UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

PROGRAMA DE INGENIERÍA DE SISTEMAS

TRABAJO DE GRADO PRESENTADO PARA OPTAR AL TÍTULO

DE INGENIERO DE SISTEMAS

TEMA:

HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES CIENTIFICAS: ESTUDIO COMPARATIVO

AUTOR:

JHON JAIRO LOZANO BUITRAGO

PAMPLONA, NORTE DE SANTANDER

DICIEMBRE DE 2016

UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

PROGRAMA DE INGENIERÍA DE SISTEMAS

TRABAJO DE GRADO PRESENTADO PARA OPTAR AL TÍTULO

DE INGENIERO DE SISTEMAS

TEMA:

HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES CIENTIFICAS: ESTUDIO COMPARATIVO

AUTOR:

JHON JAIRO LOZANO BUITRAGO

DIRECTOR: PHD. JOSE ORLANDO MALDONADO BAUTISTA

DIRECTOR DE PROGRAMA: PHD. CARLOS ARTURO PARRA ORTEGA

PAMPLONA, NORTE DE SANTANDER

DICIEMBRE DE 2016

UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

PROGRAMA DE INGENIERÍA DE SISTEMAS

TRABAJO DE GRADO PRESENTADO PARA OPTAR AL TÍTULO

DE INGENIERO DE SISTEMAS

TEMA:

HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES CIENTIFICAS: ESTUDIO COMPARATIVO

FECHA DE INICIO DEL TRABAJO: AGOSTO DE 2016

FECHA DE TERMINACIÓN DEL TRABAJO: DICIEMBRE DE 2016

NOMBRES Y FIRMAS DE AUTORIZACIÓN PARA SUSTENTACIÓN

_____

Jhon Jairo Lozano Buitrago

AUTOR

_____ _____

PhD. José Orlando Maldonado B. PhD. Carlos Arturo Parra Ortega

Tabla de contenido

1. Introducción

1.1 Planteamiento del problema

El desarrollo de software es algo que ha venido evolucionando a medida que avanza el tiempo, así mismo van evolucionando las herramientas necesarias para llevar acabo dicho desarrollo de software, donde estas pueden tener varios tipos de licenciamiento entre ellos libre o comercial, cada uno con características diferentes las cuales pueden limitar o beneficiar el desarrollo de software. Particularmente en el caso de aplicaciones científicas ya sea para el modelado, simulación o análisis de datos, la oferta de herramientas de desarrollo se ha ampliado, generando un gran número de posibilidades, dependiendo de la aplicación y las necesidades particulares del área de estudio.

Es por tanto que debido al gran número de herramientas de desarrollo ya sean de tipo comercial o licenciamiento libre se desea realizar un estudio donde se puedan evidenciar estas características, ámbito en el que se aplican, ventajas y desventajas que presentan unas frente a las otras para así poder establecer una herramienta acorde a las necesidades que se tienen frente al desarrollo de aplicaciones científicas.

El estudio finaliza con un documento en el cual queda escrito todas y cada una de las consideraciones importantes que se encontraron, para que sirva como base para futuros desarrolladores de aplicaciones científicas, logrando así que encuentren la mejor opción que se puede llevar a cabo para dicho desarrollo.

1.2 Justificación

En diferentes grupos de investigación de la Universidad de Pamplona, se requiere el desarrollo de aplicaciones científicas para modelar problemas de diversa naturaleza. En la mayoría de los casos los investigadores utilizan software propietario como Matlab, Matemática, SPSS, Arena, entre otros. Diferentes comunidades que apoyan el uso y generación de software libre ofrecen alternativas para el desarrollo de aplicaciones de este tipo. Por tanto se pretende determinar las ventajas y dificultades que se tiene con el uso de estas herramientas y en lo posible incentivar a la comunidad académica al uso de herramientas de desarrollo libres, de forma que por un lado se reduzca el costo en licenciamiento, y de otro se cree la cultura del uso y desarrollo de software libre. En forma específica en este proyecto se pretende solucionar la necesidad de crear software educativo para estudiantes de Ingeniería el cual les ayude a entender y aplicar los modelos matemáticos objeto de estudio siendo este el pretexto para que dicho software sea desarrollado con herramientas de licenciamiento libre y código abierto para así ser utilizado en la Universidad de Pamplona y posteriormente poder seguir implementándole funcionalidades.

2. Objetivos

2.1 Objetivo general

- Realizar un estudio comparativo sobre algunas herramientas de software libre para el desarrollo de aplicaciones científicas.

2.2 Objetivos específicos

- Identificar las principales herramientas de software libre disponibles para el desarrollo de aplicaciones científicas describiendo sus características principales.
- Desarrollar una aplicación para el tratamiento de imágenes, utilizando al menos dos de las herramientas identificadas.
- Comparar cada una de las herramientas seleccionadas para el desarrollo de la aplicación, destacando sus ventajas y desventajas al momento de implementar la misma.

3. Estado del arte.

A continuación se muestra los resultados de una revisión de la bibliografía reciente, sobre el desarrollo de software científico en diferentes áreas del conocimiento. El objetivo de la revisión es identificar algunas características de las aplicaciones que desarrollan los científicos en la actualidad así como las principales herramientas utilizadas.

En el documento escrito por **(Judith Segal & Chris Morris, 2008),** se da una idea de que es lo que hace diferente el software científico del software comercial, y la principal característica radica en que el objeto de estudio del software científico es muy complejo y solo puede ser entendido por los mismo científicos que dominan el tema, es por esto que en la mayoría de los casos el científico es quien se convierte en el desarrollador.

