop con conexión inalámbrica.

- ***Modelaje y simulación:*** Modelos matemáticos y numéricos para la ingeniería.

Más adelante en la sección 1.3.1 se aborda el tema de la Visualización Científica empleando asistentes matemáticos para el modelaje y la simulación.

1.2. Interactividad.

Vivimos en una época en que parece que todo debería ser "interactivo" y todo cuanto aparece acompañado de este adjetivo adquiere un valor añadido que lo hace más preciado, actual e innovador. La interactividad podría definirse como el término que describe la relación de comunicación entre un usuario/actor y un sistema (informático, vídeo u otro). El nivel de interactividad mide las posibilidades y el grado de libertad del usuario dentro del sistema, así como la capacidad de respuesta de este sistema en relación al usuario, en cualidad y en cantidad; y esta relación se podría poner en paralelo con el esquema de comunicación: emisor, receptor , respuesta. Si se incrementan las posibilidades de interactuar con los productos, siendo cada vez más grande el grado de libertad del usuario a la hora de tomar decisiones en relación a "que hacer o buscar" y "como hacerlo", y se consigue aumentar el grado de eficacia de la aplicación al "obedecer" las instrucciones que de el sujeto, se podrá decir que se incrementa el nivel de interactividad. Las herramientas interactivas constituyen un elemento muy potente que permite reflejar la componente visual subyacente al control automático bajo la abstracción de conceptos matemáticos, así como estimular la intuición de los estudiantes[9]. De esta forma un alumno puede trabajar sobre un problema de forma gráfica y observar como el cambio en un determinado elemento se ve reflejado de forma inmediata en el resto, como si estuviera ante el proceso real. Los objetos interactivos son aquellos que responden a determinados eventos ante las acciones del usuario, por ejemplo: botones y palabras enfatizadas o calientes[10]. Se dice que un software es interactivo cuando responde inmediatamente a las acciones de los usuarios; o sea, permite un diálogo o intercambio de informaciones entre la computadora y los usuarios[11]. Si el usuario va a interactuar con el sistema de forma tal que el pueda elegir la forma de presentación de la información, si se le ofrecen alternativas a elegir por parte del sistema, se dice que el sistema dispone de interactividad[12]. Con el análisis anterior se llega a la conclusión de que para

que un software desarrollado posea una eficiente interactividad el mismo debe contar con las siguientes características:

- Poseer un gran número de objetos interactivos para la realización de las funciones del sistema.
- Conceder al usuario un elevado nivel de intervención-decisión.
- Ofrecer un amplio abanico de opciones de acceso a la información.
- Poseer gran sencillez en el modo de comunicarse.
- Presentar rapidez en la realización de los procesos y soluciones de tareas.

Estas constituyen elementos indispensables a tomar en cuenta durante el proceso de comprobación de la propuesta desarrollada ante la componente de eficiencia en la interactividad.

1.3. Asistentes Matemáticos.

Los asistentes matemáticos constituyen herramientas computacionales utilizadas para la resolución de problemas vinculados a la matemática. Desde el punto de vista de la formación, el uso de un asistente matemático abre la atractiva posibilidad de experimentar con las matemáticas. A veces, la mejor forma de comprender el verdadero alcance de un teorema o la efectividad de un algoritmo es analizar los resultados que se obtienen al variar las hipótesis, condiciones iniciales, etc. Desde un punto de vista efectivo, el dedicar menos tiempo a la realización de cálculos rutinarios permite primar la reflexión y el análisis de los resultados. Importancia relevante adquiere cuando se trata de su utilización en los cálculos aproximados debido a las posibilidades tecnológicas que cada vez son mayores. Un subconjunto de los asistentes matemáticos lo constituyen los de esencia numérica destacándose como el más utilizado a nivel mundial el Matlab[3]. Debido a la limitante que impone la licencia privativa y los altos precios que posee el Matlab, la comunidad internacional se ve en la obligación de

utilizar un homólogo pero de licencia libre siendo el Octave la mejor opción para una sustitución de acuerdo a las características que posee. Las posibilidades gráficas que posibilitan estas herramientas permiten la mejor comprensión de muchos modelos.

1.3.1. Graficación mediante asistentes matemáticos

La visualización de funciones para lograr un mejor entendimiento del proceso que se modela constituye el objetivo principal de un graficador. La mayoría de los asistentes matemáticos brindan la posibilidad a los usuarios de realizar representaciones gráficas de los elementos que procesan. Algunos asistentes poseen graficadores propios, otros como el Octave utilizan los que existen con propósitos generales brindando la posibilidad a los usuarios de representar de forma visual los modelos matemáticos deseados.
En los próximos epígrafes se darán elementos precisos de la utilización de diferentes graficadores mediante Octave.

1.4. GNU Octave.

Octave se puede definir como un lenguaje de alto nivel inspirado en un software comercial MATLAB (MATrix LABoratory). MATLAB estuvo pensado inicialmente para el álgebra numérica lineal (matrices, vectores y sus operaciones), y con el tiempo se le ha sacado partido a esta forma de trabajo. De la misma forma, Octave empezó siendo un software para que los alumnos de Ingeniería Química de las Universidades de Wisconsin-Madison y Texas[13] efectuaran cálculos asociados a reacciones químicas. A partir de ese momento, las contribuciones de los usuarios han hecho evolucionar este software mediante el añadido de librerías y funcionalidades. Ahora, las aplicaciones de Octave ya no se limitan a simple trabajo con matrices y vectores, como una mera calculadora, sino que además de aplicaciones puramente matemáticas o numéricas, es aplicable a múltiples campos de la ciencia e ingenierías entre ellos, el procesamiento de señales (sonido), de imágenes (filtrados, análisis, etc), estadística, geometría, redes neuronales, sistemas de control realimentados y hasta el dibujo vectorial.

Octave es un software de distribución completamente gratuita basado en la filosofía GNU[1], lo cual implica tener acceso al programa y al código fuente del mismo para transformarlo si se desea, sin la necesidad de abonar un solo dólar en el proceso.

Octave fue escrito en C++ usando la librería STL[2] y posee un intérprete de su propio lenguaje que permite ejecutar sentencias de forma interactiva, aunque también posibilita la ejecución por lote, o lo que es lo mismo: la ejecución de acciones sin mediación del usuario. El procesamiento interactivo se realiza a través del lenguaje Octave, un lenguaje interpretado de alto nivel y que soporta gran parte de las funciones de la librería estándar de C y otras similares, las cuales pueden extenderse con el uso de funciones y procedimientos por medio de módulos dinámicos. Entre las funcionalidades que incorpora Octave se pueden mencionar:

- Integrar ecuaciones diferenciales.
- Herramienta para resolver problemas de álgebra lineal.
- Integrar funciones.
- Calcular raíces de ecuaciones no lineales.
- Manipular polinomios.

1.4.1. Graficación en Octave.

Octave no posee un ambiente propio para la representación gráfica de funciones sino que utiliza otras herramientas de propósito general o específico para la graficación de los elementos que procesa. Dentro de los graficadores que utiliza Octave se pueden mencionar:

- ***Grace:*** Programa anteriormente conocido como Xmgr que permite a los usuarios dibujar gráficos en los ejes X y Y. El usuario puede definir la escala, los tipos de marcadores, las etiquetas, el estilo de la línea y los colores. Permite graficación de regresiones polinómicas, splines, obtención de promedios e imágenes en varios formatos[14].

[1]Proyecto nacido en 1984 para desarrollar un sistema operativo similar a UNIX, pero bajo el concepto de software libre. En sí, GNU es un acrónimo recursivo que significa "GNU No es Unix".

[2]Acrónimo del ingles Standard Template Library

- ***PLplot:*** Plataforma de software para la creación de gráficas científicas. Esta librería puede ser usada para la creación de gráficas en los ejes coordenados X y Y, contornos, gráficas 3D de superficies, gráficos de malla, gráficos de barras y de pastel. Múltiples gráficos (de tamaños iguales o diferentes) se puede colocar en una sola página[15].

- ***OctPlot:*** Manejador de gráficas para Octave. Proporciona calidad postcript(TM) y gráficos en pantalla mediante OpenGL[3]. En la actualidad, sólo cuentan con el apoyo de gráficos 2D, incluyendo superficies, objetos de parches y el zoom. OctPlot puede producir pdf, png y jpg con la ayuda de la gs (ghostscript[4])[16].

- ***Gnuplot:*** Constituye una línea de comandos multiplataforma la cual proporciona graficación en 2D y 3D[7].

- ***Octaviz:*** Sistema para la graficación desde Octave utilizando la librería gráfica VTK para la representación de las gráficas en 2D y 3D[8].

Dentro de estas herramientas las más empleadas son Gnuplot y Octaviz por lo que serán objeto de un análisis más profundo en las secciones siguientes, destacando en cada caso sus principales características y principios de funcionamiento.

1.4.1.1. Gnuplot.

Gnuplot es un programa muy flexible para generar gráficas de funciones y datos[7]. Este programa es compatible con los sistemas operativos más populares (Linux, UNIX, Windows, Mac OS X...). El origen de Gnuplot data de 1986 y la última versión, es la 4.2 fechada el 3 de marzo de 2007. Gnuplot puede producir sus resultados directamente en pantalla, así como en multitud de formatos de imagen, como PNG, EPS, SVG, JPEG, etc. Su distribución se realiza bajo licencia de software libre la cual permite la copia y modificación de su código fuente. Gnuplot permite ejecutar varios comandos agrupados en un archivo o se puede ejecutar mediante tuberías con el envío de un archivo ya existente o una secuencia de comandos para

[3]Acrónimo del ingles Open Graphics Library

[4]Programa intérprete por excelencia de documentos en formato PS (y también PDF).

la entrada estándar y es esta funcionalidad la que aprovecha Octave para la graficación de sus funciones. Cuando el usuario envía el comando específico para la graficación de alguna función de Octave mediante Gnuplot se presenta a este una ventana con la función graficada. Una vez que tenemos la ventana con la gráfica representada solo es posible cambiar los valores de las propiedades(color de línea, color de fondo, formato de los marcadores, las rejillas) mediante la consola de Octave a través de comandos, de modo que la gestión de las gráficas representadas carece de una eficiente interactividad. La funcionalidad de representar varias gráficas en una misma ventana cuando se utilizan funciones en 3D trae consigo errores de visualización cuando intentamos rotar las funciones. En estos casos siempre desaparecen de la ventana de visualización la mayoría de las gráficas sin restablecimiento de las mismas luego de terminada la acción.

1.4.1.2. Octaviz.

Octaviz es una herramienta constituida por una serie de ficheros tipo M-file[5] que son introducidos dentro de las carpetas de Octave además de los ficheros fuentes que permiten la conexión de las funciones escritas en los scripts y la librería gráfica VTK. Se trata de un envoltorio que hace que todas las clases VTK sean accesibles desde dentro de Octave utilizando la misma sintaxis orientado a objetos como en C++ o Python. Octaviz también proporciona funciones de alto nivel para la visualización 2D y 3D. Utilizando estas funciones, la mayoría de las tareas comunes de visualización (3D parcelas de superficie, contorno de parcelas, mallas, etc) se pueden lograr sin necesidad de tener conocimientos acerca de VTK. La representación de funciones mediante Octaviz genera gráficas en una ventana con elevados niveles de resolución; brindando la posibilidad de realizar una rotación de la gráfica, un acercamiento y desplazamientos laterales mediante la interacción del ratón y el teclado aprovechando las potencialidades que brinda la VTK. Como mismo sucede en Gnuplot, no es posible tener una interactividad sobre las propiedades de color, marcadores y rejillas de los ejes coordenados entro otras; no permite la representación de varias gráficas en una misma ventana ni la superposición de dos o más representaciones que son los elementos más comunes

[5]Archivo donde se almacenan un gran número de comandos de Octave

en los graficadores más utilizados mundialmente; además, como es un paquete que no se actualiza con periodicidad su instalación provoca que algunas de las funciones de las versiones más recientes de Octave no funcionen de manera correcta adicionando a esto la incompatibilidad con algunos frontends implementados para Octave, por lo que constituyen un hecho que un mayor número de usuarios emplean el Gnuplot antes del Octaviz independientemente de que este último posee una calidad de imagen superior.

1.5. Librerías Gráficas.

Las librerías gráficas constituyen programas que generan imágenes en base a modelos matemáticos, patrones de iluminación, textura etc. En los próximos epígrafes se hará referencia a las principales librerías gráficas que se utilizaron en la solución propuesta, comentando sus potencialidades y usos más frecuentes.

1.5.1. Librería VTK.

VTK es una librería de código abierto bajo la licencia BSD para la visualización de gráficos 2D y 3D y procesamiento de imágenes[17]. Consiste en una librería de clases en C++ incluyendo la posibilidad de una extensión para Java, Python y Tcl. Soporta una amplia variedad de algoritmos de visualización, incluyendo: vectores, escalares, tensores, texturas y métodos volumétricos y técnicas de modelado avanzadas tales como los modelos implícitos, reducciones de polígonos, suavizado de mallas, cortes de contorno y la triangulación de Delaunay[6]. VTK cuenta con un amplio marco de de visualización de la información, posee un conjunto de widgets 3D de interacción, apoya el proceso paralelo y se integra con varias bases de datos sobre la interfaz gráfica de usuario tales como herramientas Qt y Tk. Esta librería es multiplataforma funcionando en Windows, Linux, Mac y Unix. Muchos generosos patrocinadores han apoyado el desarrollo de esta librería dentro de los que se destacan:

- Los Alamos National Lab (LANL).

[6]Se le denomina así por el matemático ruso Boris Nikolaevich Delone quien lo inventó en 1934

- National Library of Medicine (NLM).
- National Alliance for Medical Image Computing (NA-MIC).
- Department of Energy (DOE) ASC Program.
- Sandia National Laboratories.
- Army Research Laboratory (ARL).

La librería VTK es usada en universidades, corporaciones, e instituciones de investigación en todo el mundo. Su uso está limitado a varias áreas científicas como por ejemplo en el campo de la acústica, en la exploración de los datos astronómicos, en la realidad virtual, en proyectos del cuerpo humano entro otros. Dentro de los proyectos más relevantes en que se usa esta librería se destacan los siguientes:

- ***Geocap GeoScience Environment:*** Ofrece una herramienta de desarrollo rápido de aplicación de la investigación geocientífica y el desarrollo. Los bloques de construcción (datos) y su interacción lógica están disponibles en 2D y 3D a través de un alto nivel de lenguaje de scripting
- ***Virtual Creatures:*** El plan es crear una biblioteca de criaturas virtuales diseñados para la enseñanza. Los estudiantes serán capaces de explorar, visualizar, tocar, y cambiar a estas criaturas en formas que son imposibles con animales reales de laboratorios. Los estudiantes serán capaces de aprender sobre la biología, la matemática, la física, la biomecánica, la bioquímica en un entorno de aprendizaje multidisciplinario.
- ***MapInfo:*** Utilizado para la búsqueda de clientes mediante soluciones inteligentes para los negocios incluyendo funcionalidades poderosas en 3D.
- ***Simulating Acoustic Fields:*** El Centro para la Nueva Música y Tecnologías de Audio (CNMAT) de la Universidad de California en Berkeley ha desarrollado un sistema para la simulación en tiempo real y la visualización de los campos acústicos. El propósito del programa es ayudar a diseñar la colocación de los altavoces y el control

de procesamiento de sonido para crear un lugar tan grande como sea posible donde VTK fue utilizada en una aplicación de diseño acústico personalizado.

- ***DuPont on X-Ray Tomography:*** DuPont ha ido cambiando en la investigación de semillas de Arabidopsis desde que su código genético es totalmente conocida. Semillas de tipo salvaje, sin alteraciones genéticas se comparan con las semillas que tienen las alteraciones genéticas realizadas. La modelación de esta situación conllevó a la realización de una película de tres dimensiones utilizando VTK.

- ***National Library of Medicine Visible Human:*** Este proyecto proporciona la tomografía computarizada (TC), imágenes de resonancia magnética (MRI) y las secciones transversales de los humanos. VTK se utiliza para generar los isosuperficies y las representaciones de volumen de los datos.

1.5.2. Librería Qt.

Qt apareció como biblioteca en 1992 desarrollada por Quasar Technologies en aquel entonces, hoy Trolltech. Es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica, como herramientas de consolas y servidores. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizada en varios otros lenguajes de programación a través de enlaces. El API[7] de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL[8], así como uso de XML[9], gestión de hilos, soporte de red, un API multiplataforma unificado para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales[18]. Dentro de las características más importantes de la librería Qt podemos mencionar las siguientes:

- Librería de programación C++.

- Portabilidad a todas las plataformas y sistemas operativos.

[7] Acrónimo del ingles Application Programming Interface
[8] Acrónimo del ingles Structured Query Language
[9] Acrónimo del ingles Extensible Markup Language

- Herramienta de desarrollo integrada.
- Es orientada a objetos, lo que facilita el desarrollo de software.
- Es una librería que se basa en los conceptos de objetos, Señales-Slots y Eventos.

1.6. Herramientas, lenguaje y metodología.

En el desarrollo de un sistema informático es de vital importancia escoger las herramientas, el lenguaje y las metodología a utilizar, pues una buena selección de estos le permitirá al desarrollador realizar una óptima implementación. Para la selección de los mismos se tuvieron en cuenta los siguientes criterios:

- Librerías e IDEs[10] bajo licencias de software libre en cualquiera de sus versiones.
- Librerías orientadas a objeto.
- IDEs de desarrollo que permitan el trabajo con la librería QT y que permitan la programación en C++ y la inclusión de la VTK para su utilización.
- Metodología que permita un desarrollo ágil.

1.6.1. Anjuta.

Anjuta es un versátil entorno integrado de desarrollo (IDE) desarrollado por Naba Kumbar en 1999 para C y C + + en GNU / Linux aunque actualmente admite lenguajes como C#, Perl y Python[19]. Ha sido escrito para GTK[11] / GNOME[12] y cuenta con una serie de avanzadas opciones para la programación, como la gestión de proyectos, asistentes para aplicaciones, un depurador interactivo y un poderoso editor de código fuente con la navegación y resaltado de sintaxis. DevStudio Anjuta ha sido diseñado para ser fácil de manejar pero

[10] acrónimo en inglés de Integrated Development Environment

[11] Del inglés Graphic ToolKit

[12] Acrónimo del inglés GNU Network Object Model Environment, Entorno GNU de Modelado de Objetos en Red

lo suficientemente poderoso como para satisfacer todas las necesidades de la programación. Dentro sus potencialidades encontramos:

- Editor de código fuente.
- Depurador Integrado.
- Asistente de proyecto.
- Extensión para Subversion.
- Interprete de comandos propio.

1.6.2. Qtcreator.

Qtcreator es un IDE multiplataformas para desarrollar aplicaciones en C++ de manera sencilla y rápida creado por Trolltech[20]. Como su nombre lo indica, está basado en la librería Qt y cuenta con las siguientes características principales:

- Editor avanzado para C++.
- Diseñador de formularios (GUI[13]) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador visual.

1.6.3. Lenguaje C/C++.

El lenguaje de programación C fue creado en 1972 por Dennis M.Ritchie como evolución del lenguaje de programación B. Es un lenguaje orientado a la implementación de sistemas operativos, aunque también se utiliza para crear aplicaciones. Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone

[13]Acrónimo del ingles graphical user interface

de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria y a dispositivos periféricos. Entre sus características más importantes se encuentran:

- Un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas.
- Es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado.
- Acceso a memoria de bajo nivel mediante el uso de punteros.
- Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo.
- Tipos de datos agregados (struct) que permiten que datos relacionados se combinen y se manipulen como un todo.
- Interrupciones al procesador con uniones.
- Un conjunto reducido de palabras claves.

Por otra parte C++ es un lenguaje que se comenzó a desarrollar en 1980 creado por B. Stroustrup. Al comienzo era una extensión del lenguaje C que fue denominada C with classes[21]. El nombre C++ hace referencia al carácter del operador incremento de C (++). Ante la gran difusión y éxito que iba obteniendo en el mundo de los programadores, la AT&T[14] comenzó a estandarizarlo internamente en 1987. En la actualidad, el C++ es un lenguaje versátil, potente y de propósito general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones

[14]Acrónimo del ingles American Telephone and Telegraph

del C original. La evolución de C++ ha continuado con la aparición de Java, un lenguaje creado simplificando, algunas cosas de C++ y añadiendo otras, que se utiliza para realizar aplicaciones en Internet. C++ contiene varias modificaciones menores sobre el C original. Normalmente se trata de aumentar la capacidad del lenguaje y la facilidad de programación en un conjunto de detalles concretos basados en la experiencia de muchos años. Como el ANSI C es posterior a los primeros compiladores de C++, algunas de estas modificaciones están ya introducidas en el ANSI C. En cualquier caso, se trata de modificaciones que facilitan el uso del lenguaje, pero que no cambian su naturaleza. Hay que indicar que el C++ mantiene compatibilidad casi completa con C, de forma que el viejo estilo de hacer las cosas en C es también permitido en C++, aunque este disponga de una mejor forma de realizar esas tareas. C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores).
- Identificación de tipos en tiempo de ejecución (RTTI).