Al revisar el documento escrito por **(Kelly, 2009),** la autora nos muestra un modelo para poder examinar la evolución que se tiene a largo plazo de la vida útil en el desarrollo de software científico, ya que estas aplicaciones requieren de un mantenimiento activo para prolongar su existencia, debido a que la ingeniería de software no ha tenido muy en cuenta este tipo de aplicaciones.

Un modelo de programación llamado Map-Reduce se da a conocer en **(Castañè, Núneñ, Filgueira, & Carretero, 2012).** Dicho modelo es muy utilizado para para procesar grandes cantidades de datos principalmente por software científico, se centra específicamente en los sistemas multi-nucleos de tamaño mediano ya que estos son los más utilizados por los desarrolladores de aplicaciones científicas. Este modelo se orienta a tres recursos, el poder

computacional, comunicación y almacenamiento, en lo cual se presentan dificultades para poder satisfacer el rendimiento general del sistema.

En **(Kanewala & Bieman, 2014)** sus autores afirman que el software científico funciona como base para la toma de decisiones de gran importancia, es por esto que es fundamental la realización de pruebas sistemáticas a dichas aplicaciones con el fin de evitar tener que refutar afirmaciones, el objetivo de este artículo es identificar soluciones propuestas, y los problemas no resueltos a los que se enfrentan el software científico.

En **(Heaton & Carver, 2015)** los autores nos quieren dar a conocer que es cada vez más la demanda de los investigadores por crear software científico ya que les ayuda a reducir el tiempo, costo y riesgo que se requiere para llevar a cabo sus investigaciones físicamente.
Es por esto que se pretende que los científicos que desarrollen software adopten más las metodologías y prácticas de la ingeniera de software, para que aumenten la exactitud y eficiencia de los resultados de unas investigaciones, por otro lado que los desarrolladores de software también adapten más estas metodologías de ingeniería de software para que se ajuste de una mejor manera a las necesidades del desarrollo de software científico.

En **(Nazarenko & Prokhorov, 2015)** se expone que debido a que los problemas modernos de tipo científico y de ingeniería implican realizar múltiples cálculos, se presenta en este artículo un modelo de flujo de trabajo el cual contribuya a la automatización y ejecución de los procesos iterativos de cálculos complejos, esto se logra teniendo en cuenta la reutilización de los datos, el almacenamiento en cache, y ejecución en paralelo.

De otro lado, al revisar el documento escrito por **(Barrett et al., 2015)** sus autores nos dan a conocer que las aplicaciones científicas y de ingeniería son muy grandes, complejas y dinámicas, es por esto que existe un grupo de MiniApps creadas para simular las aplicaciones a escala real. Estas MiniApp están diseñadas para observar el rendimiento en tiempo de ejecución de las aplicaciones.

En el artículo de **(Li, Guzman, Tsiamoura, Schneider, & Bruegge, 2015)** se plantea que todo proyecto de software debe contar con unos requisitos de ingeniería, los cuales nos son tenidos en cuenta en el desarrollo de aplicaciones científicas, ya que los científicos no ven los beneficios y tampoco conocen mucho sobre estos. El objetivo del documento es ayudar a los científicos a adquirir los conocimientos previos sobre ingeniería, para lo cual se aplica una extracción de conocimientos y requisitos que los científicos aplican a sus aplicaciones sacándolos de informes de proyectos para posteriormente pasarlos a un lenguaje natural.

En su artículo **(Kelly, 2015),** la autora propone un modelo para el desarrollo de software basándose en la adquisición de conocimientos. En dicho documento se hace una comparación entre los modelos que existen de programación de ingeniería de software y lo forma en que desarrollan software los científicos. Se observa como los científicos desarrollan software fuera de los paradigmas establecidos para el desarrollo del mismo, y contrario a esto ellos buscan el conocimiento fuera de estas modelos establecidos por la ingeniera de software.

En el documento escrito por (**Arabnejad & Barbosa, 2016)** se muestra una estrategia de recursos para que nuestras aplicaciones científicas tengan un mayor éxito, está limitada por dos características que son el presupuesto y el tiempo, es decir que no se interesa por la

optimización del rendimiento pero si garantiza que el tiempo y presupuesto estipulado para cada puesto de trabajo no sea excedido y que la programación sea implementada con soluciones válidas.

Como resultado de la revisión podemos concluir, que el denominado software científico tiene unas características particulares por lo cual la comunidad científica, los desarrolladores e ingenieros de software ha encontrado la necesidad de mejorar los procesos de desarrollo, y aplicar metodologías adaptadas a este tipo de aplicativos, para mejorar la calidad del producto. Con este proyecto por tanto se espera aportar en esa dirección, realizando un análisis de las herramientas que pueden ser utilizadas para el desarrollo de este tipo de software de forma tal que investigadores y desarrolladoras tengan elementos de juicio para la selección de aquellas que se adapten a sus necesidades y permitan construir de mejor manera el producto adecuado.

4. Marco teórico

4.1 Software científico.

Es aquel que normalmente se realiza en ámbitos de investigación para estudiar, simular o mostrar algún comportamiento especifico, donde los desarrolladores en algunos casos son los mismos investigadores interesados en el producto y a su vez los usuarios finales.

Características.

- ✓ Surgen de la necesidad de evaluar una técnica de procesamiento de datos para un área de conocimiento específica.
- ✓ La complejidad es muy alta.
- ✓ Al iniciar con el proyecto no se tiene claro los requerimientos de la aplicación, estos se van aclarando a medida que avanza el proyecto.
- ✓ Las interfaces graficas son simples en cuando al manejo de la aplicación, ya para visualización de resultados si pueden ser más complejas.
- ✓ Es muy poco lo que se documentan por esto que sea difícil de entender para otros usuarios.
- ✓ El objetivo de estos no es la creación del software sino los resultados para analizar algún comportamiento específico.
- ✓ No se rigen por ningún estándar de diseño.

Cuando se resuelven estos problemas científicos utilizando cálculos numéricos se deben de seguir algunos pasos:

- ✓ Se debe tener una comprensión del problema para formular un modelo matemático.
- ✓ Utilizar métodos numéricos para poder resolver los modelos matemáticos.

- ✓ Seleccionar el lenguaje y herramienta de programación más adecuada.
- ✓ Hacer una implementación de los métodos numéricos en la herramienta seleccionada con anterioridad.
- ✓ Verificar que los datos arrojados sean resueltos de manera correcta.
- ✓ Por ultimo aplicar la herramienta para resolver el problema.

Estos pasos se realizan de una manera repetitiva donde se analizan los resultados y vuelve y se le hacen ajustes de ser necesario.

El software científico también cuenta con algunos requisitos:

- ✓ Tiene que ser matemáticamente correcto.
- ✓ Ser eficiente en cuanto a velocidad y uso de memoria.
- ✓ Debe ser de fácil mantenimiento y poder agregar nuevas funcionalidades.

Debe de cumplir muy bien con todos estos requisitos ya que un software que esta con errores matemáticos arroja resultados que son inútiles, su eficiencia debe ser máxima ya que pueden existir casos donde los resultados toman días o semanas para ser arrojados.

Dentro de sus aplicaciones podemos encontrar las simulaciones numéricas, análisis de datos y optimización.

- ✓ Simulación numérica.

  Está enfocada en reconstruir y entender algunos fenómenos conocidos como pueden ser los desastres naturales o hacer predicciones sobre tiempo climático.

- ✓ Análisis de datos.

  Resolver ecuaciones para obtener resultados de ciertas observaciones, por ejemplo la exploración de petróleo.

- ✓ Optimización.

  Mejoramiento de las aplicaciones en el modo de resolver los problemas o en las interfaces.

Se pueden clasificar por áreas de aplicación.

- ❖ MATEMATICAS.
  - ✓ **SageMath**

    Software de licenciamiento libre para trabajar todo lo relacionado a matemática elemental, avanzada, pura y aplicada. Está basada en Python como herramienta alternativa a Maple, Mathematica y Matlab.

  - ✓ **FreeMat**

    Software para el procesamiento de datos y prototipos de ingeniería y científicos, tiene similitud con software comercial como lo son Matlab pero con la diferencia de que cuenta con licenciamiento libre.

- ❖ FISICA.
  - ✓ **Termograf**

    Es un simulador termodinámico el cual fue desarrollado por el grupo de didáctica de la Termodinámica de la Universidad de Zaragoza, entre sus funciones podemos encontrar: cálculo de propiedades termodinámicas, crear gráficas y diagramas termodinámicos en tiempo real, cálculo de intercambiadores de calor y rendimiento de ciclos etc.

- ❖ QUIMICA
  - ✓ **Avogrado.**

Es un avanzado editor molecular, implementado para el campo científico de la química, está escrito en c++ y es de licenciamiento libre. Permite agregar mejoras en sus funcionalidades.

- ✓ **OpenBabel.**

  Caja de herramientas diseñara para hablar los muchos lenguajes de datos que tiene la química. Licenciamiento libre, multiplataforma, se trabaja por línea de comandos pero también cuenta con una interfaz gráfica.

- ✓ **BKChen**

  Programa para edición molecular en 2D con gran cantidad de funciones y características, fue escrito en Python, licenciamiento libre y multiplataforma.

❖ MEDICINA.

- ✓ **InVesalius**

  Software de licenciamiento libre para la reconstrucción de imágenes de tomografía computarizada y resonancia magnética. Se desarrolló en 2001 por el centro de tecnología de información Renato Archer de Brasil, está escrito en Python.

- ✓ **MedINRIA**

  Software para el procesamiento y visualización de imágenes médicas, es de uso libre, permite ver imágenes en 2D, 3D Y 4D, registro de imágenes, tractografia, etc.

- ✓ **EEGLAB**

  Es un conjunto de herramientas Matlab de licencia libre para procesar datos de la electroencefalografía, magnetoencefalografia etc.

- BIOLOGIA.
  - ✓ **Virtual Lab.**

    Es un microscopio virtual que brinda instrumentación científica simulada para estudiantes e investigadores, está respaldado por la nasa.
- ASTRONOMIA.
  - ✓ **Stellarium**

    Software de código abierto el cual muestre un cielo en 3D tal cual como se puede apreciar a simple vista con telescopios.

4.2 Metodología de desarrollo aplicada al desarrollo de software científico.

- **KDDP (Knowledge Discovery-Driven Project)**

Es una propuesta metodológica llamada Proyecto Guiado por el Descubrimiento del Conocimiento la cual está basada en un modelo de espiral propuesto por Boehm (Ingeniero informático).