1.6.4. Comunicación entre procesos.

La comunicación entre procesos habilita mecanismos para que los procesos puedan intercambiarse datos y sincronizarse. Existen tres formas muy elementales para que dos procesos se comuniquen: el envío de señales para la sincronización, el uso de ficheros ordinarios para el intercambio de datos y la llamada a ptrace (comunicación unidireccional) para que un proceso padre pueda manipular el espacio de direcciones virtuales de su hijo. Las señales no deben considerarse parte de la forma habitual de comunicar dos procesos y su uso debe restringirse a la comunicación de eventos o situaciones excepcionales. Los ficheros ordinarios tampoco son la forma más eficiente de comunicarse, ya que se ve involucrado el acceso a disco que suele ser 3 ó 4 órdenes de magnitud más lento que el acceso a memoria. Con llamada a ptrace sólo el padre puede leer datos del hijo[22].

1.6.5. Hilo de Ejecución.

Es una característica que permite a una aplicación realizar varias tareas a la vez (i.e. de manera concurrente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente. Un hilo es básicamente una tarea que puede ser ejecutada en paralelo con otra tarea[23].

1.6.6. Lenguajes interpretados o de scripting.

Un script o guión es una serie de órdenes que se pasan a un intérprete para que las ejecute. No cumplen la definición de programa porque no son ejecutables por ellos mismos. Un programa se comunica directamente con el sistema operativo mientras que un script lo hace con un intérprete que a su vez envía comandos al sistema operativo. En este proceso de comunicación el programa no es el script, el archivo de código, sino el intérprete que lee línea por línea el código y que no ejecuta la siguiente orden hasta que no ha terminado con la anterior. Esta es la diferencia entre los lenguajes basados en código fuente de los lenguajes de scripting. Los primeros son C, C++, Fortran, Ada, Cobol y Pascal. El código fuente escrito es transformado por un compilador en un archivo ejecutable binario que sólo es capaz de entender el ordenador. Los lenguajes de scripting más conocidos son, en el caso de los lenguajes de uso general, Java, Python y Ruby. La popularidad de Java se debe a su naturaleza de producto comercial muy sencillo de administrar mientras que Python y Ruby son Software Libre; de igual o más calidad pero sin publicidad. Python es un lenguaje basado en la consistencia que ofrece una gran productividad y versatilidad. Ruby es uno de los lenguajes más recientes, su popularidad está aumentando gracias a la aplicación Ruby on Rails orientada al desarrollo de páginas web. Existe una gran variedad en los lenguajes de scripting orientado a matemáticas. Matlab, Maple, Mathematica, Scilab, Octave, Euler, O-Matrix, R o S son lenguajes de scripting siendo los más conocidos Matlab, Mathematica y Maple.

El scripting científico es una gran herramienta que se debe dominar con independencia de la aplicación que lo implementa. A pesar de que en todas posee lineamientos generales, no es menos cierto que se establecen particularidades en cada caso; dicho así, una vez aprendido a usar Matlab no es posible exportar literalmente el conocimiento al empleo del asistente R, orientado a análisis de datos[24].

1.6.7. Lenguaje Octave.

El lenguaje Octave es un lenguaje de scripting científico, potente y bien diseñado, como parte del proyecto GNU su integración en un entorno GNU/Linux es excelente; es por ello que se comunica perfectamente con otros lenguajes de programación, como es el C y C++. Sus principales características son:

- La sintaxis es similar a la utilizada en MATLAB.
- No permite pasar argumentos por referencia. Siempre se pasan por valor.
- No se permiten punteros.
- Soporta gran parte de las funciones de la librería estándar de C.
- Puede extenderse para ofrecer compatibilidad a las llamadas al sistema UNIX.
- El lenguaje está pensado para trabajar con matrices y provee mucha funcionalidad para trabajar con éstas.
- Un conjunto reducido de palabras claves.
- No es un lenguaje de programación orientado a objetos. Por lo tanto, no tiene clases ni objetos.
- Soporta estructuras similares a los "struct"s de C.

1.6.8. Metodologías de desarrollo.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no llevamos una metodología de por medio, lo que obtenemos son clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños de dos o tres meses. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo. Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí si toma sentido el basarse en una metodología de desarrollo, y empezar a buscar cual sería la más apropiada para el desarrollo del software. Las metodologías ágiles forman parte del movimiento de desarrollo ágil de software conocidos anteriormente como metodologías livianas, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto. Se le denomina ágil como la habilidad de responder de forma versátil al cambio para maximizar los beneficios. Intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados. Las Metodologías Ágiles se basan en los principios siguientes:

- Realizar entregas cortas en el tiempo y continuas.
- Dar la bienvenida a los cambios.
- Entregas periódicas y frecuentes que funcionen.
- Los clientes forman parte del equipo de desarrollo.
- Equipo con individuos motivados. Darles para ello el ambiente, apoyo y confianza.
- La comunicación directa es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo. Intenta evitar el teléfono, correos electrónicos, fax, etc.
- La medida principal de progreso es el software que funciona.

- Desarrollo sostenible. Es indispensable que exista paz y armonía en el equipo para que el proyecto tenga éxito.
- Buen diseño y calidad técnica.
- La simplicidad es algo básico.
- Equipos autoorganizados.
- El equipo debe realizar reflexiones periódicamente para plantearse como llegar a ser más efectivo.

1.6.9. XP.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo[25]. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP en "Una explicación de la programación extrema: Aceptar el cambio" sin cubrir los detalles técnicos y de implantación de las prácticas. XP se basa en 4 valores fundamentales:

- Comunicación.
- Sencillez.
- Retroalimentación.
- Valentía.

El objetivo principal que persigue XP es la satisfacción del cliente. Esta metodología fue diseñada para proporcionar el software que el cliente necesita cuando lo necesite. Debe responder de forma rápida a los cambios en las necesidades del cliente, incluso cuando estos cambios se produzcan al final del ciclo de vida de dicho software (simplicidad y realimentación). El segundo objetivo es potenciar al máximo el trabajo en equipo. Tanto los jefes de proyecto, como los clientes y desarrolladores, son parte del equipo encargado de la implementación de software de calidad (comunicación). Esto implicará que los diseños deberán ser claros y sencillos. Y los clientes deberán disponer de versiones operativas cuanto antes para poder participar en el proceso creativo mediante sus sugerencias y aportaciones. Las iteraciones serán radicalmente más cortas de lo que es usual en otros métodos, de manera que nos podamos beneficiar de la retroalimentación tan a menudo como sea posible. "Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando este tiene lugar" [26].

1.6.10. Scrum.

Scrum es un proceso de desarrollo de software iterativo e incremental (una iteración es un ciclo corto de construcción repetitivo) utilizado comúnmente en entornos basado en la metodología ágil de desarrollo de software[27]. Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming. Se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se puede adaptar en tiempo real el producto que se está construyendo a las

necesidades del cliente. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción del cliente. En Scrum, el equipo se focaliza en una única cosa: construir software de calidad. Por el otro lado, la gestión de un proyecto Scrum se focaliza en definir cuales son las características que debe tener el producto a construir (qué construir, qué no y en qué orden) y en remover cualquier obstáculo que pudiera entorpecer la tarea del equipo de desarrollo. Se busca que los equipos sean lo más efectivos y productivos que sea posible. Scrum tiene un conjunto de reglas muy pequeño y muy simple y está basado en los principios de inspección continua, adaptación, auto-gestión e innovación. El cliente se entusiasma y se compromete con el proyecto dado que ve crecer el producto iteración a iteración y encuentra las herramientas para alinear el desarrollo con los objetivos de negocio de su empresa. Por otro lado, los desarrolladores encuentran un ámbito propicio para desarrollar sus capacidades profesionales y esto resulta en un incremento en la motivación de los integrantes del equipo.

1.6.11. SXP.

SXP está compuesta por las las mejores prácticas de las metodologías XP y Scrum constituyendo un híbrido Cubano, propuesta en el 2008 por la ingeniera Gladys Marsi Peñalver Romero; aprobada y puesta en práctica en varios de los proyectos que trabajan con Software Libre, en la Facultad 10 de la Universidad de la Ciencias Informáticas[28].
SXP es especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos. Donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una elevada flexibilidad. Se utiliza SCRUM como forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos, de forma que sepamos por dónde andamos, mientras que de XP se aprovecha su concepción de estar encaminada al desarrollo. Consiste en una programación extrema, cuya peculiaridad es tener como parte del equipo al usuario final. Las 4 fases definidas para SXP son:

- Planificación-Definición: donde se establece la visión, se fijan las expectativas y se

realiza el aseguramiento del financiamiento del proyecto.

- Desarrollo: es donde se realiza la implementación del sistema hasta que este listo para ser entregado.
- Entrega: puesta en marcha.
- Mantenimiento: donde se realiza el soporte para el cliente.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la lista de reserva del producto (LRP), definición de las historias de usuario (HU), diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad.

Selección de la metodología a utilizar

A partir del análisis anteriormente expuesto, se determina que SXP guiará el desarrollo del software, debido a las características que presenta el equipo de desarrollo; donde se cuenta con un pequeño grupo de desarrolladores, que trabajan en pares y donde el objetivo fundamental es desarrollar un producto tangible, dejando de lado un poco en exceso de documentación generada.

1.7. Conclusiones

En este capítulo se presentaron los aspectos relacionados con los elementos fundamentales sobre la Visualización Científica; que constituye el objeto de estudio del trabajo y las librerías, herramientas, lenguajes y metodologías empleadas en la solución propuesta. Se realizó un análisis deductivo sobre los elementos a tener en cuenta para clasificar una aplicación como eficiente en términos de interactividad. Se caracterizaron además los graficadores

más utilizados por Octave así como las potencialidades de la librería gráfica VTK dando cumplimiento al primero de los objetivos específicos.

Capítulo 2

Descripción y análisis de la solución propuesta

En este capítulo se describen las principales funcionalidades con que cuenta la propuesta. Primeramente se plasman los argumentos de la selección de la librería VTK, luego se realiza una descripción del proceso de conexión desde Octave hacia otros graficadores y se plantea la estrategia de conexión desde Octave hacia la nueva propuesta. Posteriormente se ejemplifica cada funcionalidad implementada en Qtoctaviz que permite la graficación y posterior gestión de las funciones a partir de los elementos procesados por Octave.

2.1. Influencia de Octaviz en la nueva propuesta

Luego de un estudio realizado sobre el funcionamiento de la aplicación Octaviz se llegó a la conclusión de que teniendo el código fuente de este programa, se puede contar con los algoritmos de cada una de las funciones que permite graficar Octaviz, lo que constituyó un factor crucial en la selección de la librería gráfica VTK para la representación de las funciones matemáticas, ya que esto constituye un ahorro en tiempo de implementación, además los desarrolladores no necesitan poseer conocimientos avanzados en cuanto a diferentes términos matemáticos necesarios para la representación de varias funciones. Teniendo en cuenta que la librería VTK-5.0 se puede integrar a un programa utilizando la librería gráfica Qt4, se decidió que toda la interactividad con los botones y ventanas de la nueva aplicación propuesta fueran implementadas con Qt4. Como la aplicación Octave se puede encontrar en varias pla-

taformas, las librerías gráficas seleccionadas y el lenguaje de programación utilizado también poseen estas características.

2.2. Comunicación desde GNU Octave a Gnuplot

Gnuplot es un programa muy flexible para generar gráficas de funciones y datos, este es utilizado por Octave para la visualización de sus resultados por ser un graficador que brinda al usuario un gran número de prestaciones, con el objetivo de representar y modificar distintos tipos de gráficas. Gnuplot no es un graficador desarrollado específicamente para Octave sino que constituye una herramienta de propósito general que es utilizada por otros programas mediante el envío de comandos y esta es la funcionalidad que aprovecha Octave. La comunicación desde Octave a Gnuplot se basa en una serie de ficheros *M* denominados scripts que se encuentran en /usr/share/octave/3.0.1/m/plot/ en dependencia de la versión de Octave instalada. Al llamar uno de estos scripts, mediante Octave se validan los datos procesados y se realiza la ejecución de los archivos precompilados *__contorc__.oct* y *__gnuplot__raw__.oct*, entre otros; que se encuentran en /usr/lib/octave/3.0.1/oct/i486-pc-linux-gnu/ y son los encargados de abrir una tubería para el posterior envío de comandos a Gnuplot conteniendo los valores a graficar. Una vez graficada una función existen otros archivos como por ejemplo *axis.m* y *grid.m* que permiten la modificación de algunos valores representados. Este tipo de comunicación entre Octave y Gnuplot permite una buena validación de los datos y un buen funcionamiento de Octave sin embargo todo el proceso se realiza mediante comandos desde la propia consola.

2.3. Comunicación desde GNU Octave a Octaviz

Octaviz constituye uno de lo graficadores más utilizados por Octave para realizar gráficas, tanto en 2D como en 3D. La comunicación de este con Octave se realiza a través de una serie de ficheros *M* denominados scripts. Estos scripts son introducidos en el núcleo de Octave mediante una API, que posibilita el flujo de datos desde Octave a Octaviz. La ubi-

cación de la API dentro Octave es la siguiente: /usr/share/octave/site/api-v32/m/octaviz. La llamada de uno de estos scripts, desde Octave le permite al mismo la visualización de sus datos mediante la librería gráfica VTK, pues cada script está especializado en un tipo de gráfica, el cual se encarga de obtener los datos procesados por Octave y de realizar la llamada a las clases de VTK relacionadas con la función requerida. En el proceso de instalación de Octaviz las clases VTK son convertidas en librerías de Octave y ubicadas en la dirección /usr/lib/octave/site/oct/api-v32/i486-pc-linux-gnu/octaviz. Esta comunicación entre Octave con Octaviz está muy bien diseñada permitiendo una rápida obtención de las gráficas, además permite una buena validación de los datos introducidos.

2.4. Comunicación entre la solución propuesta y GNU Octave

Qtoctaviz constituye una aplicación independiente, que aprovecha las mejores prácticas en cuanto a funcionamiento de Octaviz y Gnuplot. La conexión de este con GNU Octave es muy similar a la empleada por Gnuplot e incluye elementos de la filosofía de Octaviz (véase la figura 2.1). Para el envío de datos desde Octave a Qtoctaviz se tienen en cuenta una serie de ficheros *M* denominados scripts que se encuentran en /usr/share/octave/site/m/qtoctaviz. Estos scripts no son más que funciones en lenguaje Octave encargados de validar los datos procesados. A su vez los scripts pasan estos datos ya validados a una librería precompilada; ubicada en /usr/lib/octave/3.0.1/oct/i486-pc-linux-gnu/, la cual se encarga de ejecutar el Qtoctaviz en segundo plano mediante el uso de hilos de ejecución. Los datos son enviados a un fichero que es leído constantemente por Qtoctaviz, a través del uso de hilos de ejecución, con el objetivo de representar nuevas funciones. Una vez leídos estos datos, Qtoctaviz utilizará para la visualización de la gráfica las funciones de la librería VTK y mediante componentes Qt se realizará la posterior gestión de mismas.

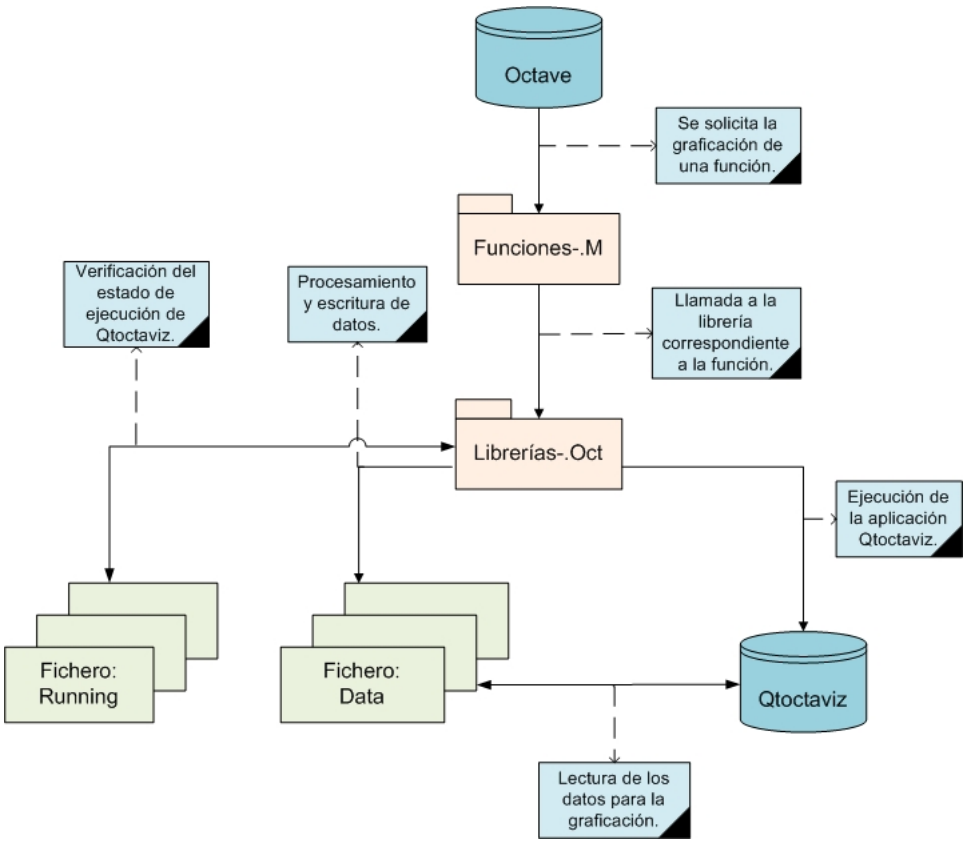


Figura 2.1: Diagrama de comunicación de Octave con Qtoctaviz

2.5. Funcionalidades del sistema

El sistema desarrollado permite la representación gráfica de diferentes funciones desde Octave así como una interactividad a partir de botones y otros componentes, mediante operaciones con el ratón directamente sobre las gráficas y a través de combinaciones de teclas del teclado. Las funciones representativas implementadas son las siguientes:

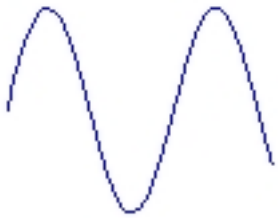
- vtk_plot.
- vtk_plot3.
- vtk_line3.
- vtk_arrows3.

- vtk_contour.
- vtk_mesh.
- vtk_meshc.
- vtk_poly3.
- vtk_quiver3.
- vtk_surf.
- vtk_surfc.
- vtk_trimesh.
- vtk_trisurf.
- vtk_hold.

A continuación se presentan las características significativas de cada tipo de función.

vtk_plot

Traza un típico gráfico de línea (ver figura 2.2). Debe ser pasado por parámetro un vector x de entrada de datos y otro vector y que grafica contra los valores de x. Si el vector y es omitido entonces x se grafica frente a sus índices. Los argumentos opcionales constituyen elementos de formato de línea. La forma de ejecución desde Octave es mediante el comando **vtk_plot(x, y, [prop, val])**. La propiedad representada por ***prop*** constituye el elemento *LineColor* representando el color de la gráfica. Los valores de *LineColor* pueden especificarse en dos formatos, mediante vectores de números binarios de tres elementos o mediante un caracter que representa el color.



```
x = 1:0.2:10;

vtk_plot(x,sin(x),´LineColor´,´b´);
```

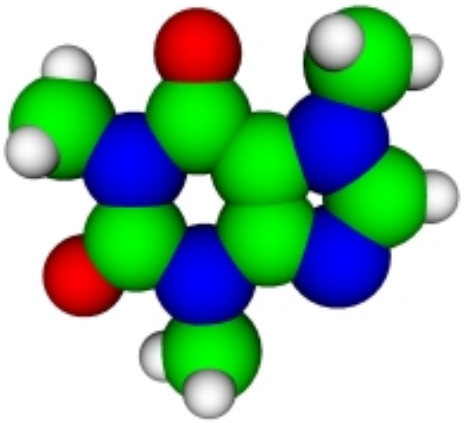
Figura 2.2: Representación de una función seno mediante el comando vtk_plot

vtk_plot3

Grafica parcelas de una nube de puntos en 3 dimensiones (ver figura 2.3). Los puntos de coordenadas de datos representados por los vectores *x*, *y* y *z* dados, deben tener la misma longitud. Los argumentos opcionales constituyen elementos de formato de línea, color y opacidad. La forma de ejecución desde Octave es mediante el comando **vtk_plot3(x, y, [fmt |prop, val])** donde ***fmt*** representa la especificación de la línea por ejemplo la opción "*" posibilita la graficación en forma de esferas. Por otra parte ***prop*** representan los elementos *FigureColor*, *FigureSize* y *Opacity* los cuales representan el color de la figura, su tamaño y el valor de la opacidad respectivamente. Los valores representados por ***val*** dependen del tipo de propiedad, en el caso de *FigureColor* puede especificarse en dos formatos, mediante vectores de números binarios de tres elementos o mediante un caracter que representa el color, mientras que para *FigureSize* se representa por un número decimal al igual que el valor de la propiedad *Opacity*.