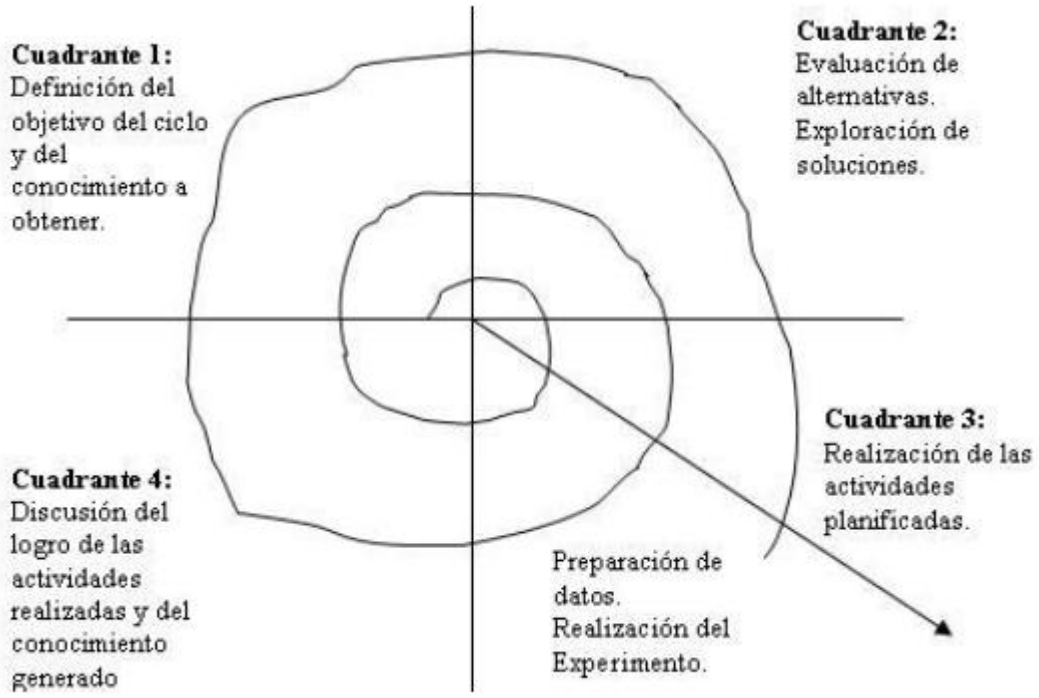


Figura 1. Modelo propuesto en espiral.
Somerville, Ian, Software Engineering, 2011.

Cuadrante 1: se tiene como tarea definir el objetivo del ciclo, la cual es liderada por un jefe de proyecto y la participación de sus demás integrantes de equipo de trabajo, esta tarea debe ser realizada entre 1 y 2 semanas. Al finalizar se debe contar con un informe donde contenga el objetivo establecido y las actividades que debe desempeñar cada integrante del equipo de trabajo.

Cuadrante 2: los integrantes deben de realizar una rápida investigación donde pueda calcular cuánto tiempo va a requerir para completar la tarea asignada. Se hace una valoración de las alternativas que se tienen para realizar cada tarea asignada y se procede a escoger una de estas.

Si en ese ciclo se requiere de implementar software se tendrá que definir los requerimientos que debe tener la aplicación, el tiempo que se tendrá como plazo para realizar la implementación y los roles que debe cumplir cada integrante.

Cuadrante 3: se debe tener las actividades desarrolladas por cada integrante, si en este ciclo había implementación se debe definir las pruebas a realizar y se realizan con los datos seleccionados y por último se analizan los resultados lo cual se hace hasta que el equipo este conforme con los resultados.

Cuadrante 4: se hace una discusión por parte de los integrantes del grupo de trabajo para ver si es necesario o no un nuevo ciclo en espiral.

Para el desarrollo de esta metodología también se debe contar con algunas recomendaciones:

- ✓ Realizar reuniones de 15 minutos donde se tratara lo que se lleva hecho, lo que falta por hacer y los problemas que ha tenido hasta ese momento.
- ✓ Para la parte de documentación se sugiere utilizar un sistema de control de versiones en este caso SVN (Subversión).
- ✓ Una matriz de trazabilidad la cual registre todas las versiones de software generado, los datos que se utilizaron y los resultados que arrojo, la fecha en que se realizó y en que computador se encuentra y así mismo el responsable de la creación de esa versión.

- ✓ Deben existir los roles y responsabilidades de cada integrante del equipo de trabajo, se sugiere el rol de Jefe de proyecto el cual es el que dirige y orienta el trabajo del equipo, Desarrollador-Documentador es que programa el código y lo documenta, Encargado de Datos y Tester debe saber que datos sirven para procesar, ejecutar las aplicaciones y hacer revisión de los resultados.

5. Herramientas de desarrollo.

5.1 Herramientas comerciales más destacadas.

Software comercial.

Aquel que es desarrollado con el fin de ganar dinero por el uso de este, donde su uso depende de los términos establecidos en su licencia, es para un usuario final, donde este no podrá tener todas las libertades que se tienen en el software libre como son copiarlo, distribuirlo, estudiarlo, y hacerle modificaciones, las ventajas con que cuenta este es que las interfaces graficas tienen un mejor diseño, también cuenta con mayor soporte y mayor compatibilidad con el hardware.

5.1.1 Matlab

Origen y desarrollo histórico.

Se origina de la necesidad de crear herramientas muy poderosas que contribuyan a la solución de problemas de cálculos muy complejos. El lenguaje M fue creado en 1970 para tener acceso a software matriciales como LINPACK y EISPACK sin tener que usar el lenguaje que poseían ellos en este caso Fortran.

Ya en 1984 se genera la primera versión la cual fue creada por Cleve Moler un matemático americano y programador de ordenadores con especialidad en análisis numérico, esto con la idea de utilizar paquetes de análisis numérico y algebra lineal escritos en fortran pero sin tener que utilizar dicho lenguaje. El nombre de Matlab proviene de las palabras matrix laboratory (laboratorio matricial), en 2004 se puede notar que ya tenía una gran cantidad de usuarios tanto en la parte académica como empresas y hoy en día es propiedad de MathWorks Inc.

Características del lenguaje.

- **Sintaxis**
  - Declaración de variables.
    - ✓ Variables escalares: se coloca el nombre que se le quiere asignar a la variable y un valor para esta. A=3;
    - ✓ Números simbólicos: representación exacta de un número se define con la palabra sym. Sym (1/5)
    - ✓ Variables simbólicas: se definen anteponiendo la palabra sym. Sym x

Estructuras de control

- **FOR**

```
>> for k=x:y:t
instrucciones
end
```

La letra “k” representa la variable índice del ciclo, “x” el valor inicial que va tomar para la primera vuelta, “y” el incremento que va adquiriendo k después de cada vuelta, “t” el valor final que puede tomar k para que termine el ciclo.

- **WHILE**

```
>> while(condicionlogica)
instrucciones
end
```

La condición lógica tiene que ser verdadera para que entre en el ciclo while, al momento de ser falsa termina el ciclo, se debe inicializar la variable antes del ciclo y cambiar su valor al hacer la iteración dentro del ciclo, ya que si no cambia el ciclo se mantendría infinito.

- **IF, ELSE**

```
>> if(condicion)
instrucciones de condicion verdadera
else
instrucciones de condicion falsa
end
```

Nos permite realizar ciertas instrucciones de acuerdo a decisiones que se den dentro del programa.

Expresiones.

- ✓ **(+)** suma
- ✓ **(-)** resta
- ✓ **(*)** multiplicación
- ✓ **(/)** división
- ✓ **(<)** menor que
- ✓ **(>)** mayor que
- ✓ **(<=)** menor o igual que
- ✓ **(>=)** mayor o igual que
- ✓ **(==)** igual que
- ✓ **(~=)** distinto que
- ✓ **(&)** operador lógico “Y”
- ✓ **(|)** operador lógico “O”
- ✓ **(~)** negación

Vectores

Se escriben como un conjunto de datos.

```
>> v=[1 2 3 4 5]

v =

     1     2     3     4     5
```

Para acceder a un dato del vector se coloca el nombre del vector y entre paréntesis el número de la posición que queremos obtener.

```
>> v(3)

ans =

     3
```

Otra manera de crear vectores es de forma vertical o columna para este caso se debe separar cada valor con un punto y coma.

```
>> u=[1; 2; 3; 4]

u =

     1
     2
     3
     4
```

Matrices

Se definen como un conjunto de datos donde cada fila va separada por un punto y coma.

```
>> A=[1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>>
```

Cadena de caracteres

Para crear una variable que sea de tipo cadena de caracteres el valor asignado debe estar encerrado por comillas simples, esas cadenas no funcionan para realizar cálculos matemáticos son utilizadas para facilitar la programación.

```
>> a='esto es una prueba 123'

a =

esto es una prueba 123
```

Trazado de gráficas.

Se cuenta con una función llamada plot donde nos permite hacer graficas de diferentes tipos.

Como ejemplo podemos graficar los valores de un vector los cuales pueden representar temperaturas tomadas cada tiempo determinado

```
>> v=[12.5 17.4 20 22 14.8 16]

v =

   12.5000   17.4000   20.0000   22.0000   14.8000   16.0000

>> plot (v)
>>
```

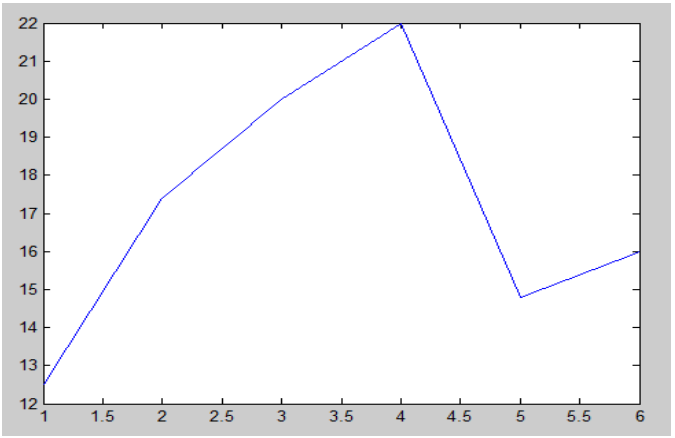


Figura 2. Muestra el resultado de la orden plot.

Para mejorar un poco el resultado de la gráfica se pueden agregar algunas órdenes para ponerle título a la gráfica así como nombre al eje vertical y horizontal para una mayor interpretación.

```
>> plot(v);xlabel('hora'); ylabel('temperatura'); title('Datos');
>>
```

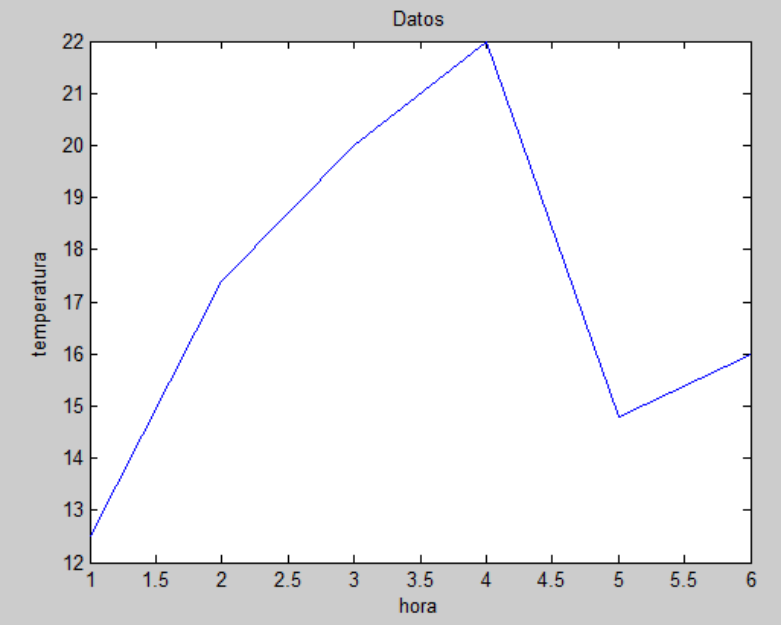


Figura 3. Grafica con título y nombre en el eje horizontal y vertical

Paradigmas De Programación.