vtk_line3

Grafica líneas de segmentos en 3 dimensiones entre los puntos especificados (ver figura 2.4). Los puntos de datos de coordenadas constituyen las matrices *x*, *y* y *z* las cuales deben poseer el mismo tamaño. Los argumentos opcionales constituyen elementos de formato como son el radio de la línea, el color y la opacidad. La forma de ejecución desde Octave es mediante el comando **vtk_line3(x, y, [fmt |prop, val])** donde ***fmt*** representa la especificación de la línea por ejemplo *"r"* representa que la línea sea graficada de color rojo.



```
o = [30 62 19;8 21 10];
n = [31 21 11;18 42 14;55 46 17;56 25 13];
c = [5 49 15;30 50 16;42 42 15;43 29 13;18 28 12;32 6 8;63 36 15;59 60 20];
h = [23 5 7;32 0 16;37 5 0;73 36 16;69 60 20;54 62 28;57 66 12;6 59 16;1 44
22;0 49 6];
vtk_plot3(o(:,1),o(:,2),o(:,3),'FigureSize', 8, 'FigureColor', [1 0 0]);
vtk_hold("on")
vtk_plot3(n(:,1),n(:,2),n(:,3),'FigureSize',10, 'FigureColor', [0 0 1]);
vtk_plot3(c(:,1),c(:,2),c(:,3),'FigureSize',10, 'FigureColor', [0 1 0]);
vtk_plot3(h(:,1),h(:,2),h(:,3),'FigureSize', 5, 'FigureColor', [1 1 1]);
```

Figura 2.3: Representación tridimensional mediante el comando vtk_plot3

Por otra parte ***prop*** incluye los elementos *Color*, *Radius* y *Opacity* los cuales representan el color de la línea, su radio y el valor de la opacidad respectivamente. Los valores representados por ***val*** dependen del tipo de propiedad por ejemplo en el caso de la propiedad *Color* es un vector de tres componentes en formato binario(0 o 1) o un caracter, en el caso de *Radius* se representa por un número decimal al igual que el valor de la propiedad *Opacity*.

```
nmeridian = 6; nlongitude = 11;
phi = 0:pi/1000:2*pi;
mu = phi * nmeridian;
x = cos(mu) .* (1 + cos(nlongitude*mu/nmeridian) / 2.0);
y = sin(mu) .* (1 + cos(nlongitude*mu/nmeridian) / 2.0);
z = sin(nlongitude*mu/nmeridian) / 2.0;
vtk_line3(x',y',z','Radius',0.05);
```

Figura 2.4: Representación espacial de una función mediante el comando vtk_line3

vtk_arrows3

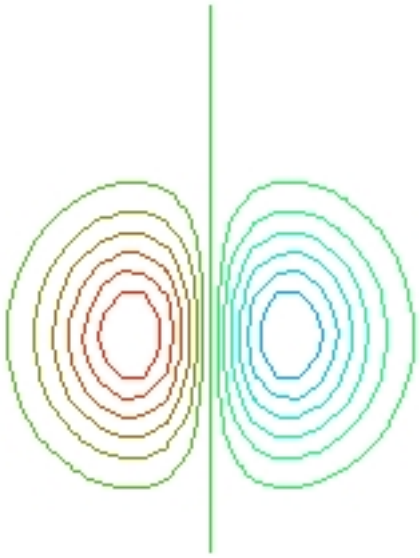
Son visualizados una serie de conos orientados en los puntos definidos por tríos (x, y, z) (ver figura 2.5). Los conos se orientan a lo largo de los vectores definidos por los tríos (nx, ny, nz). La forma de ejecución desde Octave es mediante el comando **vtk_arrows3(x, y, z, nx, ny, nz, [fmt |prop, val])** donde ***fmt*** representa la especificación de la línea por ejemplo *"r"* representa que la flecha sea graficada de color rojo. Por otra parte ***prop*** incluye los elementos *Color*, *TipLength* y *ShaftRadius* los cuales representan el color de la figura, el largo de cabeza de la flecha y el radio del cuerpo de la flecha respectivamente. Los valores representados por ***val*** dependen del tipo de propiedad por ejemplo en el caso de la propiedad *Color* es un vector de tres componentes en formato binario(0 o 1) o un caracter, en el caso de *TipLength* se representa por un número decimal al igual que el valor de la propiedad *ShaftRadius*.

vtk_arrows3(0,5,-2,5,3,-5,'TipLength',0.4,'ShaftRadius',.04);

Figura 2.5: Representación de un vector en el espacio mediante el comando vtk_arrows3

vtk_contour

Muestra un gráfico de contorno definido por las matrices x, y, z donde x y y son formadas típicamente por meshgrid (ver figura 2.6). La forma de ejecución desde Octave es mediante el comando **vtk_contour(x, y, z [prop, val])** donde ***prop*** incluye el elemento *NConts* el cual representa el número de parcelas representado por un número entero.



```
[x,y]=meshgrid(-2:.2:2,-2:.2:3);
z=x.*exp(-x.^2-y.^2);
vtk_contour(x,y,z,ŃConts',15)
```

Figura 2.6: Representación de contornos mediante el comando vtk_contour

vtk_mesh

Muestra un gráfico de malla definida por las matrices x, y, z donde x y y son formadas típicamente por meshgrid (ver figura 2.7). La forma de ejecución desde Octave es mediante el comando **vtk_mesh(x, y, z [prop, val])** donde ***prop*** incluye los elementos *BallRadius* y *LineRadius* los cuales representan el radio de las esferas y las líneas respectivamente. Tanto la propiedad *BallRadius* como *LineRadius* constituyen valores decimales.

```
[u,v] = meshgrid(0.01:pi/20:pi,0:pi/20:2*pi);
x = ( 1 - cos(u) ) .* cos( u + 2*pi/3 ) .* cos( v + 2*pi/3 ) / 2;
y = ( 1 - cos(u) ) .* cos( u + 2*pi/3 ) .* cos( v - 2*pi/3 ) / 2;
z = cos( u - 2*pi/3 );
vtk_mesh(x,y,z);
```

Figura 2.7: Representación de un mallado mediante el comando vtk_mesh

vtk_meshc

Muestra tanto un gráfico de malla como un gráfico de contorno definidos por las matrices x, y, z donde x y y son formadas típicamente por meshgrid (ver figura 2.8). La forma de ejecución desde Octave es mediante el comando **vtk_meshc(x, y, z [prop, val])** donde ***prop*** incluye el elemento *NConts* el cual representa el número de parcelas.

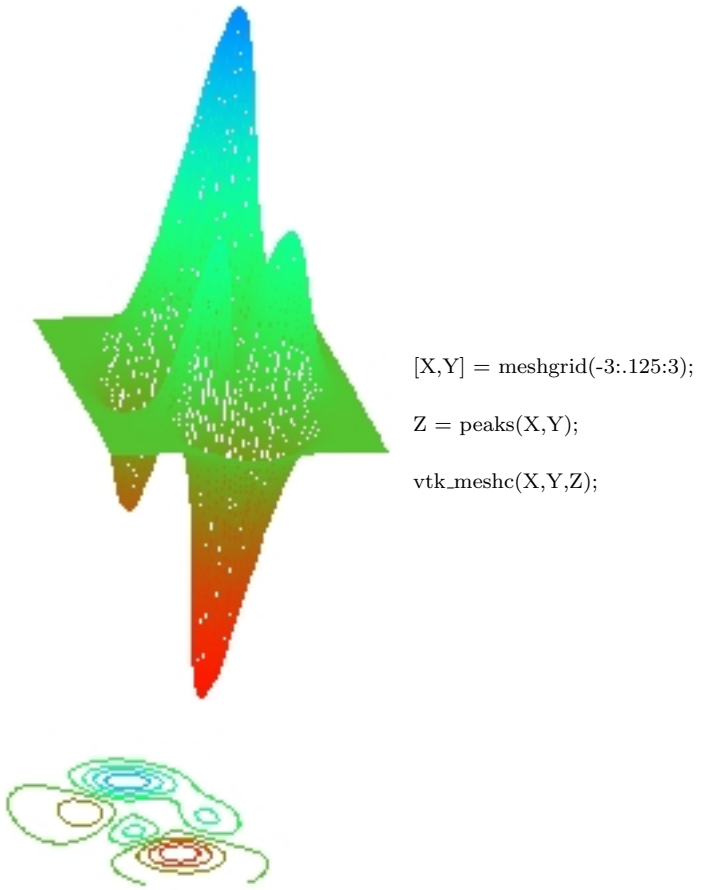


Figura 2.8: Representación de un mallado con contornos mediante el comando vtk_meshc

vtk_poly3

Representa los puntos especificados como un polígono de tres dimensiones (ver figura 2.9). Los puntos de coordenadas de datos deben ser vectores x, y y z de igual longitud. La

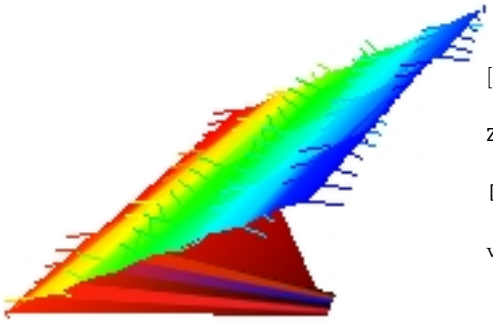
forma de ejecución desde Octave es mediante el comando **vtk_poly3(x, y, z [fmt |prop, val])** donde el parámetro **fmt** representa la especificación de la línea por ejemplo *"r"* indica que el polígono sea graficado de color rojo. Por otra parte ***prop*** incluye los elementos *Color* y *Opacity* los cuales representan el color y la opacidad de la figura. Los valores representados por ***val*** dependen del tipo de propiedad por ejemplo en el caso de la propiedad *Color* es un vector de tres componentes en formato binario(0 o 1), en el caso de *Opacity* toma valor decimal entre 0 y 1.

```
x=[1,2.8,3,3.5,5.1];
y=[5,3,2.7,0.5,2.3];
z=[-1,2,1.5,3,4];
vtk_poly3(x,y,z,¨r¨)
```

Figura 2.9: Representación de polígonos en 3D mediante el comando vtk_poly3

vtk_quiver3

Representa la superficie definida por la matriz z y líneas de segmento desde la superficie definida por las matrices **(u, v, w)** (ver figura 2.10). La forma de ejecución desde Octave es mediante el comando **vtk_quiver3(z, u, v, w)**.

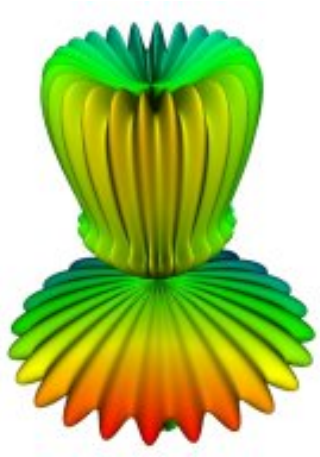


```
[X,Y] = meshgrid(-2:0.25:2,-1:0.2:1);
Z = X.* exp(-X.^2 - Y.^2);
[U,V,W] = surfnorm(X,Y,Z);
vtk_quiver3(X,Y,Z,U,V,W,0.5);
```

Figura 2.10: Representación de campos vectoriales mediante el comando vtk_quiver3

vtk_surf

Muestra un gráfico de superficie definido por las matrices x, y, z donde x y y son formadas típicamente por meshgrid (ver figura 2.11). La forma de ejecución desde Octave es mediante el comando **vtk_surf(x, y, z [prop, val])** donde ***prop*** incluye el elemento *Opacity* el cual representa la opacidad de la figura. El valor de la opacidad estará definido por un número decimal entre 0 y 1.

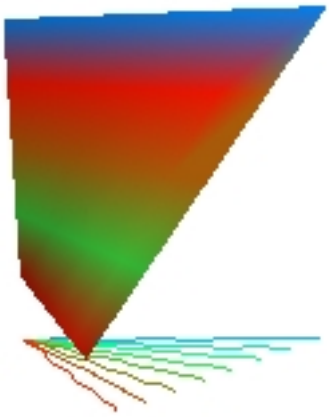


```
[phi,theta] = meshgrid(0:pi/250:pi,0:pi/250:2*pi);
m0 = 4; m1 = 3; m2 = 2; m3 = 3; m4 = 6; m5 = 2; m6 = 6; m7 = 4;
r = sin(m0*phi).^m1 + cos(m2*phi).^m3 + sin(m4*theta).^m5 +
cos(m6*theta).^m7;
x = r .* sin(phi) .* cos(theta);
y = r .* cos(phi);
z = r .* sin(phi) .* sin(theta);
vtk_surf(x,y,z);
```

Figura 2.11: Representación de superficies mediante el comando vtk_surf

vtk_surfc

Muestra tanto un gráfico de superficie como un gráfico de contorno definidos por las matrices x, y, z donde x y y son formadas típicamente por meshgrid (ver figura 2.12). La forma de ejecución desde Octave es mediante el comando **vtk_surfc(x, y, z [prop, val])** donde ***prop*** incluye el elemento *NConts* el cual constituye el número de parcelas representado por un número entero.

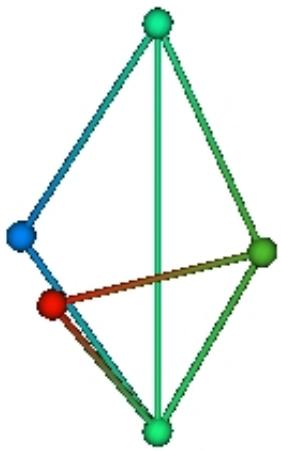


```
[a,b]=meshgrid(0:0.25:1,0:0.5:2);
x=sin(a);
y=sin(a).*b;
z=b;
c=[0,0,0,0,0;2,2,2,2,2;1,1,1,1,1;0,0,0,0,1;4,4,4,4,4];
vtk_surfc(x,y,z,c,ÑConts',10)
```

Figura 2.12: Representación de superficies con contornos mediante el comando vtk_surfc

vtk_trimesh.

Muestra un gráfico de mallas definido por los triángulos especificados (ver figura 2.13). El parámetro t constituye una lista de los índices de vértices para la construcción de los triángulos. Los vectores x, y y z definen los vértices de la figura. La forma de ejecución desde Octave es mediante el comando **vtk_trimesh(t, x, y, z, [prop, val])** donde ***prop*** incluye los elementos *BallRadius* y *LineRadius* los cuales representan el radio de las esferas y las líneas respectivamente. Tanto la propiedad *BallRadius* como *LineRadius* toman valores decimales.



```
x=[1.5 2.1 1.4 3.6 2.8];
y=[2.5 4.1 1.9 3.1 3.8];
z=[3.1 6.4 1.3 2.8 4.1];
t = delaunay(x,y);
vtk_trimesh(t,x,y,z,'BallRadius',0.2, 'LineRadius', 0.05);
```

Figura 2.13: Representación de una malla simple triangular mediante el comando vtk_trimesh

vtk_trisurf.

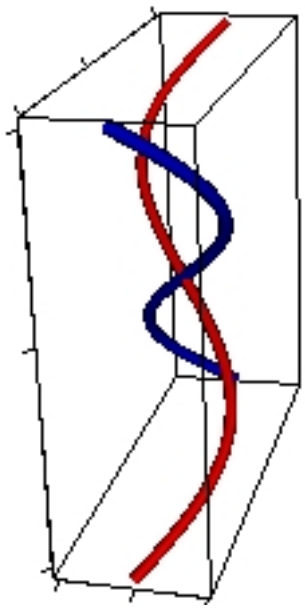
Muestra un gráfico de superficie definido por los triángulos especificados (ver figura 2.14). El parámetro t constituye una lista de los índices de vértices para la construcción de los triángulos. Los vectores x, y y z definen los vértices de la figura. La forma de ejecución desde Octave es mediante el comando **vtk_trisurf(t, x, y, z, [prop, val])** donde ***prop*** está determinado por el elemento *Opacity* el cual representa la opacidad de la figura la cual toma valor decimal entre 0 y 1.

```
x=[1.5 2.1 1.4 3.6 2.8];
y=[2.5 4.1 1.9 3.1 3.8];
z=[3.1 6.4 1.3 2.8 4.1];
t = delaunay(x,y);
vtk_trisurf(t,x,y,z);
```

Figura 2.14: Representación de una superficie triangulada mediante el comando vtk_trisurf

vtk_hold.

Mediante la opción vtk_hold se pueden representar varias funciones en un mismo eje de coordenadas (ver figura 2.15). La forma de ejecución desde Octave es mediante el comando **vtk_hold(val)** donde ***val*** toma valor *on* u *off* en dependencia de la necesidad del usuario. Cuando el usuario introduce la opción **vtk_hold(¨on¨)** luego de graficada una función, las nuevas gráficas serán introducidas en el eje de coordenadas activo en la aplicación hasta que se introduzca la opción **vtk_hold(¨off¨)**.



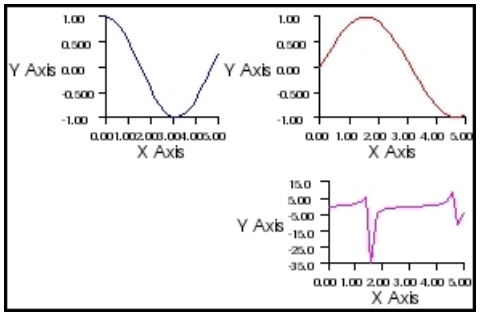
```
x=0:pi/1000:2*pi;

y=sin(x);

z=x;

z2=2*pi:-pi/1000:0;

vtk_line3(x',y',z','r','Radius',.09)

vtk_hold(¨on¨)

vtk_line3(x',y',z2','b','Radius',.1)

vtk_hold(¨off¨)
```

Figura 2.15: Representación de varias funciones mediante el empleo del comando vtk_hold

vtk_subplot.

Mediante la opción vtk_subplot se puede representar una gráfica en solo una porción de la ventana ocupando un espacio determinado por los valores dados en la opción vtk_subplot (ver figura 2.16). Esta funcionalidad es aprovechada cuando el usuario desea representar varias gráficas en una misma ventana pero cada una con su eje de coordenadas independientes. Cuando se intenta representar una gráfica con la opción vtk_subplot y los valores de posición ocupan parte del espacio de otra gráfica anteriormente representada esta última es borrada. Luego de ser representada una gráfica con la opción mencionada el usuario tiene la posibilidad de redimensionar las gráficas por medio de una interacción directa con el ratón sobre la gráfica deseada. La forma de ejecución desde Octave es mediante el comando **vtk_subplot(m,n,p)** seguido de una coma y el comando para la representación de la gráfica deseada, donde m y n representan los valores de una matriz m por n y p constituye la posición en la cual se graficará la figura por ejemplo con los valores **vtk_subplot(2,3,2)**

la pantalla se divide en dos filas y tres columnas y la gráfica se representará en el segundo espacio.



x=0:0.2:5;

vtk_subplot(2,2,2), vtk_plot(x,sin(x))

vtk_subplot(2,2,1), vtk_plot(x,cos(x))

vtk_subplot(2,2,4), vtk_plot(x,tan(x))

Figura 2.16: Representación de varias gráficas en una misma ventana mediante el empleo del comando vtk_subplot

2.6. Opciones de interactividad del sistema

La nueva propuesta brinda a los usuarios varias opciones de interactividad; inexistentes en los graficadores utilizados por Octave; a través de botones interactivos en un sistema de ventanas y desplazamientos con el ratón sobre las gráficas representadas. Mediante el trabajo con la nueva propuesta el usuario tiene la posibilidad de gestionar de forma rápida y eficiente la información relacionada con los ejes coordenados y las funciones graficadas. Los siguientes elementos constituyen características relativas a la interactividad y gestión de la información presentes:

- Posee un gran número de objetos interactivos para la gestión de la información de las gráficas representadas.
- Concede al usuario un elevado nivel de intervención-decisión.
- Ofrece un amplio abanico de opciones de acceso a la información.
- Posee una gran sencillez en el modo de comunicarse con el usuario.
- Posee una gran rapidez en la realización de los procesos y soluciones de tareas.

2.6.1. Opciones del menú

La aplicación cuenta con un menú principal que permite al usuario la gestión de la información de la gráfica activa (la última que el usuario seleccionó) de forma directa.

Menú Archive

Dentro del menú **Archive** se encuentra el submenú **Save as** el cual cuenta con las opciones **PNG Image**, **JPEG Image** y **BMP Image** las cuales permiten la exportación de las figuras visualizadas en la ventana principal en formato, PNG, JPEG y BMP respectivamente (véase la figura 2.17). Otra de las opciones del menú **Archive** lo constituye el elemento **Exit** lo que permite cerrar la aplicación de forma inmediata.

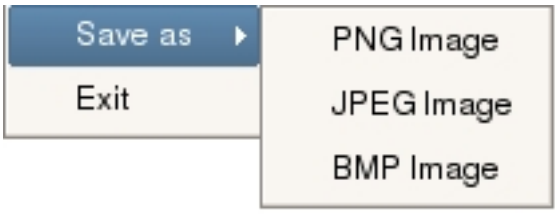


Figura 2.17: Menú Archive

Menú Edit

A través del menú **Edit** (véase la figura 2.18) el usuario puede visualizar tres ventanas acoplables (**Property Editor**, **Grid Editor 3D** y **Plot Browser**) las cuales brindan varias opciones de gestión de información de las gráficas representadas.

Cuando el usuario selecciona la opción **Property Editor** es visualizada una ventana en alguna de sus tres vistas (**Property Editor-Figure**, **Property Editor-Axes** o **Property Editor-Grafic**) en dependencia de la opción de interactividad con la gráfica activa. Si no está seleccionada alguna gráfica de forma activa al habilitar la opción **Property Editor** se muestra la vista **Property Editor-Figure** (véase la figura 2.19) mediante la cual se puede

Figura 2.18: Menú Edit

seleccionar uno de los mapas de colores brindados así como el color de fondo de la ventana que visualiza las gráficas.