Multiparadigma.

Traductor.

Matlab utiliza un lenguaje M el cual es propio de él, es interpretado.

Entorno de desarrollo.

Posee un entorno de desarrollo interactivo para facilitar el desarrollo de software.

Cuenta con cuatro tipos de ventanas principales: command Windows es la que se utiliza con mayor frecuencia, en ella se escriben los diferentes comandos u órdenes.

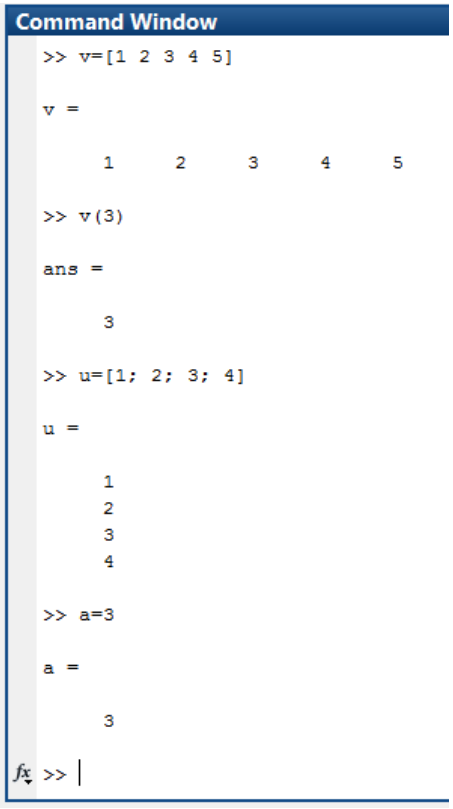


Figura 4. Ventana de comandos Matlab.

La siguiente ventana es la de workspace (espacio de trabajo) donde se muestra las variables con sus atributos que posee como lo son el nombre, tipo de datos, tamaño, y el valor que tiene cada uno en ese momento.

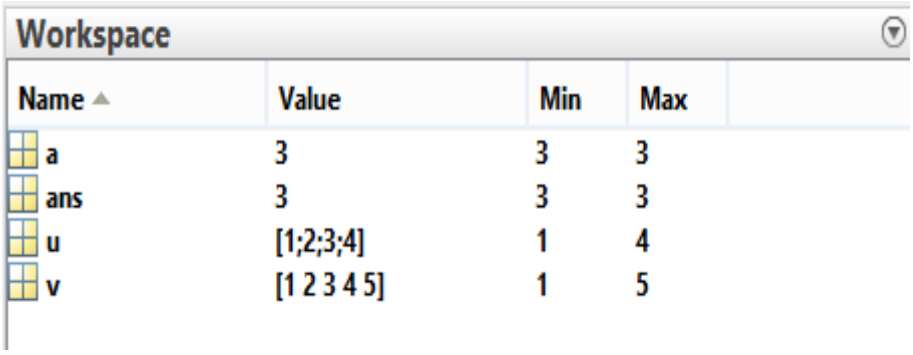


Figura 5. Ventana espacio de trabajo Matlab

Desde esta misma ventana se pueden manipular las variables para editarlas, eliminarlas o representarlas gráficamente.

También se cuenta con la ventana de command history (historial de comandos) en ella quedan almacenados en buffer todas las ordenes que hemos tecleado en la ventana de comandos, nos es de gran ayuda cuando obtenemos algún resultado que estábamos buscando pero no nos acordamos de que comando fue el que utilizamos.

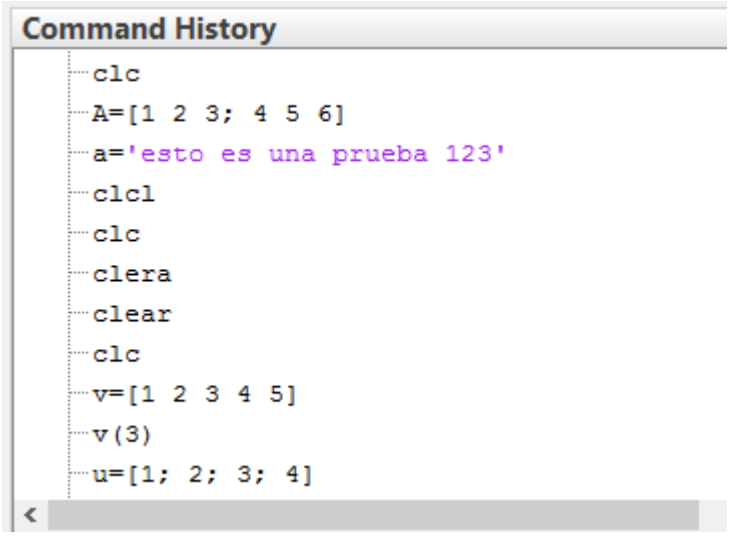


Figura 6. Ventana historial de comandos Matlab.

En esta ventana podemos elegir varios comandos y con uso del botón derecho del mouse se puede seleccionar la opción de volver a ejecutar los comandos, copiarlos, o crear un script o programa a partir de todas estas órdenes.

Por ultimo contamos con la ventada Edit que puede ser llamada desde la ventana de comando con la orden “edit”.

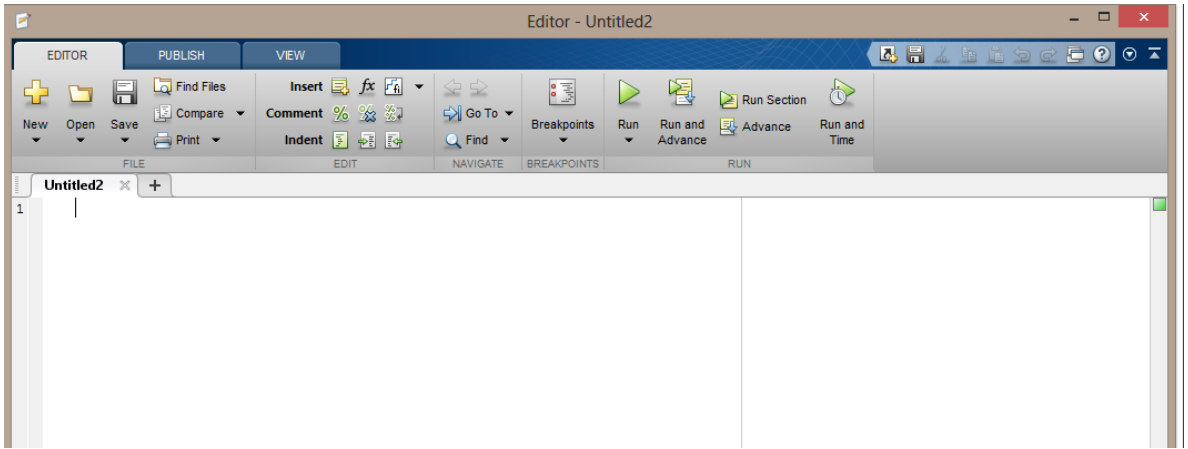


Figura 7. Ventana de edición para crear scripts Matlab

En ella se crea un programa completo con varios comandos donde se ejecutan secuencialmente siempre y cuando estén correctamente escritos, los programas se guardan con una extensión .m, para poner a funcionar estos programas el editor cuenta con un botón llamado run, o simplemente basta con escribir el nombre del archivo en la ventana de comando y este se ejecutara.

Librerías y apis.

- **Signal Processing Toolbox**

  Contiene gran cantidad de funciones para el procesamiento de señales.

  - ✓ Análisis de filtros digitales incluyendo respuesta en frecuencia, retardo de grupo, retardo de fase.
  - ✓ Implementación de filtros, tanto directo como usando técnicas en el dominio de la frecuencia basada en la FFT.
  - ✓ Procesamiento de la transformada rápida de Fourier FFT, incluyendo la transformada para potencias de dos y su inversa, y transformada para no potencias de dos.

- **Matlab C Math Library**.

  Esta librería nos proporciona la opción de utilizar funciones de Matlab y su compilador pero de una manera independiente con un lenguaje diferente en ese

caso C, para que los programadores acostumbrados a este lenguaje, puedan realizar grandes operaciones de alto rendimiento. Las funciones que se pueden utilizar son las siguientes.

- ✓ Algebra lineal
- ✓ Funciones matemáticas básicas y avanzadas.
- ✓ Operaciones lógicas y aritméticas.
- ✓ Manipulación de matrices y vectores.
- ✓ Analizar datos y realizar estadísticas.
- ✓ Gestión de memoria y errores.

- **Image Processing Toolbox**

  Cuenta con un grupo de funciones para el procesamiento de imágenes, gracias al entorno matemático que posee Matlab, ya que en si estas imágenes no son más que matrices. Algunas funciones son.

  - ✓ Filtros
  - ✓ Mejorar y retocar imágenes.
  - ✓ Operaciones de morfología, geométricas y de color.
  - ✓ Transformaciones 2D.

Interacción con otros lenguajes.

- ✓ Visual Basic .NET.
- ✓ Java
- ✓ C/C++
- ✓ Python

Comunidad de desarrollo.

Matlab cuenta con 225.000 colaboradores, se da respuestas por día de 120, las descargas por día llegan a las 25.000 y soluciones por día 730.

La participación por parte de personas ajenas a Matlab solo puede realizar preguntas para buscar soluciones a sus problemas, ganar algunos puntos de reputación los cuales le permite tener algunos privilegios de añadir etiquetas a sus preguntas, eliminar preguntas, respuestas y comentarios.

Otra opción es obtener y compartir aplicaciones donde a través de una cuenta creada en MaathWorks puede subir los archivos a un repositorio GitHub (plataforma de desarrollo colaborativo) y así mismo navegar en el para buscar y descargar.

Cody es un juego donde los usuarios pueden competir entre ellos resolviendo problemas, se puede aprender soluciones que brinde la comunidad, crear problemas para que sean resueltos por otros jugadores.

Soporte.

Matlab cuenta con una ventana de ayuda (f1) donde se encuentra documentación sobre todas las herramientas que posee.