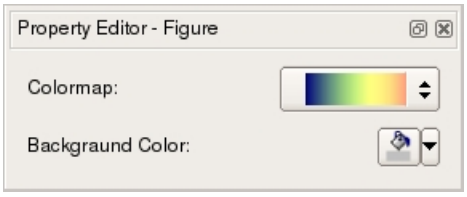


Figura 2.19: Ventana emergente Property Editor-Figure

Cuando se selecciona directamente con el ratón alguna de las gráficas visualizadas y está habilitada la opción **Property Editor** se muestra la vista **Property Editor-Axes** (véase la figura 2.20) la cual contiene toda la información relativa a los ejes coordenados. Las opciones que permite esta vista son las siguientes:

- Introducir un nuevo título a la gráfica activa, o cambiar el que poseía anteriormente.
- Visualizar u ocultar las mallas en los diferentes ejes (x, y, z).
- Introducir o modificar el título de cada uno de los ejes coordenados (x, y, z).
- Modificar los intervalos definidos para cada uno de los ejes coordenados (solo en el caso de las figuras de dos dimensiones).
- Modificar las opciones de visualización de texto tales como la fuente de texto (**Arial**, **Couriel**, **Times**), texto con sombra (**Shadow**), en cursiva (**Italic**) y en negrita (**Bold**).

- Modificación del color de la información relativa a los ejes (color de las líneas que componen los ejes coordenados y el texto).
- Visualizar u ocultar el cubo que engloba la gráfica activa (solo en el caso de las figuras de tres dimensiones).
- Visualizar u ocultar la barra de colores referida a la gráfica activa (solo en el caso de las figuras de tres dimensiones).

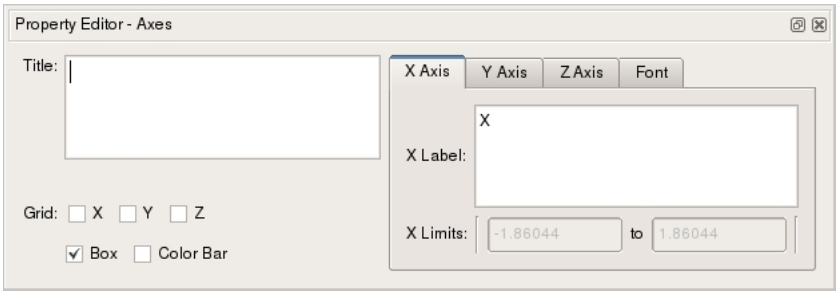


Figura 2.20: Ventana emergente Property Editor-Axes

Cuando se selecciona una de las funciones representadas en uno de los ejes de coordenadas mediante la ventana **Plot Browser** y está habilitada la opción **Property Editor** se muestra la vista **Property Editor-Grafic** (véase la figura 2.21) la cual gestiona el nombre, el color y el tipo de marcador de la figura que compone la función. Los tipos de marcadores son los siguentes:

- **none:** Permite la visualización de la figura sin marcadores.
- **circle:** Permite la visualización de círculos en los índices de la figura.
- **triangle:** Permite la visualización de triángulos en los índices de la figura.
- **square:** Permite la visualización de cuadrados en los índices de la figura.
- **diamond:** Permite la visualización de diamantes en los índices de la figura.
- **arrow:** Permite la visualización de flechas en los índices de la figura.

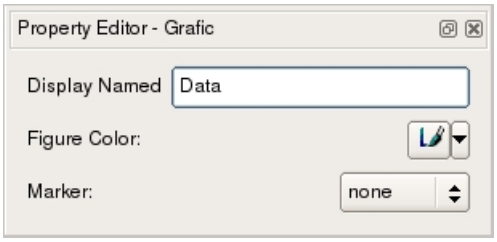


Figura 2.21: Ventana emergente Property Editor-Grafic

Cuando el usuario selecciona la opción **Grid Editor 3D** es visualizada una ventana emergente (véase la figura 2.22) mediante la cual se realiza un corrimiento de las mallas en los diferentes ejes coordenados (solo en gráficas en tres dimensiones) además da la posibilidad de gestionar la opacidad de cada una de estas mallas.

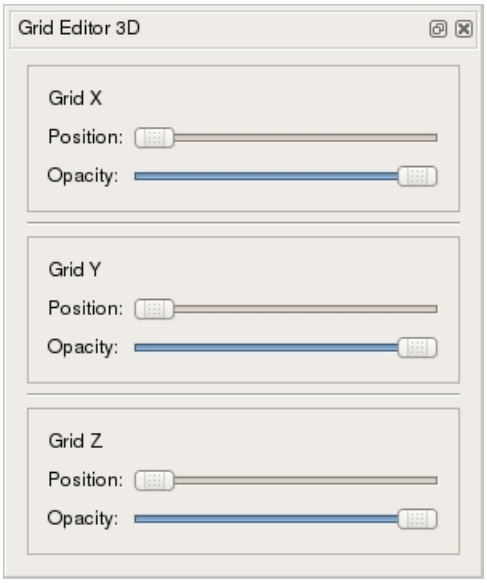


Figura 2.22: Ventana emergente Grid Editor 3D

Cuando el usuario selecciona la opción **Plot Browser** es visualizada una ventana emergente (véase la figura 2.23) la cual permite la modificación de algunas propiedades de las gráficas. Mediante la vista de árbol que proporciona esta ventana se organizan cada una de las gráficas representadas y dentro de estas se encuentran las funciones correspondientes. Cuando se desmarca algunas de las casillas de la vista de árbol perteneciente a una función

esta desaparece de la ventana principal, ocurriendo lo mismo en el caso de las casillas correspondientes a los ejes coordenados de una gráfica. Adicionalmente esta ventana es utilizada para seleccionar la función deseada para la gestión de sus propiedades mediante la ventana **Property Editor**, en el caso de seleccionar un término referente a una gráfica en general se brinda la posibilidad de cambiar las propiedades de los ejes coordenados.

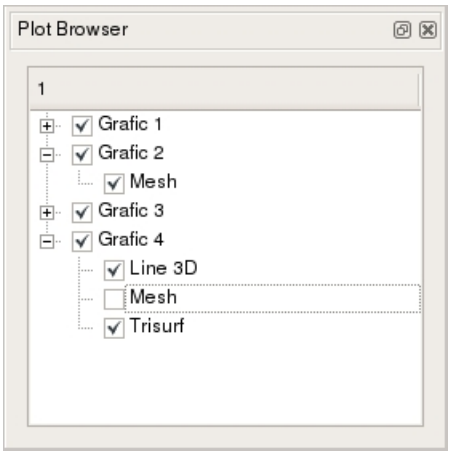


Figura 2.23: Ventana emergente Plot Browser

Menú View

El menú **View** incluye varias opciones de visibilidad de componentes de las gráficas representadas (véase la figura 2.24). Dentro de las opciones de este menú se encuentran los submenús **X Axis**, **Y Axis** y **Z Axis** los cuales posibilitan visualizar u ocultar diferentes elementos de los ejes de coordenadas. El elemento **All** se refiere a todos los componentes del eje seleccionado, el elemento **Label** al título del eje, el elemento **Scalars** a los valores numéricos del eje, el elemento **Ticks** a las líneas marcadoras de los valores numéricos y el elemento **Line** se refiere a la línea del eje. Por otra parte el submenú **Grid** dentro del menú **View**, brinda la posibilidad de visualizar u ocultar alguna de las mallas en los diferentes ejes. Adicionalmente este menú cuenta con los elementos **Title** , **Legend** y **Colorbar**, los cuales representan el título, la leyenda y la barra de colores de la gráfica activa.

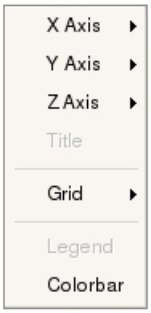


Figura 2.24: Menú View

Menú Insert

Mediante el menú **Insert** el usuario tiene la posibilidad de insertar o modificar el título de la gráfica y los títulos de cada uno de los ejes coordenados. Estas operaciones están dadas por los elementos **Title**, **X Label**, **Y Label** y **Z Label** (véase la figura 2.25) y dependen de la gráfica activa.

Figura 2.25: Menú Insert

2.6.2. Opciones de interactividad directa con el ratón

Una de las opciones que brinda la aplicación propuesta es la interactividad directamente sobre las gráficas a través del ratón, para lo que se cuenta con una barra de herramientas (véase la figura 2.26) con cuatro botones permitiendo cuatro estados de interactividad (**None**, **Pan**, **Rotate 3D** y **Rotate 3D All**). En el caso del estado **None** el usuario tiene una interactividad nula con las gráficas. Cuando se selecciona la opción **Pan**, el usuario puede redimensionar el tamaño de la figura y moverla a través del plano que representa la

ventana principal. A través de la opción **Rotate 3D** se puede seleccionar una gráfica para desarrollar una interactividad en tres dimensiones de forma individual. Al dar clic sobre una gráfica cuando está activa la opción **Rotate 3D** emerge una ventana que incluye todos los elementos visibles de la gráfica señalada, permitiendo ejecutar giros de la gráfica en tres dimensiones, la posibilidad de realizar acercamientos y de eliminar funciones independientes dentro de la gráfica. Las propiedades de los elementos en la ventana emergente para la interactividad individual dependen completamente de las definidas en la ventana principal. La opción **Rotate 3D All** posibilita la rotación al unísono de todas las gráficas en tres dimensiones presentes en la ventana principal.

Figura 2.26: Barra de herramientas

2.7. Conclusiones

En este capítulo fueron presentados los aspectos relacionados con la descripción del funcionamiento y potencialidades de la nueva aplicación. En cumplimiento al objetivo número dos se establece una conexión desde Octave hacia Qtoctaviz, tomando en cuenta las mejores prácticas de otros graficadores utilizados por Octave. Con posteriorida,d se realiza una explicación de las funcionalidades inherentes a las graficaciones en dos y tres dimensiones, la gestión de las gráficas y funciones una vez visualizadas y la exportación de estas en diferentes formatos de imágenes dando cumplimiento así al objetivo número tres.

Capítulo 3

Desarrollo ágil de la aplicación Qtoctaviz

El presente capítulo recoge los elementos más importantes abordados durante el desarrollo ágil del sistema propuesto, utilizando la metodología SXP. Se explicará toda la dinámica del sistema en forma de historias de usuarios, tareas de ingeniería y algunos modelos auxiliares.

3.1. Planificación del proyecto por roles

SXP define diferentes roles para lograr un exitoso resultado en el proceso de desarrollo de software, los roles involucrados en el desarrollo de la nueva propuesta son:

Rol	**Nombre**
Gerente	Alexeis Companioni Guerra
Cliente	EIDMAT
Programadores	Jorge Luis Hernández Cruz Miguel Angel Chavez Alfonso
Analistas	Jorge Luis Hernández Cruz Miguel Angel Chavez Alfonso
Diseñadores	Jorge Luis Hernández Cruz Miguel Angel Chavez Alfonso

Encargado de Pruebas	Vianka Orovio Cobo
Arquitectos	Jorge Luis Hernández Cruz Miguel Angel Chavez Alfonso

Tabla 3.1: Planificación del proyecto por roles

3.2. Historias de Usuarios

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado. Las mismas son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Son utilizadas como el único documento de requisitos que se genera en XP.

Historia de Usuario	
Número: U-QTO-01	**Nombre Historia de Usuario:** Diseño de interfaz de usuario
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: Se diseña la interfaz de usuario que va a tener la aplicación.	
Observaciones:	
Prototipo de interfase:	

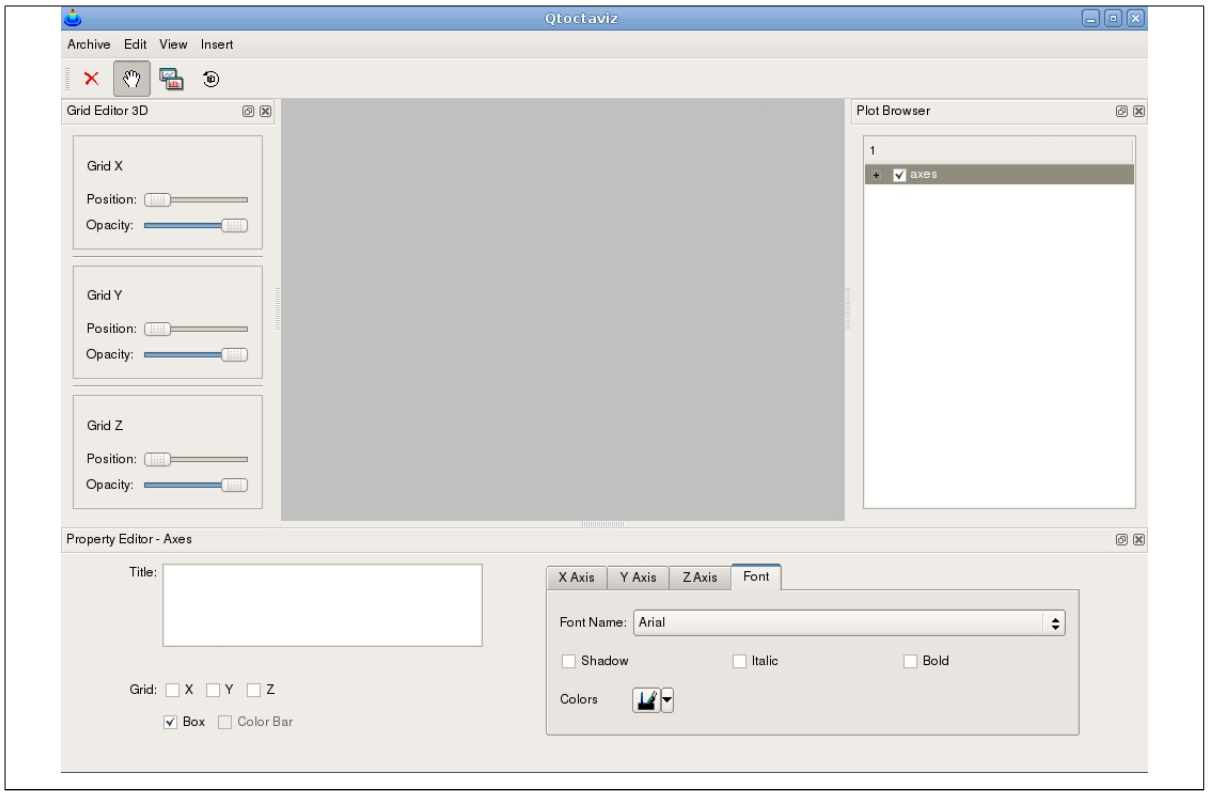


Tabla 3.2: Historia de Usuario U-QTO-01. Diseño de interfaz de usuario

Historia de Usuario	
Número: U-QTO-02	**Nombre Historia de Usuario:** Plot 2D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 3
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 4
Descripción: Se realiza la construcción de una gráfica en 2 dimensiones.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.3: Historia de Usuario U-QTO-02. Plot 2D

Historia de Usuario	
Número: U-QTO-03	**Nombre Historia de Usuario:** Plot 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de una gráfica polar en 3 dimensiones.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.4: Historia de Usuario U-QTO-03. Plot 3D

Historia de Usuario	
Número: U-QTO-04	**Nombre Historia de Usuario:** Arrow 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica en forma de flechas.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.5: Historia de Usuario U-QTO-04. Arrow 3D

Historia de Usuario	
Número: U-QTO-05	**Nombre Historia de Usuario:** Cone 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica formada por conos.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.6: Historia de Usuario U-QTO-05. Cone 3D

Historia de Usuario	
Número: U-QTO-06	**Nombre Historia de Usuario:** Contour 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica de contorno.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.7: Historia de Usuario U-QTO-06. Contour 3D

Historia de Usuario	
Número: U-QTO-07	**Nombre Historia de Usuario:** Line 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica de tipo línea en 3D.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.8: Historia de Usuario U-QTO-07. Line 3D

Historia de Usuario	
Número: U-QTO-08	**Nombre Historia de Usuario:** Poly 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de un polinomio en 3D.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.9: Historia de Usuario U-QTO-08. Poly 3D

Historia de Usuario	
Número: U-QTO-09	**Nombre Historia de Usuario:** Trimesh 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica Trimesh.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.10: Historia de Usuario U-QTO-09. Trimesh 3D

Historia de Usuario	
Número: U-QTO-10	**Nombre Historia de Usuario:** Trisurf 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica Trisurf.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.11: Historia de Usuario U-QTO-10. Trisurf 3D

Historia de Usuario	
Número: U-QTO-11	**Nombre Historia de Usuario:** Surf 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica Surf.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.12: Historia de Usuario U-QTO-11. Surf 3D

Historia de Usuario	
Número: U-QTO-12	**Nombre Historia de Usuario:** Mesh 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica Mesh.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.13: Historia de Usuario U-QTO-12. Mesh 3D

Historia de Usuario	
Número: U-QTO-13	**Nombre Historia de Usuario:** SurfC 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica SurfC.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.14: Historia de Usuario U-QTO-13. SurfC 3D

Historia de Usuario	
Número: U-QTO-14	**Nombre Historia de Usuario:** MeshC 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica MeshC.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.15: Historia de Usuario U-QTO-14. MeshC 3D

Historia de Usuario	
Número: U-QTO-15	**Nombre Historia de Usuario:** Quiver 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la construcción de la gráfica Quiver.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.16: Historia de Usuario U-QTO-15. Quiver 3D

Historia de Usuario	
Número: U-QTO-16	**Nombre Historia de Usuario:** Axes 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Jorge Luis Hernández Cruz	**Iteración Asignada:** 3
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza la gestión de la información relativa a los ejes coordenados representados en la figura visualizada.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.17: Historia de Usuario U-QTO-16. Axes 3D

Historia de Usuario	
Número: U-QTO-17	**Nombre Historia de Usuario:** Salvar Como
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 3
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Se realiza una copia de la imagen visualizada en Qtoctaviz y se archiva en el disco duro en algunos de sus tres formatos (png, jpeg, bmp).	
Observaciones:	
Prototipo de interfase:	

Tabla 3.18: Historia de Usuario U-QTO-17. Salvar Como

Historia de Usuario	
Número: U-QTO-18	**Nombre Historia de Usuario:** Subplot
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 3
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Son visualizadas varias gráficas en una misma ventana.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.19: Historia de Usuario U-QTO-18. Subplot

Historia de Usuario	
Número: U-QTO-19	**Nombre Historia de Usuario:** Hold
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 3
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Son visualizadas varias funciones en una misma gráfica.	
Observaciones:	
Prototipo de interfase:	

Tabla 3.20: Historia de Usuario U-QTO-19. Hold

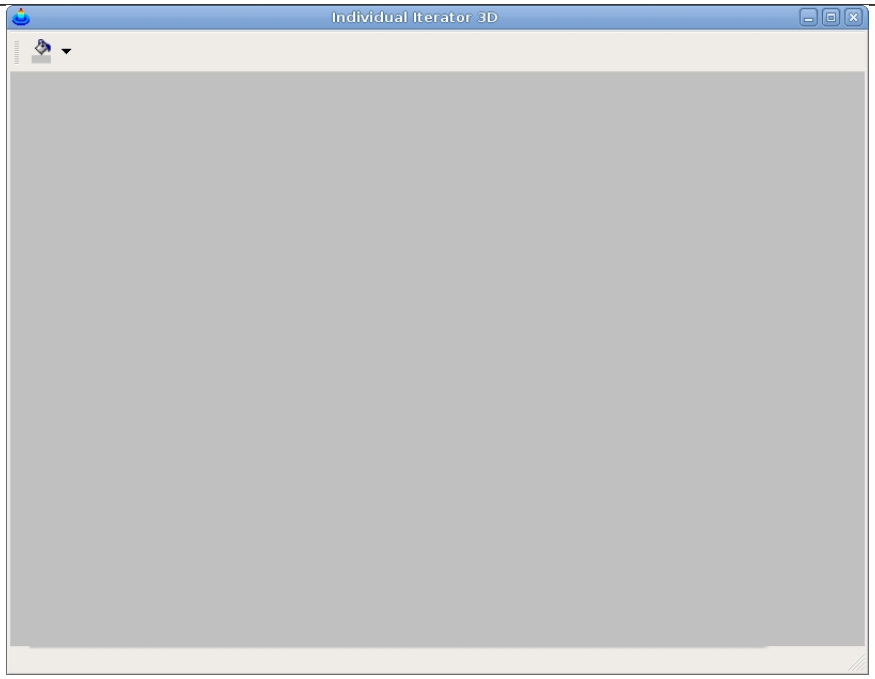
Historia de Usuario	
Número: U-QTO-20	**Nombre Historia de Usuario:** Interacción Individual 3D
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Miguel Angel Chavez Alfonso	**Iteración Asignada:** 3
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: Se visualiza una gráfica de tipo 3D en una nueva ventana	
Observaciones:	
Prototipo de interfase:	
	

Tabla 3.21: Historia de Usuario U-QTO-20. Interacción Individual 3D

3.3. Tareas de Ingeniería

En la fase de desarrollo de la metodología SXP se propone la entrega de las tareas realizadas para dar cumplimiento a cada historia de usuario. Estas tareas son llamadas Tareas de Ingeniería.

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-01
Nombre de la Tarea: Diseñar el prototipo de interfaz de usuario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1,2 semanas
Fecha de Inicio: 20/09/2009	**Fecha de Fin:** 27/09/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se diseña un prototipo de interfaz de usuario no funcional y se crea la estructura general que va a tener la aplicación.	

Tabla 3.22: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-01

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-02
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la graficación en 2D de una línea.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 28/09/2009	**Fecha de Fin:** 14/10/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación en 2 dimensiones desde Octave a Qtoctaviz. Los scripts a desarrollar vtk_plot.m y vtk_plot_2.oct	

Tabla 3.23: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-02

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-02
Nombre de la Tarea: Implementación de la clase VTKPlot.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 28/09/2009	**Fecha de Fin:** 14/10/2009
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas de tipo 2D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.24: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-02

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-03
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la graficación en 3D de una gráfica polar.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 28/09/2009	**Fecha de Fin:** 14/10/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación en 3 dimensiones desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_plot3.m y vtk_plot_3.oct.	

Tabla 3.25: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-03

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-03
Nombre de la Tarea: Implementación de la clase VTKPlot3.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 28/09/2009	**Fecha de Fin:** 14/10/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Gestiona la información de las gráficas polares en 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.26: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-03

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-04
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la graficación de la gráfica Arrows.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 15/10/2009	**Fecha de Fin:** 28/10/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Arrow en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_arrows3.m y vtk_arrows_3.oct.	

Tabla 3.27: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-04

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-04
Nombre de la Tarea: Implementación de la clase VTK_Arrows3.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 15/10/2009	**Fecha de Fin:** 28/10/2009
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Arrows 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.28: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-04

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-05
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Cone.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 15/10/2009	**Fecha de Fin:** 28/10/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Cone en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_cone3.m y vtk_cone_3.oct.	

Tabla 3.29: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-05

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-05
Nombre de la Tarea: Implementación de la clase VTK_Cone3.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 15/10/2009	**Fecha de Fin:** 28/10/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Gestiona la información de las gráficas Cone 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.30: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-05

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-06
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Contour.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 29/10/2009	**Fecha de Fin:** 21/11/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Contour en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_contour.m y vtk_contouer_3d.oct.	

Tabla 3.31: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-06

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-06
Nombre de la Tarea: Implementación de la clase VTK_Contour.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 29/10/2009	**Fecha de Fin:** 21/11/2009
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Contour 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.32: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-06

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-07
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Line.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 22/11/2009	**Fecha de Fin:** 8/12/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación de Line en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_line3.m y vtk_line_3.oct.	

Tabla 3.33: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-07

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-07
Nombre de la Tarea: Implementación de la clase VTKLine3.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 22/11/2009	**Fecha de Fin:** 8/12/2009
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Line 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.34: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-07

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-08
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Polygon.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 8/12/2009	**Fecha de Fin:** 20/12/2009
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación de Poly en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_poly3.m y vtk_poly_3.oct.	

Tabla 3.35: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-08

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-08
Nombre de la Tarea: Implementación de la clase VTK_Poly3.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 8/12/2009	**Fecha de Fin:** 20/12/2009
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Poly 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.36: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-08

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-09
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Trimesh.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 12/01/2010	**Fecha de Fin:** 26/01/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Trimesh en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_trimesh.m y vtk_trimesh_3d.oct.	

Tabla 3.37: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-09

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-09
Nombre de la Tarea: Implementación de la clase VTK_Trimesh.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 12/01/2010	**Fecha de Fin:** 26/01/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Trimesh 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.38: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-09

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-10
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Trisurf.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 12/01/2010	**Fecha de Fin:** 26/01/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Trisurf en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_trisurf.m y vtk_trisurf_3d.oct.	

Tabla 3.39: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-10

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-10
Nombre de la Tarea: Implementación de la clase VTK_Trisurf.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 12/01/2010	**Fecha de Fin:** 26/01/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Trisurf 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.40: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-10

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-11
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Trisurf.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 27/01/2010	**Fecha de Fin:** 13/02/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Trisurf en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_surf.m,vtk_trisurf.m y vtk_trisurf_3d.oct.	

Tabla 3.41: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-11

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-11
Nombre de la Tarea: Implementación de la clase VTK_Trisurf.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 27/01/2010	**Fecha de Fin:** 13/02/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Trisurf 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.42: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-11

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-12
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Mesh.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 27/01/2010	**Fecha de Fin:** 13/02/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Mesh en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_mesh.m,vtk_trimesh.m y vtk_trimesh_3d.oct.	

Tabla 3.43: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-12

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-12
Nombre de la Tarea: Implementación de la clase VTK_Trimesh.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 27/01/2010	**Fecha de Fin:** 13/02/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Trimesh 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.44: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-12

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-13
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Surf C.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 14/02/2010	**Fecha de Fin:** 28/02/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del SurfC en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_surfc.m,vtk_trisurf,vtk_contour.m,vtk_contour_3d.oct y vtk_trisurf_3d.oct.	

Tabla 3.45: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-13

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-13
Nombre de la Tarea: Implementación de la clase VTK_Trisurf, VTK_Contour.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 14/02/2010	**Fecha de Fin:** 28/02/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas SurfC 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.46: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-13

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-14
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica MeshC.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 14/02/2010	**Fecha de Fin:** 28/02/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del MeshC en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_meshc.m,vtk_trimesh.m,vtk_contour,vtk_contour_3d.oct y vtk_trimesh_3d.oct.	

Tabla 3.47: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-14

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-14
Nombre de la Tarea: Implementación de la clase VTK_Trimesh,VTK_Contour.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 14/02/2010	**Fecha de Fin:** 28/02/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Trimesh 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.48: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-14

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-15
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la visualización de la gráfica Quiver.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 14/02/2010	**Fecha de Fin:** 28/02/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación del Trisurf en 3D desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_quiver3.m y vtk_quiver_3d.oct.	

Tabla 3.49: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-15

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-15
Nombre de la Tarea: Implementación de la clase VTK_Quiver3.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 14/02/2010	**Fecha de Fin:** 28/02/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información de las gráficas Quiver 3D permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.50: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-15

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-16
Nombre de la Tarea: Implementación de la clase Axes_3D.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 1/03/2010	**Fecha de Fin:** 14/03/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona la información del eje de coordenadas de las gráficas en 3D representadas permitiendo la modificación de sus propiedades en tiempo de ejecución.	

Tabla 3.51: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-16

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-17
Nombre de la Tarea: Implementación de la funcionalidad salvar como png.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 1/03/2010	**Fecha de Fin:** 14/03/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Exporta la gráfica visualizada como una imagen en formato png.	

Tabla 3.52: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-17

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-17
Nombre de la Tarea: Implementación de la funcionalidad salvar como jpeg.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 1/03/2010	**Fecha de Fin:** 14/03/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Exporta la gráfica visualizada como una imagen en formato jpeg.	

Tabla 3.53: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-17

Tarea de Ingeniería	
Número de Tarea: 3	**Número de Historia Usuario:** U-QTO-17
Nombre de la Tarea: Implementación de la funcionalidad salvar como bmp.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 1/03/2010	**Fecha de Fin:** 14/03/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Exporta la gráfica visualizada como una imagen en formato bmp.	

Tabla 3.54: Tarea de Ingeniería #3 perteneciente a la Historia de Usuario U-QTO-17

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-18
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la graficación de varias gráficas en una misma ventana.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 15/03/2010	**Fecha de Fin:** 31/03/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación de varias funciones en una misma ventana desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_subplot.m y vtk_subplot_2.oct.	

Tabla 3.55: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-18

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-18
Nombre de la Tarea: Implementación de la clase plotPositionProperties.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 15/03/2010	**Fecha de Fin:** 31/03/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Gestiona las posiciones que van a ocupar las gráficas representadas con la opción vtk_subplot.	

Tabla 3.56: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-18

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-19
Nombre de la Tarea: Implementar la conexión de Octave a Qtoctaviz para la graficación de varias funciones en una misma gráfica.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,3 semanas
Fecha de Inicio: 15/03/2010	**Fecha de Fin:** 31/03/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se deben implementar dos scripts para enviar la petición de la graficación de varias funciones en una misma gráfica desde Octave a Qtoctaviz. Los scripts a desarrollar son vtk_hold.m y vtk_hold_axes.oct.	

Tabla 3.57: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-19

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia Usuario:** U-QTO-19
Nombre de la Tarea: Implementación de la funcionalidad de Hold.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2,4 semanas
Fecha de Inicio: 15/03/2010	**Fecha de Fin:** 31/03/2010
Programador Responsable: Miguel Angel Chavez Alfonso	
Descripción: Permite la visualización de varias funciones en un mismo eje de coordenadas.	

Tabla 3.58: Tarea de Ingeniería #2 perteneciente a la Historia de Usuario U-QTO-19

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia Usuario:** U-QTO-20
Nombre de la Tarea: Desarrollo de la ventana de interacción individual en 3D	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1,2 semanas
Fecha de Inicio: 15/03/2010	**Fecha de Fin:** 31/03/2010
Programador Responsable: Jorge Luis Hernández Cruz	
Descripción: Se implementa una ventana para la visualización de gráficas en 3D donde el usuario puede eliminar una función y realizar operaciones de rotación y acercamientos.	

Tabla 3.59: Tarea de Ingeniería #1 perteneciente a la Historia de Usuario U-QTO-20

3.4. Diseño con metáforas

En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume de forma evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto, se solventan con la existencia de una metáfora. Una metáfora es una historia compartida que describe como debería funcionar el sistema. La práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema[28]. En el desarrollo de la solución propuesta no se definió una arquitectura específica en el estado inicial, sino que se utilizó el diseño con metáforas, que permite encontrar un dominio del problema sin que medie una arquitectura estática. La arquitectura de la aplicación desarrollada se describe a través de un diagrama de componentes.

Diagrama de Componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Este diagrama muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. También es considerado un grafo de componentes unidos a través de relaciones que pueden ser de compilación o de ejecución, y además se pueden representar las interfaces de esos componentes. Cuando se habla de interfaces se refiere a la descripción de las operaciones que se realizan en esos componentes y en esos subsistemas. El diagrama de componentes constituye otra forma de representar una vista estática del sistema, que representa la organización y dependencia que habría entre los componentes físicos que se necesitan para ejecutar la aplicación, sean estos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación.

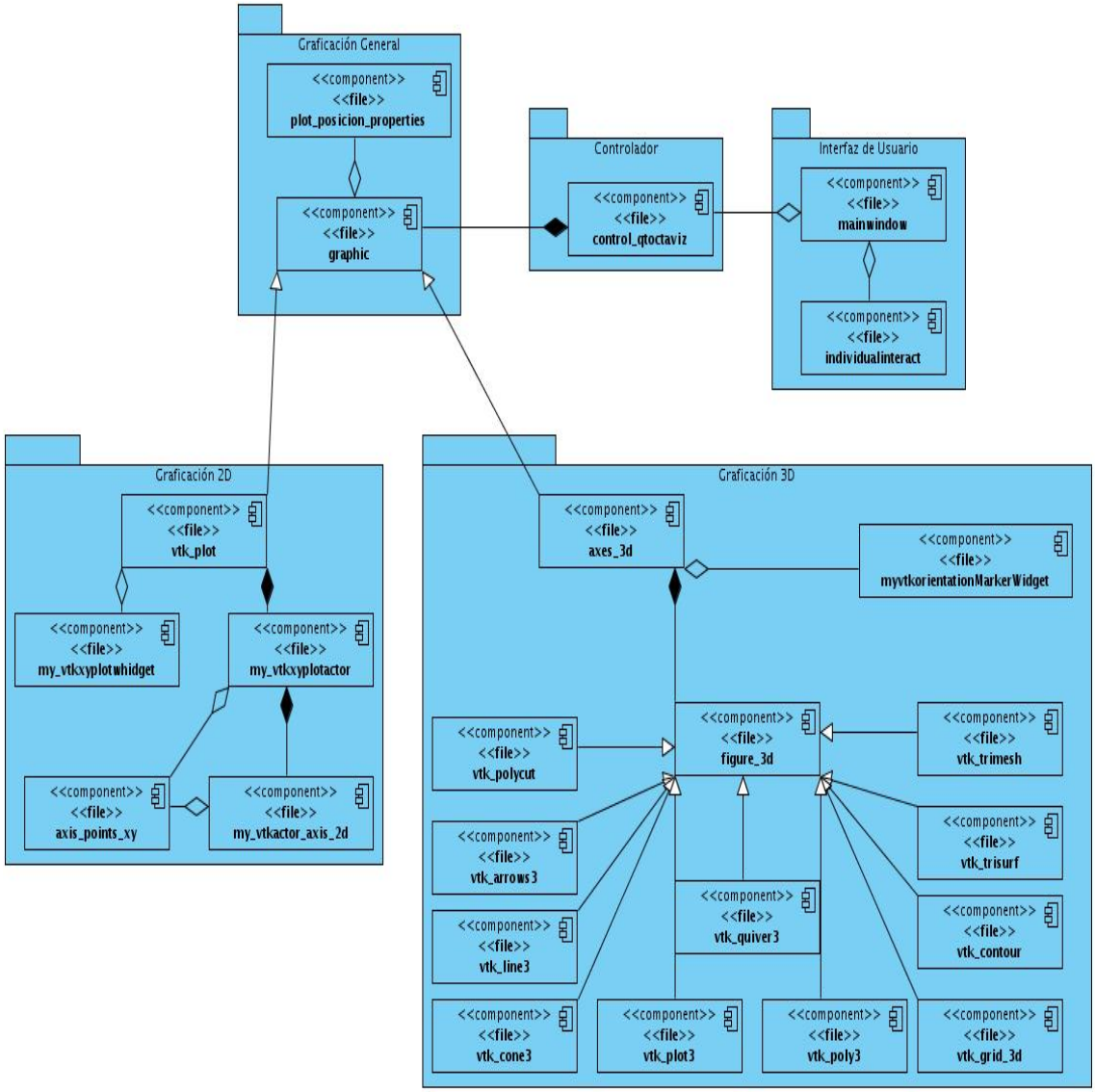


Figura 3.1: Diagrama de componentes

A continuación se realizará la descripción de los componentes que integran el diagrama representado en la figura 3.1.

Paquete Interfaz de Usuario.

Constituido por las clases que controlan todo los métodos relacionados con la interfaz visual del programa.

Contiene todos los métodos encargados de validar la interfaz de usuario, además de los métodos que gestionan todas las propiedades de las gráficas así como de controlar toda la interactividad.

Figura 3.2: Clase MainWindow.

Contiene los método que permiten realizar una interactividad de forma individual a las figuras 3D.

Figura 3.3: Clase IndividualInteract.

Paquete Control.

Constituido por una clase encargada de controlar todo el flujo de información entre los componentes visuales de la aplicación y las clases que gestionan las gráficas representadas.

Encargada de controlar todos los métodos que gestionan la información de las gráficas tanto 2D como 3D.

Figura 3.4: Clase ControlQToctaviz.

Paquete Graficación General.

Está formado por las clases generales que controlan todo el acceso y modificación de la información de todas las gráficas representadas en la ventana principal.

Superclase de la cual heredan las clases vtk_plot2 y figure_3d, contiene todos los métodos referente a las propiedades de las gráficas.

Figura 3.5: Clase graphic.

Encargada de retornar los valores donde se encuentra posicionada la gráfica, así como los valores de las propiedades de altura y ancho.

Figura 3.6: Clase plotPositionProperties.

Paquete Graficación 2D.

Contiene las clases que permiten el acceso y modificación de las propiedades de las gráficas de tipo 2D.

Encargada de representar las gráficas en 2D, utilizando el método add_curve. Posibilita la gestión de la información relativa a las gráficas de tipo 2D.

Figura 3.7: Clase VTKPlot.

Se redefinen algunos métodos de la clase vtkXYPlotWidget y controla la interactividad de las gráficas 2D una vez visualizadas.

Figura 3.8: Clase my_vtkXYPlotWidget.

Se redefinen algunos métodos de la clase vtkXYPlotActor y se encarga de visualizar las gráficas en 2D.

Figura 3.9: Clase my_vtkXYPlotActor.

Encargada de retornar el tamaño del eje coordenado x y y, para la construcción de las mallas en las gráficas de 2D.

Figura 3.10: Clase axis_points_xy.

Se redefinen algunos métodos de la clase vtkAxisActor2D que son imprescindibles para la visualización de las mallas en las gráficas 2D.

Figura 3.11: Clase my_vtkAxisActor2D.

Paquete Graficación 3D.

Contiene las clases que permiten el acceso y modificación de las propiedades de las gráficas de tipo 3D.

Encargada de construir el eje de coordenadas y la caja que engloba las figuras de tipo 3D y gestionar las propiedades del mismo.

Figura 3.12: Clase Axes_3D.

Clase que hereda de vtkOrientationMarkerWidget que permite conocer el estado en que se encuentra el widget que visualiza las gráficas 3D cuando se le pasa el ratón por encima y se da clic. Esta funcionalidad permite actualizar la gráfica activa en el programa, es decir, la última gráfica que el usuario seleccionó.

Figura 3.13: Clase myvtkOrientationMarkerWidget.

Superclase de la cual heredan todas las gráficas en 3D, conteniendo todos los métodos que gestionan sus propiedades.

Figura 3.14: Clase Figure_3D.

Encargada de visualizar la gráfica polycut 3D para lo cual utiliza el método plot_polycut y gestionar propiedades propias de la gráfica.

Figura 3.15: Clase vtk_Polycut.

Encargada de visualizar la gráfica arrows 3D para lo cual utiliza el método plot_arrows3 y gestionar propiedades propias de la gráfica.

Figura 3.16: Clase VTK_arrows3.

Encargada de visualizar la gráfica Line 3D para lo cual utiliza el método plot_line3d y gestionar propiedades propias de la gráfica.

Figura 3.17: Clase VTKLine3.

Encargada de visualizar la gráfica cone 3D para lo cual utiliza el método plot_cone_3d y gestionar propiedades propias de la gráfica.

Figura 3.18: Clase VTK_Cone3.

Encargada de visualizar la gráfica quiver 3D para lo cual utiliza el método plot_quiver3d y gestionar propiedades propias de la gráfica.

Figura 3.19: Clase VTK_Quiver3.

Encargada de visualizar la gráfica plot 3D para lo cual utiliza el método plot_plot3d y gestionar propiedades propias de la gráfica.

Figura 3.20: Clase VTKPlot3.

Encargada de visualizar la gráfica poly 3D para lo cual utiliza el método plot_poly_3d y gestionar propiedades propias de la gráfica.

Figura 3.21: Clase VTK_Poly3.

Encargada de visualizar la gráfica trimesh 3D y gestionar propiedades propias de la gráfica. Utiliza los métodos plot_surfaceActor, plot_edgeActor y plot_ballActor para la representación de los tres actores que componen la función.

Figura 3.22: Clase VTK_Trimesh.

Encargada de visualizar la gráfica trisurf 3D para lo cual utiliza el método plot_trisurf y gestionar propiedades propias de la gráfica.

Figura 3.23: Clase VTK_Trisurf.

Encargada de visualizar la gráfica contour 3D para lo cual utiliza el método plot_contour_3 y gestionar propiedades propias de la gráfica.

Figura 3.24: Clase VTK_Contour.

Encargada de construir los grid en 3D y gestionar sus propiedades.

Figura 3.25: Clase grid3D.

3.5. Plan de Releases.

El plan de releases recoge las iteraciones a realizar con sus características, además del orden de las historias de usuario con su planificación estimada para ser implementadas[28].

Release	**Descripción de la iteración**	**Orden de la HU a implementar**	**Duración total**
1	Se diseña la interfaz de usuario de la aplicación y se desarrollan las historias de usuarios que permiten graficar las funciones matemáticas en 2D y algunas funciones de tipo 3D.	U-QTO-01 U-QTO-02 U-QTO-03 U-QTO-04 U-QTO-05 U-QTO-06 U-QTO-07 U-QTO-08	20/09/2009- 20/12/2009
2	Son adicionadas varias funciones matemáticas de tipo 3D.	U-QTO-09 U-QTO-10 U-QTO-11 U-QTO-12 U-QTO-13 U-QTO-14 U-QTO-15	12/01/2010- 28/02/2010

3	Son implementadas las funcionalidades que permiten al usuario la gestión de la información relativa a las gráficas visualizadas.	U-QTO-16 U-QTO-17 U-QTO-18 U-QTO-19 U-QTO-20	1/03/2010- 31/03/2010

Tabla 3.60: Plan de Releases

3.6. Conclusiones

En este capítulo fueron presentados los elementos relacionados con la ingeniería del sistema desarrollado. Fueron plasmadas todas las historias de usuarios presentes en el sistema y las tareas de ingeniería relacionadas con cada una de las historias de usuario. De forma adicional se presenta el diagrama de componentes con la explicación de cada una de las clases y paquetes que lo componen.

Capítulo 4

Pruebas del sistema

El presente capítulo recoge el proceso de validación de la nueva propuesta a través de la solución de varios problemas que permiten visualizar las gráficas necesarias y la posterior gestión de las propiedades de los elementos de las mismas. De forma adicional, son presentados los casos de prueba de aceptación recogidos como parte de la metodología de desarrollo de software seleccionada, las cuales se enfocan en la verificación del cumplimiento de las funcionalidades declaradas en el proceso de concepción de la aplicación.
De manera colateral, estas últimas aportan información relativa a la interactividad y rendimiento de la aplicación.

4.1. Solución de problemas para la validación del sistema

Para validar la eficiente interactividad que ofrece la nueva propuesta con relación a las gráficas una vez visualizadas, se decide tomar dos ejercicios como muestra y desarrollarlos hasta la graficación de sus resultados para con posterioridad establecer una comparación detallada entre las gráficas construidas con Matlab, las desarrolladas desde Octave con el uso de Gnuplot y aquellas obtenidas mediante el empleo de Qtoctaviz. Ambos ejercicios fueron tomados del libro "**Introducción informal a Matlab y Octave**"[24]. Para la comparación

de las gráficas de tipo 2D se toma el ejercicio **Diseño de una tobera**[1] descrito en el epígrafe 9.2, mientras que para el caso de las gráficas de tipo 3D se toma como referencia el ejercicio **El Atractor de Lorentz**[2] referido en el epígrafe 9.3 del citado libro.

4.1.1. Solución del ejercicio *¨Diseño de una tobera¨*

Para la resolución de este ejercicio se cuenta con dos funciones auxiliares (consultar *Anexos 4 y 5*) utilizadas por el asistente matemático para el cálculo de los valores necesarios para la posterior graficación. Con la ejecución del fichero de comandos referido en el *Anexo 7* tanto desde Matlab como desde Octave, puede observarse que se obtienen gráficas similares (ver *Anexos 8 y 9*). A partir de algunas variaciones en el fichero de comandos (ver Anexo 6) para la ejecución de la nueva propuesta desde Octave, se obtiene como resultado la gráfica mostrada en la figura 4.1 donde partiendo de una inspección visual y posterior comparación con los resultados obtenidos empleando Matlab y Gnuplot puede aseverarse que la nueva propuesta arroja resultados positivos durante los procesos de graficación de funciones en 2D.

[1] Consultar Anexo 3

[2] Consultar Anexo 10

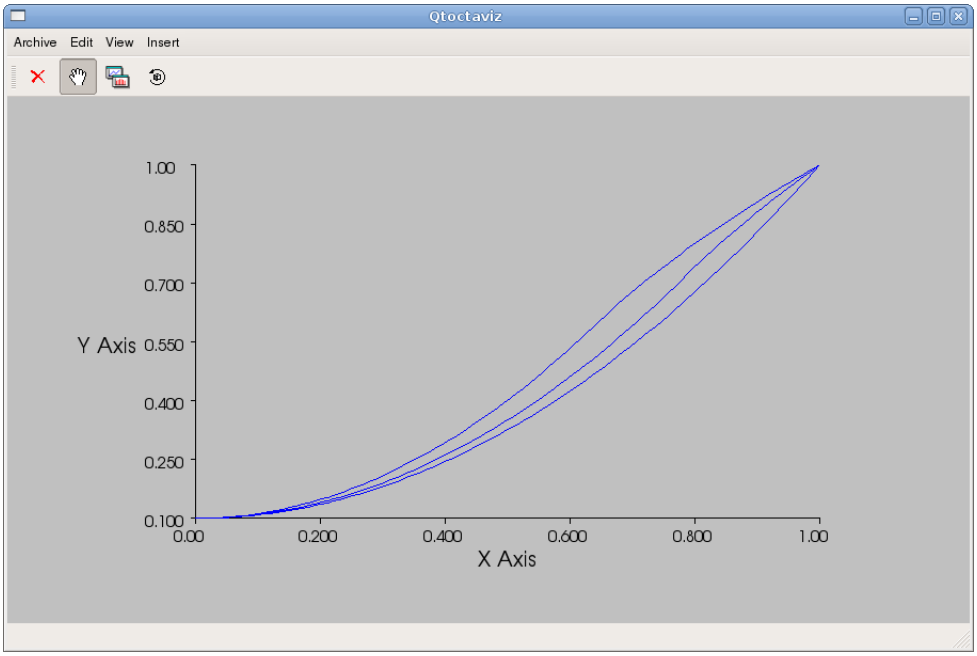


Figura 4.1: Representación de los resultados del diseño de una tobera mediante Qtoctaviz

A partir de esta gráfica se procede a denotar un título y un color a cada curva de forma independiente mediante la activación del elemento **Property Editor** del menú **Edit** a través de las ventanas emergentes **Property Editor - Grafic** y **Plot Browser**. Adicionalmente se activa la casilla **Legend** del menú **View** para visualizar la leyenda de la gráfica y se inserta un título a través del elemento **Title** del menú **Insert** (ver figura 4.2).

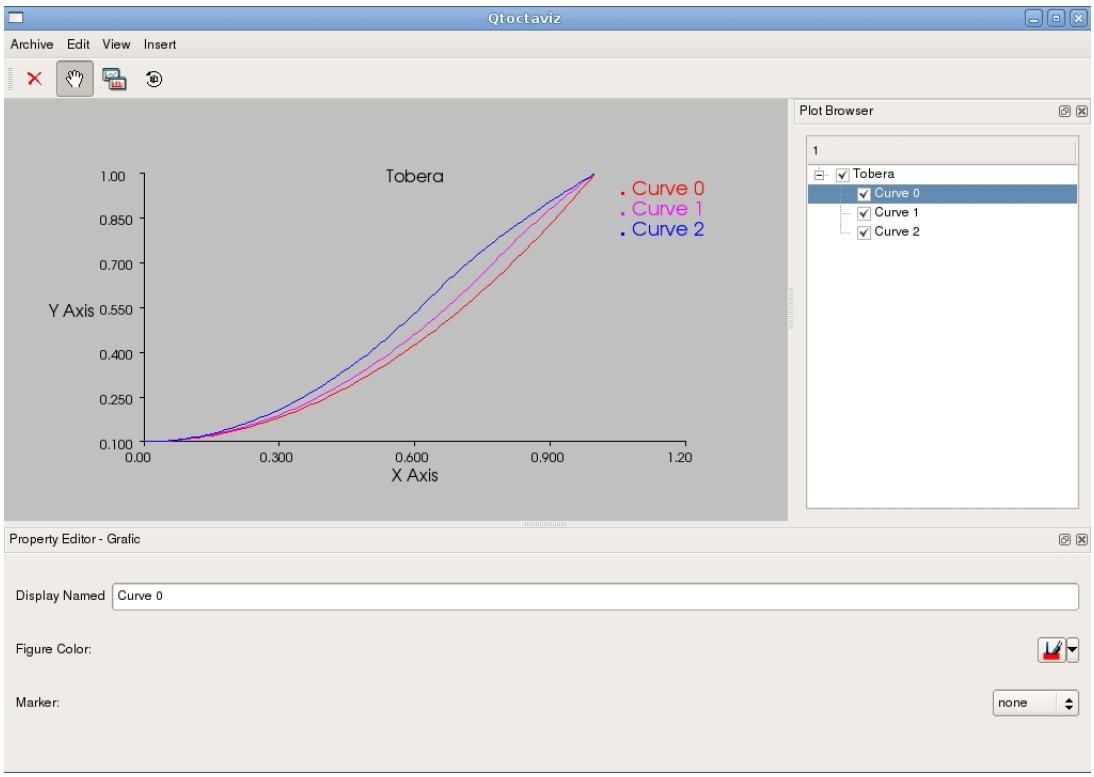


Figura 4.2: Modificación de las propiedades de las curvas y el título

Para modificar las propiedades de formato de los ejes de coordenadas, es visualizada la ventana emergente **Property Editor - Axes** donde se selecciona un nuevo color para las líneas y el texto de los ejes, se modifica el tipo de letra seleccionando **Times**, se visualizan las mallas del eje x y se activan las propiedades de texto con sombra, cursivo y en negrita(ver figura 4.3).

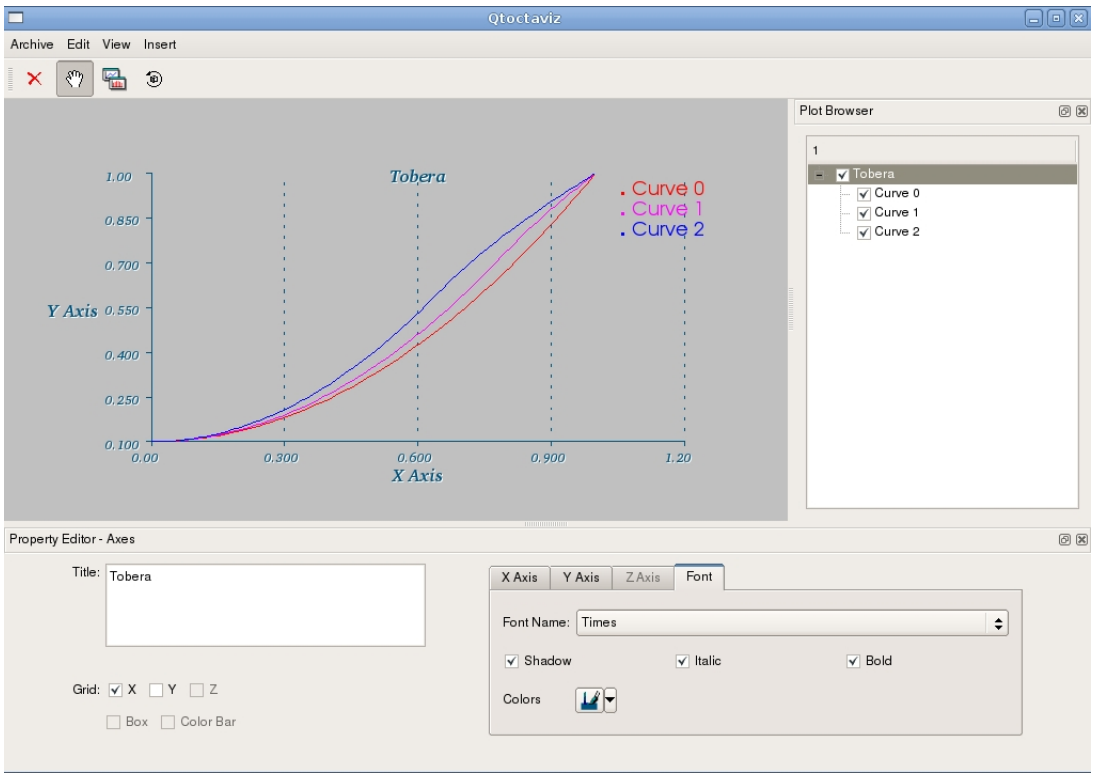


Figura 4.3: Modificación de las propiedades de color de los ejes coordenados

Con el objetivo de visualizar los puntos guías de un intervalo acotado en el eje x se modificaron los límites relacionados a este eje, se introduce como título del eje un nuevo texto, se oculta la información referente al eje y mediante el elemento **Y Axis** del menú **View** y se selecciona un marcador diferente para cada curva representada (ver figura 4.4).

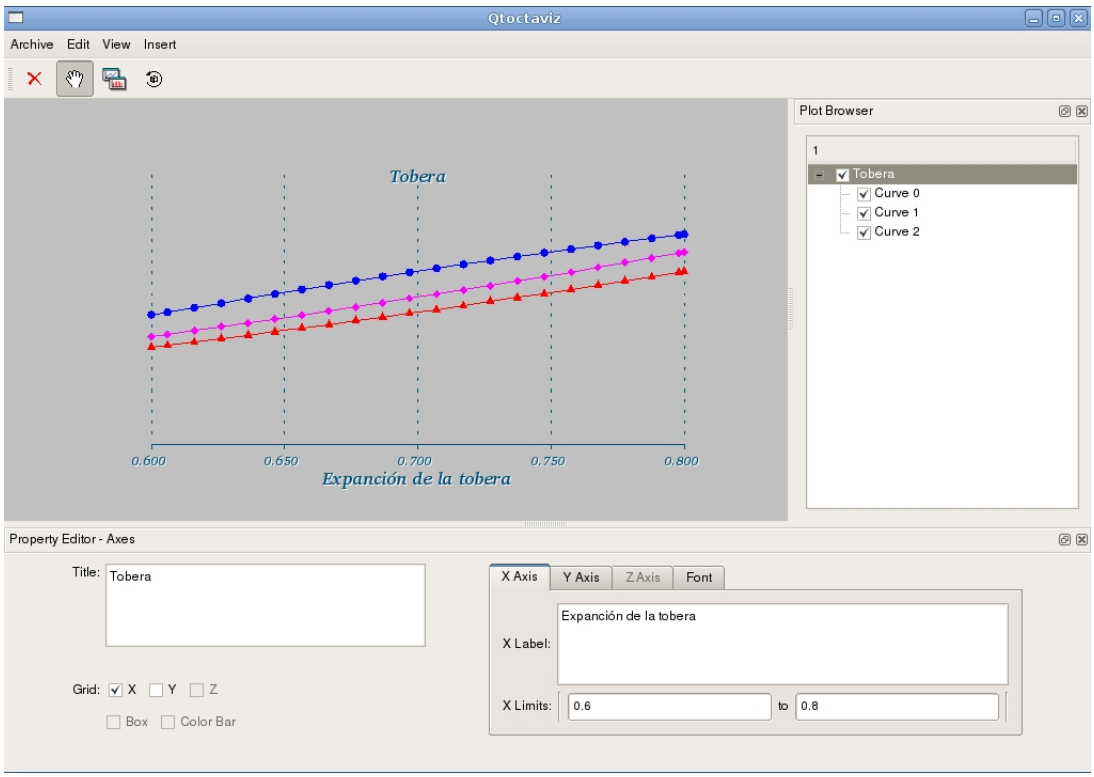


Figura 4.4: Acotación y visualización de los marcadores de las funciones

4.1.2. Solución del ejercicio *¨El Atractor de Lorentz¨*

Para la resolución de este ejercicio se cuenta con las función auxiliares que se recogen en los *Anexos 11 y 14* del trabajo las cuales son empleadas por Octave y Matlab respectivamente para el cálculo de los valores necesarios para la posterior graficación. Con la ejecución del fichero de comandos del *Anexo 12 y 15* se obtiene una representación en tres dimensiones de la solución del problema en los ambientes de Gnuplot (consultar *Anexo 16*) y Matlab (consultar *Anexo 17*). De manera similar al caso anterior, tras algunas modificaciones al referido fichero de comandos la gráfica relacionada es visualizada en Qtoctaviz siendo su apariencia la recogida en la figura 4.5.

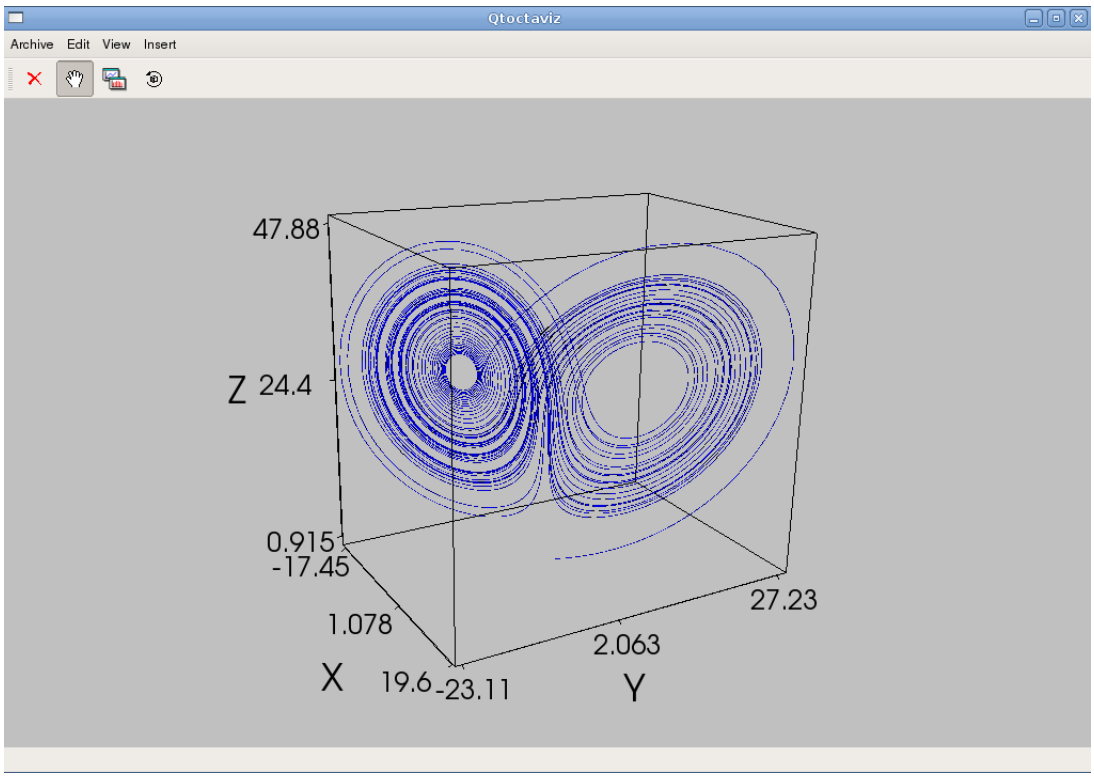


Figura 4.5: Representación de los resultados del atractor de Lorentz mediante Qtoctaviz

A partir de esta gráfica se procede a denotar un título y modificar el color de la figura representada mediante la activación del elemento **Property Editor** del menú **Edit** a través de las ventanas emergentes **Property Editor - Grafic** y **Plot Browser**. Adicionalmente se visualizan los marcadores que definen los puntos guías en forma de triángulos y se deshabilitan los elementos del eje de coordenadas para una mejor apreciación de la figura (ver figura 4.6).

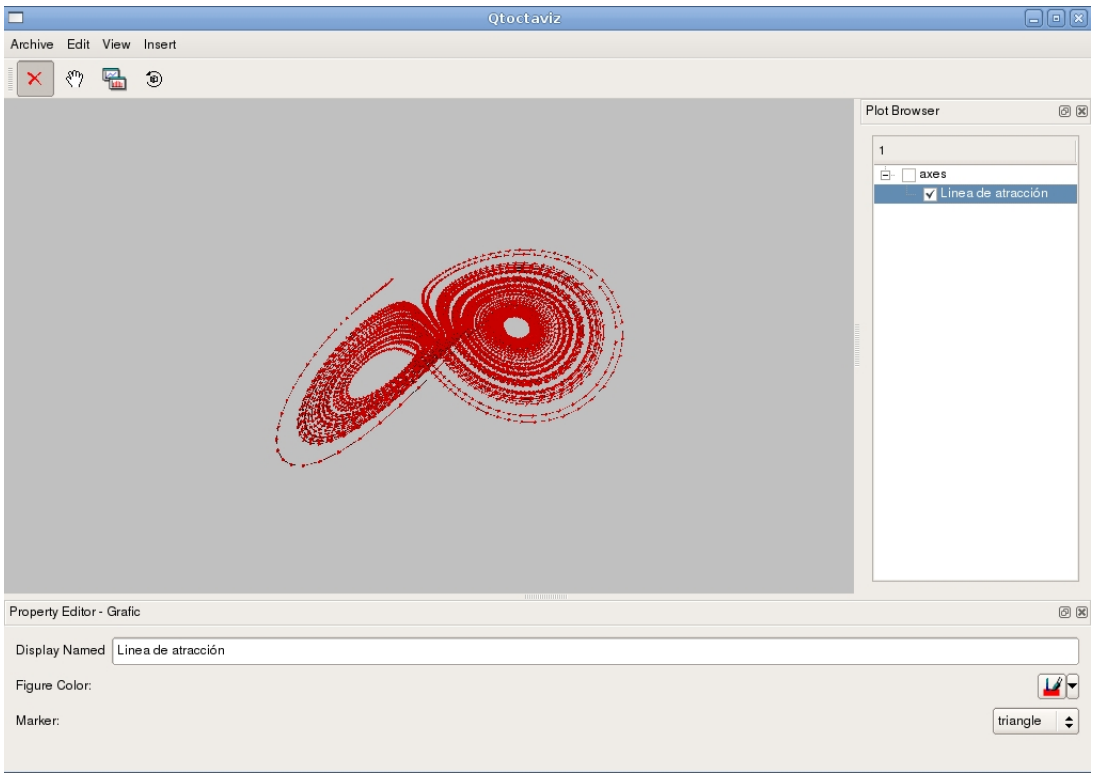


Figura 4.6: Modificación de las propiedades de la gráfica

Para modificar las propiedades de formato de los ejes de coordenadas, es visualizada la ventana emergente **Property Editor - Axes** donde se selecciona un nuevo color para las líneas y el texto de los ejes, se modifica el tipo de letra seleccionando **Couriel** y se activan las propiedades de texto con sombra. De forma adicional se visualizan las mallas del eje x modificando su opacidad y posición mediante la ventana emergente **Grid Editor 3D**, se oculta la caja que engloba la figura, los elementos de los ejes y y z, se introduce un nuevo título al eje x y un titulo a la gráfica en general (ver figura 4.7).

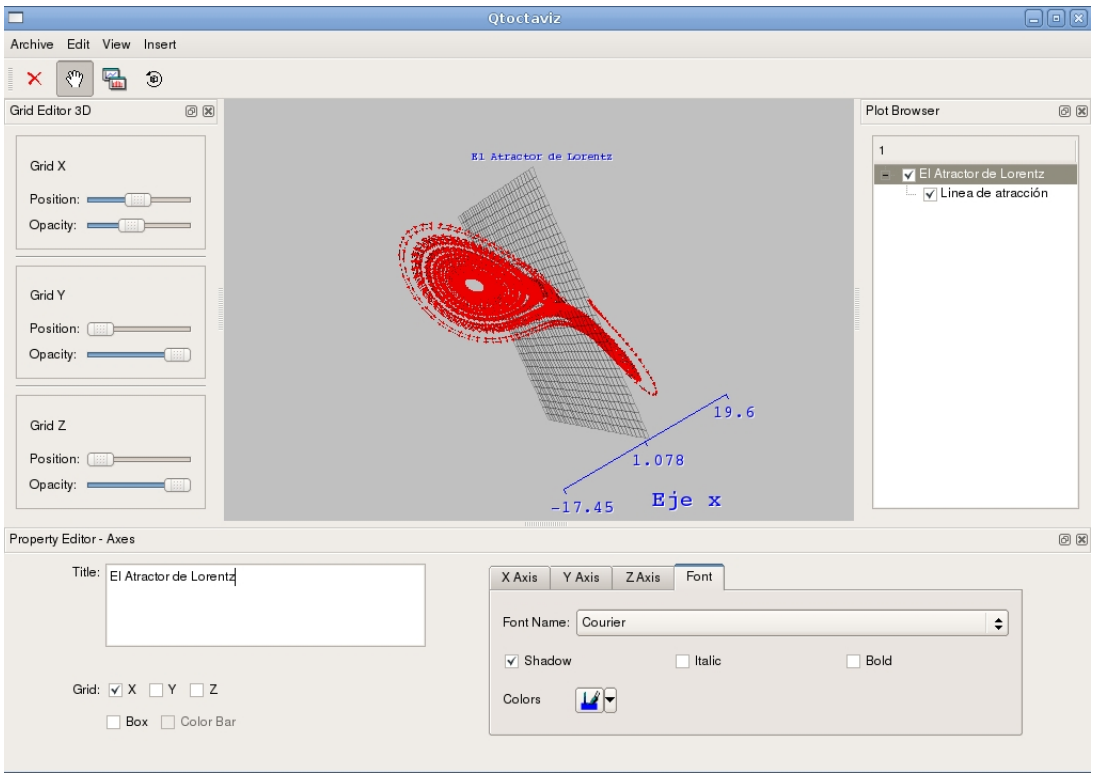


Figura 4.7: Modificación de las propiedades de los ejes coordenados y las mallas

Con el objetivo de obtener una mejor vista de los elementos de la figura representada se activa la opción de interactividad **Rotate 3D**; la cual visualiza una nueva ventana con los componentes visibles, permitiendo un acercamiento de la figura y un cambio de color de fondo (ver figura 4.8).

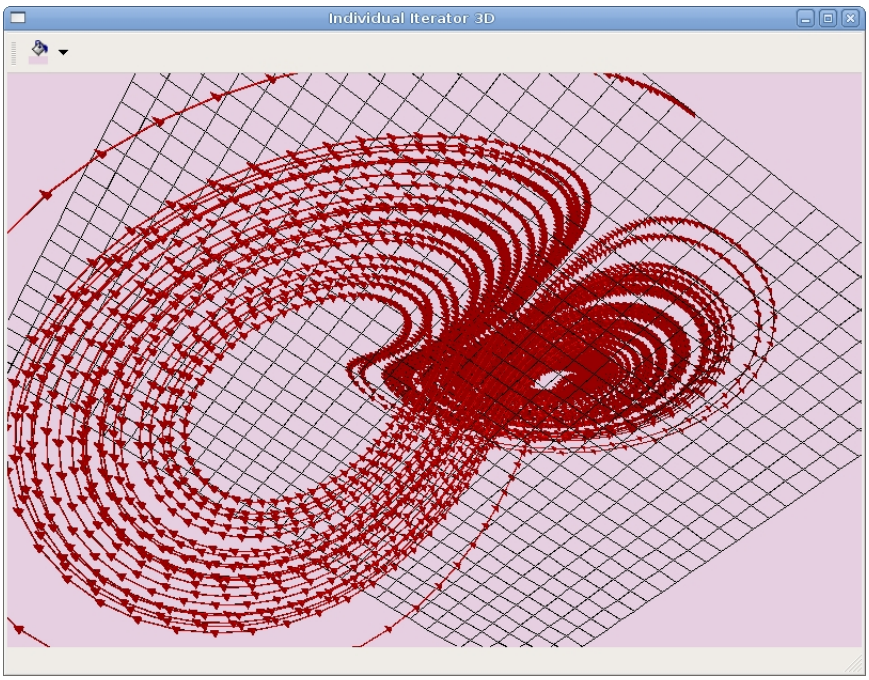


Figura 4.8: Opción de interactividad *Rotate 3D*

4.2. Casos de Pruebas

Las pruebas de aceptación son definidas por el cliente y preparadas por el equipo de desarrollo; no obstante, la ejecución y aprobación final corresponden al primero. La utilización de estas pruebas son de vital importancia en el proceso de desarrollo; toda vez que permiten a los programadores tener una idea más clara e inequívoca de la calidad de su trabajo, a la vez que se garantiza la entrega de un producto en elevada correspondencia con las necesidades del usuario final. En esta sección son planteados los casos de pruebas más importantes, con el objetivo de demostrar el correcto funcionamiento de la aplicación desarrollada. No son plasmados los casos de pruebas referidos a las historias de usuarios entre la U-QTO-04 y la U-QTO-14 pues son similares a las pruebas realizadas a la historia de usuario U-QTO-03 alcanzando en todos los casos una evaluación satisfactoria[3].

[3]La omisión obedece a limitaciones propias de este tipo de reporte. Puede consultarse el resto de los casos de prueba en la documentación oficial del desarrollo.

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-02-001	**Nombre Historia de Usuario:** Plot 2D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave, se ejecuta la aplicación Qtoctaviz y se muestra la gráfica resultante.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Escribir comando o series de comandos.	
Entrada / Pasos de ejecución: 1. Ejecutar la aplicación (Octave). 2. Introducir a la consola de Octave los comandos siguientes: ▪ $x = 0 : 0{,}2 : 2;$ ▪ $vtk\_plot(x, sin(x))$	
Resultado Esperado: Se muestra la gráfica requerida en el ambiente de Qtoctaviz.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.1: Caso de prueba U-QTO-02-001

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-02-002	**Nombre Historia de Usuario:** Plot 2D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se modifica el color de la figura representada.	

Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).
Entrada / Pasos de ejecución: 1. Habilitar la opción “Axes Properties” en el menú “Edit”. 2. Habilitar la opción “Plot Browser” en el menú “Edit”. 3. Señalar la gráfica requerida en la ventana “Plot Browser”. 4. Seleccionar un color en el botón de herramienta referente a “Figure Color”
Resultado Esperado: La gráfica representada es visualizada con el color seleccionado en el botón de herramientas referente a “Figure Color”.
Evaluación de la Prueba: Satisfactoria

Tabla 4.2: Caso de prueba U-QTO-02-002

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-02-003	**Nombre Historia de Usuario:** Plot 2D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se visualizan las mallas en los diferentes ejes(x,y) con la utilización de las cajas de eventos chequeables de la ventana “Axes Properties”.	

Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).
Entrada / Pasos de ejecución: 1. Habilitar la opción “Axes Properties” en el menú “Edit”. 2. 1.Dar clic en la casilla referente a “Grid” en sus dos componentes (x,y).
Resultado Esperado: Se visualizan las mallas en los distintos ejes.
Evaluación de la Prueba: Satisfactoria

Tabla 4.3: Caso de prueba U-QTO-02-003

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-02-004	**Nombre Historia de Usuario:** Plot 2D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se visualizan las mallas en los diferentes ejes(x,y) mediante el menú “View”.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	

Entrada / Pasos de ejecución: 1. Seleccionar en el menú “View” la opción “Grid” en uno de sus componentes (X Grid, Y Grid).
Resultado Esperado: Se visualizan las mallas en los distintos ejes según la opción seleccionada.
Evaluación de la Prueba: Satisfactoria

Tabla 4.4: Caso de prueba U-QTO-02-004

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-02-005	**Nombre Historia de Usuario:** Plot 2D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se modifica el marcador para la representación de los puntos guías de la figura	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	

Entrada / Pasos de ejecución: 1. Habilitar la opción “Axes Properties” en el menú “Edit”. 2. Habilitar la opción “Plot Browser” en el menú “Edit”. 3. Señalar la gráfica requerida en la ventana “Plot Browser”. 4. Seleccionar una de las opciones del menú desplegable referente a la opción “Marker”.
Resultado Esperado: La gráfica representada es visualizada con el marcador seleccionado en el menú desplegable.
Evaluación de la Prueba: Satisfactoria

Tabla 4.5: Caso de prueba U-QTO-02-005

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-03-001	**Nombre Historia de Usuario:** Plot 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave, se ejecuta la aplicación Qtoctaviz y se muestra la gráfica resultante.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Escribir comando o series de comandos.	

Entrada / Pasos de ejecución:

1. Ejecutar la aplicación (Octave).
2. Introducir a la consola de Octave los comandos siguientes:

- $o = [306219; 82110];$
- $n = [312111; 184214; 554617; 562513];$
- $c = [54915; 305016; 424215; 432913; 182812; 3268; 633615; 596020];$
- $h = [2357; 32016; 3750; 733616; 696020; 546228; 576612; 65916; 14422; 0496];$
- $vtk\_plot3(o(:, 1), o(:, 2), o(:, 3));$

Resultado Esperado: Se muestra la gráfica requerida en el ambiente de Qtoctaviz.

Evaluación de la Prueba: Satisfactoria

Tabla 4.6: Caso de prueba U-QTO-03-001

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-03-002	**Nombre Historia de Usuario:** Plot 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se modifica el color de la figura representada.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	

Entrada / Pasos de ejecución: 1. Habilitar la opción “Axes Properties” en el menú “Edit”. 2. Habilitar la opción “Plot Browser” en el menú “Edit”. 3. Señalar la gráfica requerida en la ventana “Plot Browser”. 4. Seleccionar un color en el botón de herramienta referente a “Figure Color”.
Resultado Esperado: La gráfica representada es visualizada con el color seleccionado en el botón de herramientas referente a “Figure Color”.
Evaluación de la Prueba: Satisfactoria

Tabla 4.7: Caso de prueba U-QTO-03-002

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-03-003	**Nombre Historia de Usuario:** Plot 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se visualizan las mallas en los diferentes ejes(x,y,z) con la utilización de las cajas de eventos chequeables de la ventana “Axes Properties”.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	

Entrada / Pasos de ejecución: 1. Habilitar la opción “Axes Properties” en el menú “Edit”. 2. Dar clic en la casilla referente a “Grid” en sus dos componentes (x,y,z).
Resultado Esperado: Se visualizan las mallas en los distintos ejes.
Evaluación de la Prueba: Satisfactoria

Tabla 4.8: Caso de prueba U-QTO-03-003

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-03-004	**Nombre Historia de Usuario:** Plot 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se visualizan las mallas en los diferentes ejes(x,y,z) mediante el menú “View”.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	
Entrada / Pasos de ejecución: 1. Seleccionar en el menú “View” la opción “Grid” en uno de sus componentes (X Grid, Y Grid, Z Grid).	
Resultado Esperado: Se visualizan las mallas en los distintos ejes según la opción seleccionada.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.9: Caso de prueba U-QTO-03-004

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-03-005	**Nombre Historia de Usuario:** Plot 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se modifica el marcador para la representación de los puntos guías de la figura.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	
Entrada / Pasos de ejecución: 1. Habilitar la opción "Axes Properties" en el menú "Edit". 2. Habilitar la opción "Plot Browser" en el menú "Edit". 3. Señalar la gráfica requerida en la ventana "Plot Browser". 4. Seleccionar una de las opciones del menú desplegable referente a la opción "Marker".	
Resultado Esperado: La gráfica representada es visualizada con el marcador seleccionado en el menú desplegable.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.10: Caso de prueba U-QTO-03-005

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-03-006	**Nombre Historia de Usuario:** Plot 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se realiza un desplazamiento de las mallas en cualquiera de sus componentes (x,y,z).	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	
Entrada / Pasos de ejecución: 1. Habilitar la opción "Grid Editor 3D" en el menú "Edit". 2. Desplazar la barra desplegable referente a la posición de una de las mallas (x,y,z).	
Resultado Esperado: La malla se desplaza a lo largo del eje seleccionado.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.11: Caso de prueba U-QTO-03-006

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-03-007	**Nombre Historia de Usuario:** Plot 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se realiza un desplazamiento de la opacidad de las mallas en cualquiera de sus componentes (x,y,z).	

Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).
Entrada / Pasos de ejecución: 1. Habilitar la opción “Grid Editor 3D” en el menú “Edit”. 2. Desplazar la barra desplegable referente a la opacidad de una de las mallas (x,y,z).
Resultado Esperado: Se modifica la opacidad de la malla seleccionada.
Evaluación de la Prueba: Satisfactoria

Tabla 4.12: Caso de prueba U-QTO-03-007

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-16-001	**Nombre Historia de Usuario:** Axes 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se modifican los títulos de los ejes coordenados en sus tres componentes(x,y,z) a través de la ventana “Axes Properties”.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	

Entrada / Pasos de ejecución: 1. Habilitar la opción “Axes Properties” en el menú “Edit”. 2. Introducir un nuevo título en el campo de título del eje coordenado seleccionado.
Resultado Esperado: SEl eje de coordenada señalado toma el nuevo título introducido.
Evaluación de la Prueba: Satisfactoria

Tabla 4.13: Caso de prueba U-QTO-16-001

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-16-002	**Nombre Historia de Usuario:** Axes 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se modifican los títulos de los ejes coordenados en sus tres componentes(x,y,z) a través del menú “Insert”.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	

Entrada / Pasos de ejecución: 1. Seleccionar en el menú “Insert” una de las opciones para insertar un título en el eje deseado. 2. Introducir un nuevo título en el campo de título del eje coordenado seleccionado y presionar “ok”.
Resultado Esperado: El eje de coordenada señalado toma el nuevo título introducido.
Evaluación de la Prueba: Satisfactoria

Tabla 4.14: Caso de prueba U-QTO-16-002

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-16-003	**Nombre Historia de Usuario:** Axes 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se modifican las propiedades de visibilidad de los componentes de los ejes coordenados.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	
Entrada / Pasos de ejecución: 1. Seleccionar una de las acciones del menú “View” en sus tres componentes (X Axis, Y Axis, Z Axis).	

Resultado Esperado: Se visualiza u oculta el elemento seleccionado en dependencia del valor de la casilla.
Evaluación de la Prueba: Satisfactoria

Tabla 4.15: Caso de prueba U-QTO-16-003

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-16-004	**Nombre Historia de Usuario:** Axes 3D
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Modificar los elementos de formato de texto de los ejes coordenados.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	
Entrada / Pasos de ejecución: 1. Habilitar la opción “Axes Properties” en el menú “Edit”. 2. 1.Cambiar los valores de la pestaña “Font” seleccionado entre estos “Font Name” de tipo Arial, “Shadow” y el color rojo en la opción “Colors”.	
Resultado Esperado: El texto de los ejes coordenados cambia su composición a tipo de letra Arial, con sombra y de color rojo.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.16: Caso de prueba U-QTO-16-004

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-17-001	**Nombre Historia de Usuario:** Salvar Como
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se realiza la exportación de la gráfica representada en formato png.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	
Entrada / Pasos de ejecución: 1. Seleccionar en el menú “Archive” la opción “Save as” y se selecciona como extinción “PNG Image”. 2. Realizar una búsqueda del directorio donde se desea guardar la imagen en la ventana emergente “Save as...”. 3. Introducir en el campo “File name” el nombre que va a tener la imagen deseada. 4. Dar clic en el botón “Save”.	
Resultado Esperado: Se genera una imagen en el formato seleccionado con el nombre dado y en el directorio señalado.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.17: Caso de prueba U-QTO-17-001

Caso de prueba de Aceptación

Código Caso de Prueba: U-QTO-17-002	**Nombre Historia de Usuario:** Salvar Como
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se realiza la exportación de la gráfica representada en formato jpeg.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).	
Entrada / Pasos de ejecución: 1. Seleccionar en el menú "Archive" la opción "Save as" y se selecciona como extinción "JPEG Image". 2. Realizar una búsqueda del directorio donde se desea guardar la imagen en la ventana emergente "Save as...". 3. Introducir en el campo "File name" el nombre que va a tener la imagen deseada. 4. Dar clic en el botón "Save".	
Resultado Esperado: Se genera una imagen en el formato seleccionado con el nombre dado y en el directorio señalado.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.18: Caso de prueba U-QTO-17-002

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-17-003	**Nombre Historia de Usuario:** Salvar Como

Nombre de la persona que realiza la prueba: Vianka Orovio Cobo
Descripción de la Prueba: Mediante el ambiente de Qtoctaviz se realiza la exportación de la gráfica representada en formato bmp.
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Mantener activa la aplicación (Qtoctaviz).
Entrada / Pasos de ejecución: 1. Seleccionar en el menú "Archive" la opción "Save as" y se selecciona como extinción "BMP Image". 2. Realizar una búsqueda del directorio donde se desea guardar la imagen en la ventana emergente "Save as...". 3. Introducir en el campo "File name" el nombre que va a tener la imagen deseada. 4. Dar clic en el botón "Save".
Resultado Esperado: Se genera una imagen en el formato seleccionado con el nombre dado y en el directorio señalado.
Evaluación de la Prueba: Satisfactoria

Tabla 4.19: Caso de prueba U-QTO-17-003

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-18-001	**Nombre Historia de Usuario:** Vtk_Subplot
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	

Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave, se ejecuta la aplicación Qtoctaviz y se muestran la gráficas resultantes con la opción "vtk_subplot".

Condiciones de Ejecución:

1. Mantener activa la aplicación (Octave).
2. Escribir comando o series de comandos.

Entrada / Pasos de ejecución:

1. Ejecutar la aplicación (Octave).
2. Introducir a la consola de Octave los comandos siguientes:

- $o = [306219; 82110];$
- $n = [312111; 184214; 554617; 562513];$
- $c = [54915; 305016; 424215; 432913; 182812; 3268; 633615; 596020];$
- $h = [2357; 32016; 3750; 733616; 696020; 546228; 576612; 65916; 14422; 0496];$
- $vtk\_subplot(2, 2, 1), vtk\_plot3(o(:, 1), o(:, 2), o(:, 3));$
- $vtk\_subplot(2, 2, 2), vtk\_plot3(o(:, 1), o(:, 2), o(:, 3));$

Resultado Esperado: Se muestran dos gráficas en las posiciones requeridas por las opciones de "vtk_subplot" en el ambiente de Qtoctaviz.

Evaluación de la Prueba: Satisfactoria

Tabla 4.20: Caso de prueba U-QTO-18-001

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-18-002	**Nombre Historia de Usuario:** Vtk_Subplot
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave, se realiza la petición de graficar una función con la opción subplot en una posición en que ya existe una gráfica representada.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Escribir comando o series de comandos.	
Entrada / Pasos de ejecución: 1. Ejecutar la aplicación (Octave). 2. Introducir a la consola de Octave los comandos siguientes: ▪ $o = [306219; 82110]$; ▪ $n = [312111; 184214; 554617; 562513]$; ▪ $c = [54915; 305016; 424215; 432913; 182812; 3268; 633615; 596020]$; ▪ $h = [2357; 32016; 3750; 733616; 696020; 546228; 576612; 65916; 14422; 0496]$; ▪ $vtk\_subplot(2, 2, 1), vtk\_plot3(o(:, 1), o(:, 2), o(:, 3))$; ▪ $vtk\_subplot(2, 2, 1), vtk\_plot3(o(:, 1), o(:, 2), o(:, 3))$;	
Resultado Esperado: El sistema remplaza la gráfica existente en las posiciones determinadas por la nueva gráfica y representa esta.	
Evaluación de la Prueba: Satisfactoria	

Tabla 4.21: Caso de prueba U-QTO-18-002

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-19-001	**Nombre Historia de Usuario:** Vtk_Hold
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave, se ejecuta la aplicación Qtoctaviz y se muestran la gráficas resultantes con la opción "vtk_hol("on")".	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Escribir comando o series de comandos.	

Entrada / Pasos de ejecución:

1. Ejecutar la aplicación (Octave).
2. Introducir a la consola de Octave los comandos siguientes:

- $o = [306219; 82110];$
- $n = [312111; 184214; 554617; 562513];$
- $c = [54915; 305016; 424215; 432913; 182812; 3268; 633615; 596020];$
- $h = [2357; 32016; 3750; 733616; 696020; 546228; 576612; 65916; 14422; 0496];$
- $vtk\_plot3(o(:,1), o(:,2), o(:,3));$
- $vtk\_hold('on');$
- $nmeridian = 6; nlongitude = 11;$
- $phi = 0 : pi/1000 : 2 * pi;$
- $mu = phi * nmeridian;$
- $x = cos(mu). * (1 + cos(nlongitude * mu/nmeridian)/2{,}0);$
- $y = sin(mu). * (1 + cos(nlongitude * mu/nmeridian)/2{,}0);$
- $z = sin(nlongitude * mu/nmeridian)/2{,}0;$
- $vtk\_line3(x', y', z', 'Radius', 0{,}05);$

Resultado Esperado: Se muestran dos funciones en un mismo eje de coordenadas en el ambiente de Qtoctaviz.

Evaluación de la Prueba: Satisfactoria

Tabla 4.22: Caso de prueba U-QTO-19-001

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-18-002	**Nombre Historia de Usuario:** Vtk_Hold
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave, se realiza la petición de graficar una función con la opción vtk_hold("on") y luego se envía la opción vtk_hold("off") y se manda a graficar otra función.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Escribir comando o series de comandos.	

Entrada / Pasos de ejecución:

1. Ejecutar la aplicación (Octave).
2. Introducir a la consola de Octave los comandos siguientes:

- $o = [306219; 82110];$
- $n = [312111; 184214; 554617; 562513];$
- $c = [54915; 305016; 424215; 432913; 182812; 3268; 633615; 596020];$
- $h = [2357; 32016; 3750; 733616; 696020; 546228; 576612; 65916; 14422; 0496];$
- $vtk\_plot3(o(:,1), o(:,2), o(:,3));$
- $vtk\_hold('on');$
- $nmeridian = 6; nlongitude = 11;$
- $phi = 0 : pi/1000 : 2 * pi;$
- $mu = phi * nmeridian;$
- $x = cos(mu). * (1 + cos(nlongitude * mu/nmeridian)/2{,}0);$
- $y = sin(mu). * (1 + cos(nlongitude * mu/nmeridian)/2{,}0);$
- $z = sin(nlongitude * mu/nmeridian)/2{,}0;$
- $vtk_line3(x', y', z', 'Radius', 0{,}05);$
- $vtk\_hold('off');$
- $vtk\_line3(x', y', z', 'Radius', 0{,}05);$

Resultado Esperado: Se muestran dos funciones en un mismo eje de coordenadas en el ambiente de Qtoctaviz.
Evaluación de la Prueba: Satisfactoria

Tabla 4.23: Caso de prueba U-QTO-19-002

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-19-003	**Nombre Historia de Usuario:** Vtk_Hold
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	
Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave se realiza la petición de graficar una función de tipo 2D con la opción vtk_hold("on") cuando no existe ninguna gráfica de tipo 2D representada.	
Condiciones de Ejecución: 1. Mantener activa la aplicación (Octave). 2. Escribir comando o series de comandos.	

Entrada / Pasos de ejecución:

1. Ejecutar la aplicación (Octave).
2. Introducir a la consola de Octave los comandos siguientes:

- $o = [306219; 82110];$
- $n = [312111; 184214; 554617; 562513];$
- $c = [54915; 305016; 424215; 432913; 182812; 3268; 633615; 596020];$
- $h = [2357; 32016; 3750; 733616; 696020; 546228; 576612; 65916; 14422; 0496];$
- $vtk\_plot3(o(:, 1), o(:, 2), o(:, 3));$
- $vtk\_hold('on');$
- $x = 0 : 0, 2 : 10;$
- $vtk\_plot(x, sin(x))$

Resultado Esperado: El sistema muestra un mensaje de error donde informa al usuario que no es posible graficar una función en 2 dimensiones con la opción vtk_hold("on") si no existe una gráfica de este tipo previamente visualizada.

Evaluación de la Prueba: Satisfactoria

Tabla 4.24: Caso de prueba U-QTO-19-003

Caso de prueba de Aceptación	
Código Caso de Prueba: U-QTO-19-004	**Nombre Historia de Usuario:** Vtk_Hold
Nombre de la persona que realiza la prueba: Vianka Orovio Cobo	

Descripción de la Prueba: Mediante la escritura de comandos en la consola de Octave se realiza la petición de graficar una función de tipo 3D con la opción vtk_hold("on") cuando no existe ninguna gráfica de tipo 3D representada.

Condiciones de Ejecución:

1. Mantener activa la aplicación (Octave).
2. Escribir comando o series de comandos.

Entrada / Pasos de ejecución:

1. Ejecutar la aplicación (Octave).
2. Introducir a la consola de Octave los comandos siguientes:

- $x = 0 : 0,2 : 10;$
- $vtk\_plot(x, sin(x))$
- $vtk\_hold('on');$
- $o = [306219; 82110];$
- $n = [312111; 184214; 554617; 562513];$
- $c = [54915; 305016; 424215; 432913; 182812; 3268; 633615; 596020];$
- $h = [2357; 32016; 3750; 733616; 696020; 546228; 576612; 65916; 14422; 0496];$
- $vtk\_plot3(o(:, 1), o(:, 2), o(:, 3));$

Resultado Esperado: El sistema muestra un mensaje de error donde informa al usuario que no es posible graficar una función en 3 dimensiones con la opción vtk_hold("on") si no existe una gráfica de este tipo previamente visualizada.

Evaluación de la Prueba: Satisfactoria

Tabla 4.25: Caso de prueba U-QTO-19-004

4.3. Conclusiones

En este capítulo fueron presentados dos ejercicios resueltos, permitiéndose establecer una comparación entre los resultados arrojados por la aplicación propuesta y aquellos obtenidos a través de Gnuplot, quien constituye actualmente el graficador por defecto del asistente matemático Octave. De manera adicional, se presentó de manera secuencial algunas de las opciones de interactividad más importante de que dispone Qtoctaviz, a la vez que fueron expuestos los casos de pruebas de aceptación del desarrollo las cuales permiten sellar el correcto funcionamiento de la aplicación.

Conclusiones

Con la realización de este trabajo se cierra un período de desafíos y gratitudes asociadas a la superación de un número importante de insatisfacciones en la esfera de la visualización científica, las que dificultaban el trabajo de los centros de investigaciones y casas de altos estudios que utilizan Octave. Entre los aspectos más significativos del trabajo se pueden mencionar:

- Fue desarrollado un sistema para la Visualización Científica, que permite la graficación interactiva en dos y tres dimensiones de elementos procesados por Octave, así como la exportación de gráficas en diferentes formatos y la gestión de la información relativa a las mismas.
- Fueron caracterizados los graficadores más utilizados por Octave así como las potencialidades de la librería gráfica VTK.
- Se desarrolló una conexión desde Octave con la aplicación externa de graficación.
- Se sometió el sistema propuesto a un importante número de pruebas concebidas como parte del proceso de desarrollo, así como a la solución de ejercicios prácticos reales validándose en cada caso tanto el funcionamiento como su elevado nivel de interactividad.

Recomendaciones

El desarrollo de herramientas que permitan un uso más eficiente de los programas que existen bajo el paradigma de software libre, constituye un aporte considerable a la calidad de la enseñanza en Universidades de todo el mundo y de nuestro país de manera especial. En la búsqueda de este anhelo el presente trabajo constituye un paso importante y en lo particular un hito para el desarrollo posterior de la herramienta Octave, por lo que se recomienda:

- Incorporar nuevos tipos de gráficas a visualizar como pueden ser los gráficos de pastel y gráficos de barras.
- Incrementar el número de funcionalidades de gestión de la información relativa a las gráficas visualizadas en la aplicación.
- Atender la compatibilidad de la propuesta con otros sistemas de cálculo como Maxima y R.
- Planificar despliegues a gran escala a fin de validar el completo funcionamiento del sistema.

Referencias

[1] "Historia de la representación." http://www.scribd.com/doc/11051999/Tema-1-Historia-de-La-Representacion, Noviembre 2009.

[2] "Breve historia de la graficación." http://www.mitecnologico.com/Main/BreveHistoriaDeLaGraficacion, Noviembre 2009.

[3] R. Goering, "*Matlab edges closer to electronic design automation world*." http://www.eetimes.com/news/design/showArticle.jhtml?articleID=4940039, Enero 2009.

[4] MathWorks, "*The MathWorks Store*." http://www.mathworks.com/store/priceListLink.do, Julio 2009.

[5] Y. Cariaga and H. Rodríguez, "Análisis socio tecnológico de la migración hacia software libre en la facultad 10 de la universidad de las ciencias informáticas(uci)," *Informática 2009, La Habana, Cuba*, Febrero 2009.

[6] A. Companioni, "Entorno integrado para el desempeño matemático," *Aceptado para publicar*, 2009.

[7] "*Gnuplot Homepage*." http://www.gnuplot.info, Octubre 2009.

[8] "*Octaviz*." http://octaviz.sourceforge.net, Octubre 2009.

[9] Y. Gámez, V. Moreno, and Y. Martínez, "Conceptualización de un nodo virtual de procesos," *Informática 2009, La Habana, Cuba*, Febrero 2009.

[10] N. Hernández, "Biblioteca digital multimedia como apoyo al estudio del sistema operativo linux.," *Informática 2009, La Habana, Cuba*, Febrero 2009.

[11] A. Leyva, "Cuamweb. sitio web sobre la cátedra universitaria del adulto mayor en el municipio puerto padre.," *Informática 2009, La Habana, Cuba*, Febrero 2009.

[12] M. C. Santos, Y. Rodríguez, and M. Capetillo, "En el geroclub las cañas. presentación multimedia viviendo lo eterno," *Informática 2009, La Habana, Cuba*, Febrero 2009.

[13] J. W. Eaton, "*About GNU Octave.*" `http://plasma-gate.weizmann.ac.il/Grace`, 2000.

[14] "*Grace.*" `http://plasma-gate.weizmann.ac.il/Grace`, 2000.

[15] "*PLplot: The ultimate in cross-platform plotting.*" `http://sourceforge.net/projects/plplot`, 2009.

[16] "*OctPlot.*" `http://plplot.sourceforge.net`, 2009.

[17] "*Vizualization Toolkit.*" `http://www.vtk.org`, 2009.

[18] "*Modular Class Library.*" `http://qt.nokia.com/products/library`, 2008.

[19] "*Anjuta DevStudio: GNOME Integrated Development Environment.*" `http://projects.gnome.org/anjuta`, 2009.

[20] "*QT Creator, desarrollando aplicaciones rápidamente.*" `http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente`, 2009.

[21] *Aprenda C++ como si estuviera en primero.* Aprenda Informática, 1998.

[22] "*Comunicación entre procesos.*" `http://platea.pntic.mec.es/amora/asi200405/tema9.htm`.

[23] P. C., "*Hilos y Mutex*," 2008.

[24] B. G., "*Introducción informal a Matlab y Octave.*" `http://iimyo.forja.rediris.es/`, 2008.

[25] M. A. Mendoza, "Metodologías de desarrollo de software," *Informatizate*, Junio 2004.

[26] *Una explicación de la Programación Extrema: Aceptar el cambio.* Addison Wesley Iberoamericana Espanya, S.A, 2002.

[27] K. Schwaber, B. M., and M. R. C., *Agile Software Development with SCRUM.* Prentice Hall, 2001.

[28] A. Meneses, G. M. Peñalver, M. Rodríguez, R. Fernández, and S. Pino, "Sxp: Metodología ágil para proyectos de software libre.," 2009.

Bibliografía

[1] C. Alvarez, *Metodología de la Investigación Científica.*, Santiago de Cuba, Cuba, 1995.

[2] C. Athó, and F. Cruz, *OpenGL y VTK.*, Universidad Nacional de Trujillo, Perú, 2006.

[3] G. B. Nogueras, *Octave: Una alternativa real a Matlab a coste de cero.*, 7 p.

[4] J. W. Eaton, D. Bateman, *Octave FAQ. Frequently asked questions about Octave*, Septiembre 2007.

[5] J. W. Eaton, D. Bateman and S. Hauberg, *GNU Octave, A high-level interactive language for numerical computations*, 3er Edition for Octave version 3.0.1, Julio 2007.

[6] M. C. Juárez, W. G. Herrera and S. T. Sánchez, *Software Libre vs. Software Propietario. Ventajas y desventajas.*, México, 2006.

[7] R. Hernández, C. Fernández, and P. Baptista, *Metodología de la Investigación.*, Segunda Edición, 1998.

[8] T. C. Rebollo, E. C. Vera and M. G. Mármol, *Notas sobre Matlab y Octave.*, 29 p.

[9] W. J. Schroeder, L. S. Avila and W. Hoffman, *Visualizing with VTK: A Tutorial.*, Octubre 2006.

Anexos

Anexo 1: Terminología de Colores para el cambio de propiedad de las gráficas.

Color	Caracter	Vector
Rojo	'r'	[1 0 0]
Azul	'b'	[0 0 1]
Magenta	'm'	[1 0 1]
Negro	'k'	[0 0 0]
Verde	'g'	[0 1 0]
Amarillo	'y'	[1 1 0]
Blanco	'w'	[1 1 1]

Tabla 4.26: Sistema de Colores

Anexo 2: Opciones para la representación de una gráfica en vtk_plot3

Opción	Gráfica
“ . ”	Representa puntos en 3D
“ ^ ”	Representa cubos en 3D
“ > ”	Representa conos en 3D
“ * ”	Representa esferas en 3D

Tabla 4.27: Opciones para vtk_plot3

Anexo 3: Ejercicio. *¨Diseño de una tobera¨*

Queremos diseñar una tobera convergente divergente. Para ello impondremos que el radio de salida sea 10 veces mayor que el radio de la garganta, y que la tobera se forma mediante dos parábolas, una con eje en y y otra con eje en x. Las condiciones serán que en el punto de empalme de los dos arcos haya continuidad tanto de la función como de la derivada primera. Las dos funciones a ajustar serán entonces (con el problema convenientemente adimensionalizado)

$$y^- = Px^2 + 0{,}1 \; y^+ = \sqrt{Ax + B}$$

Entonces tenemos el sistema de ecuaciones siguiente, donde l es el punto en x donde se empalman los dos arcos:

$$2Pl = \frac{A}{2\sqrt{Al + B}}$$
$$Pl^2 + 0{,}1 = \sqrt{Ax + B}$$
$$\sqrt{A + B} = 1$$

Donde se demuestra que existe solución para P aproximadamente $0,9 < P < 1,2$. Resolver el sistema de ecuaciones anterior y representar las soluciones de $P = 0,9, P = 1$ y $P = 1,2$.

Anexo 4: Función tobera utilizada en el ejercicio *¨Diseño de una tobera¨*

```
function[f]=tobera(x,a,b,l,P)
if x<l
      f=P*(x*x)+0.1;
    else
      f=sqrt(a*x+b);
end
```

Anexo 5: Función ttb utilizada en el ejercicio *¨Diseño de una tobera¨*

```
function[f]=ttb(x)
    global Pi
    f(1)=x(1)/(2*sqrt(x(1)*x(3)+x(2)))-2*Pi*x(3);
    f(2)=Pi*x(3)^2+0.1-sqrt(x(1)*x(3)+x(2));
    f(3)=sqrt(x(1)+x(2))-1;
end
```

Anexo 6: Comandos para la resolución del ejercicio *¨Diseño de una tobera¨* desde Octave con el uso de Qtoctaviz

```
x0=[1 1 1];
P=[0.9 1 1.2];
for i=1:3
  global Pi
  Pi=P(i);
  [res]=fsolve('ttb',x0);
  xcoord=linspace(0,1,100);
  for j=1:100
    tob(j)=tobera(xcoord(j),res(1),res(2),res(3),P(i));
  end
  vtk_plot(xcoord,tob)
  vtk_hold("on")
end
vtk_hold("off")
```

Anexo 7: Comandos para la resolución del ejercicio *¨Diseño de una tobera¨* desde Matlab y desde Octave con el uso de Gnuplot

```
x0=[1 1 1];
P=[0.9 1 1.2];
hold on
for i=1:3
  global Pi
  Pi=P(i);
  [res]=fsolve('ttb',x0);
  xcoord=linspace(0,1,100);
  for j=1:100
    tob(j)=tobera(xcoord(j),res(1),res(2),res(3),P(i));
  end
  plot(xcoord,tob)
end
hold off
```

Anexo 8: Gráfica resultante del ejercicio *¨Diseño de una tobera¨* mediante Matlab

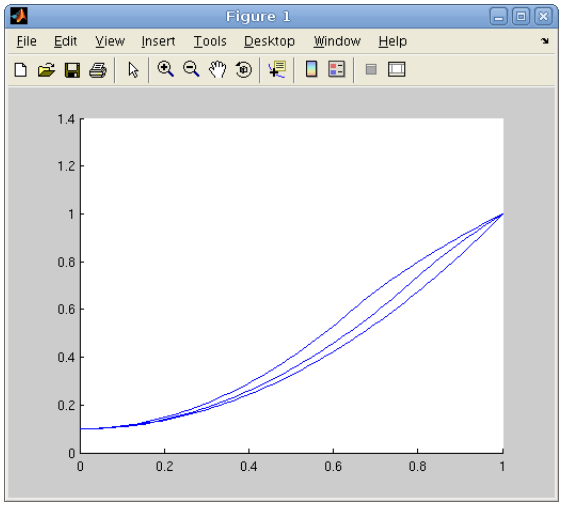


Figura 4.9: Representación gráfica de diseño de una tobera mediante Matlab

Anexo 9: Gráfica resultante del ejercicio *¨Diseño de una tobera¨* mediante Gnuplot

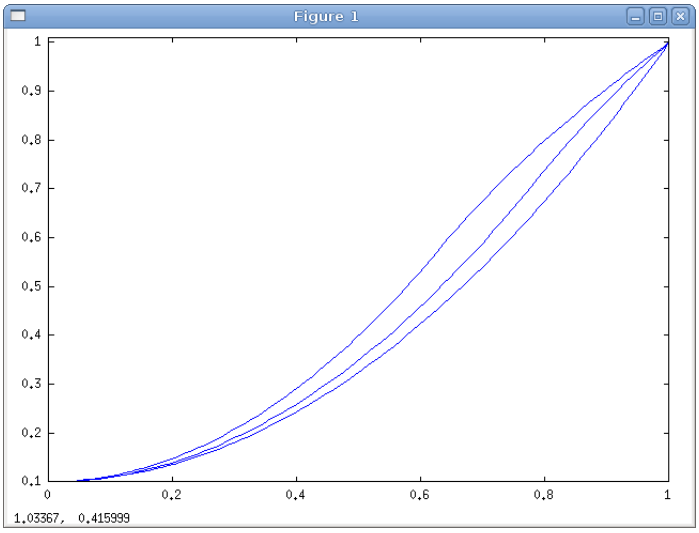


Figura 4.10: Representación gráfica de diseño de una tobera mediante Gnuplot

.

Anexo 10: Ejercicio. *¨El atractor de Lorentz¨*

Se quiere integrar la ecuación diferencial del Atractor de Lorentz, de ecuaciones:

$$\dot{x} = a(y - x)$$
$$\dot{y} = x(b - z) - y \; con \; a = 10, b = 28, c = \frac{8}{3}$$
$$\dot{z} = xy - cz$$

y representarla en el espacio.

Anexo 11: Función *func* utilizada en el ejercicio *¨El atractor de Lorentz¨* mediante Octave

```
function xdot=func(x,t)
a=10;b=28;c=8/3;
xdot(1,1)=a*(x(2)-x(1));
xdot(2,1)=x(1)*(b-x(3))-x(2);
xdot(3,1)=x(1)*x(2)-c*x(3);
end
```

Anexo 12: Comandos para la resolución del ejercicio *¨El atractor de Lorentz¨* desde Octave con el uso de Gnuplot

```
x0=[1;1;1];
t=linspace(0,50,5000);
tic;x=lsode("func",x0,t);toc
plot3(x(:,1),x(:,2),x(:,3))
```

Anexo 13: Comandos para la resolución del ejercicio *¨El atractor de Lorentz¨* desde Octave con el uso de Qtoctaviz

```
x0=[1;1;1];
t=linspace(0,50,5000);
tic;x=lsode("func",x0,t);toc
vtk_line3(x(:,1),x(:,2),x(:,3),'Radius',0.05)
```

Anexo 14: Función utilizada en el ejercicio *¨El atractor de Lorentz¨* mediante Matlab

```
function xdot=lorentz(t,x)
  a=10;b=28;c=8/3;
  xdot(1,1)=a*(x(2)-x(1));
  xdot(2,1)=x(1)*(b-x(3))-x(2);
  xdot(3,1)=x(1)*x(2)-c*x(3);
end
```

Anexo 15: Comandos para la resolución del ejercicio *¨El atractor de Lorentz¨* desde Matlab

```
x=0;t=0;
tic;[t,x]=ode45(@lorentz,[0,50],[1,1,1]);toc
plot3(x(:,1),x(:,2),x(:,3))
```

Anexo 16: Gráfica resultante del ejercicio *¨El atractor de Lorentz¨* mediante Gnuplot

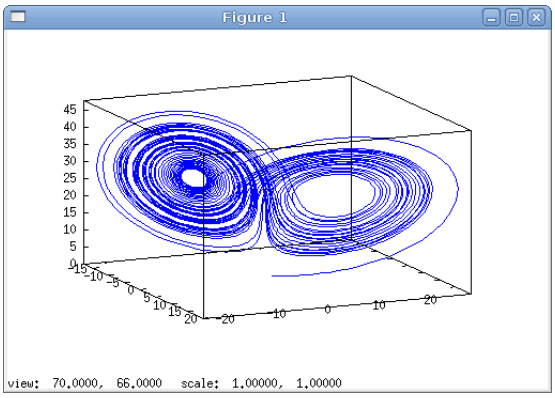


Figura 4.11: Representación gráfica de *¨El atractor de Lorentz¨* mediante Gnuplot

Anexo 17: Gráfica resultante del ejercicio *¨El atractor de Lorentz¨* mediante Matlab

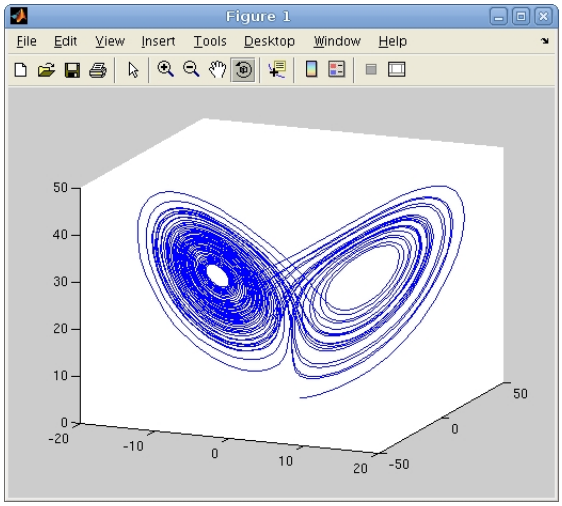


Figura 4.12: Representación gráfica de *¨El atractor de Lorentz¨* mediante Matlab