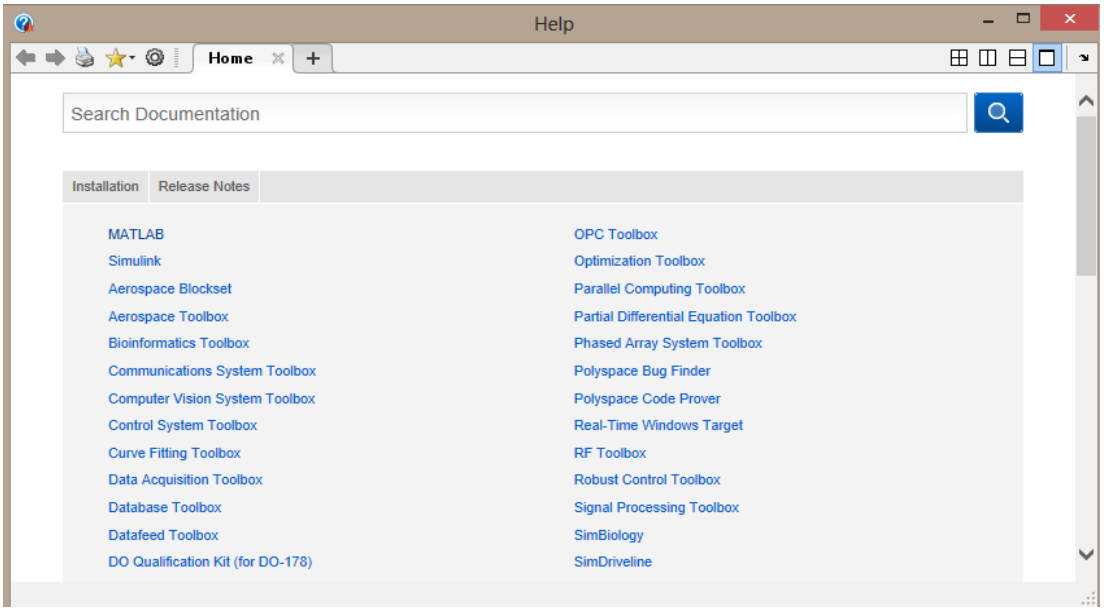


Figura 8. Documentación de Matlab.

Adicional a esto cuenta con su página oficial donde se pueden encontrar esta misma documentación. https://www.mathworks.com/help/matlab/, es tan grande la cantidad de personas que utilizan esta herramienta que es muy fácil poder encontrar información en la internet como son blogs, tutoriales en PDF, páginas de programación, se puede encontrar la información en idioma inglés y en español.

Áreas de aplicación.

Puede ser aplicado en varias áreas entre ellas podemos destacar las siguientes.

- ✓ sistemas de control
- ✓ mecatrónica
- ✓ análisis financiero
- ✓ procesamiento de señales
- ✓ procesamiento de imagen y video
- ✓ comunicaciones inalámbricas

- ✓ biología computacional
- ✓ robótica
- ✓ análisis de datos

5.1.2 Mathematica.

Origen y desarrollo histórico.

Es utilizado en áreas de ingeniería, científicas, matemática y áreas computacionales, fue creado por Stephen Wolfram, orientado hacia el álgebra computacional pero esto no significa que no pueda ser un gran lenguaje de programación para propósito general.

La primera versión se comercializo en el año 1988. La versión 10.3 tuvo su lanzamiento el 15 de octubre de 2015, siendo funcional en varias plataformas.

Características del lenguaje.

Sintaxis.

- Para declarar funciones su nombre debe iniciar con una letra mayúscula, los argumentos van entre corchetes [].
- Para evaluar una expresión se debe pulsar las teclas “shift + enter”.
- Para operaciones aritméticas elementales se representan igual que en c (+,-,*,/) y se debe tener en cuenta los paréntesis para agrupaciones matemáticas.
- Si se usa el ; al final de un comando el resultado de este no se mostrara.
- Distingue mayúsculas de minúsculas.
- Para graficar se utiliza la orden plot para funciones de una variable y plot3D para funciones de dos variables.
- Solucionar ecuaciones solve[{ecuaciones},{variables a despejar}]

- Existen constantes como son: pi (valor de pi), Infinity (infinito), E (euler), I (parte imaginaria).

Paradigmas de programación.

Cuenta con un lenguaje de programación multiparadigma, programación procedimental, programación funcional y programación basada en reglas.

Traductor.

Utiliza un lenguaje interpretado y esto se puede evidenciar en la forma de declarar las variables donde estas no se les asignan ningún tipo de dato o tamaño.

Entorno de Desarrollo.

- ✓ Editor de código fuente.
  - Cuenta con coloreado de sintaxis, información de errores.
  - Finalización de comandos.
  - Función de búsqueda
  - Resalte de soporte
  - Información de uso de comandos.

Cuenta con herramientas para depuración y pruebas unitarias para el código mostrando las fallas encontradas, que código pasó la prueba y cual no.

Comunidad.

Cuenta con un tablero de discusiones donde se pueden entrar a estas discusiones y dar sus aportes estando registrado con anterioridad. Así mismo existen grupos sobre todos los temas relacionados a wólfram, se puede ser incluido en estos grupo y participar, otra sección con que se cuenta es la de personas, es una lista alfabética con los nombres de todos los principiantes, investigadores, ingenieros, desarrolladores y diseñadores de la

comunidad de wolfran, se puede ver alguna información sobre estos por ejemplo de que localidad son, el tiempo que llevan participando, algo de información personal y aportes que han hecho a la comunidad.

Áreas de Aplicación

- ✓ Ingeniería química
- ✓ Sistemas de control
- ✓ Ingeniería eléctrica.
- ✓ Procesamiento de imágenes
- ✓ Ingeniería industrial
- ✓ Ciencias materiales
- ✓ Ingeniería mecánica
- ✓ Investigación de operaciones
- ✓ Óptica
- ✓ Ingeniería de petróleos
- ✓ Bioinformática
- ✓ Imágenes medicas
- ✓ Economía
- ✓ Astronomía
- ✓ Ciencias biológicas
- ✓ Ciencias ambientales
- ✓ Geo ciencias
- ✓ Ciencias sociales.

5.2 Herramientas libres.

Software libe.

Se debe entender que software libre no se refiere a que siempre sea gratis, este también puede tener un pago por el mismo, lo que lo hace libre es que el código fuente utilizado para desarrollarlo sea de libre acceso a los usuarios permitiéndole a estos utilizarlo, copiarlo, distribuirlo, estudiarlo, y hacerle modificaciones, es decir contar con las libertades que se debe tener,

- Libertad de utilizar el software con cualquier propósito (Libertad 0).
- Libertad de poder tener acceso al código fuente para observar cómo trabaja el software y así realizarle cambios para ajustarlo a sus necesidades (Libertad 1)
- Libertad de compartir copias del software Original (Libertad 2).
- Libertad de compartir copias del Software con modificaciones, permitiendo también el acceso al código fuente de este (Libertad 3).

5.2.1 Scilab

Origen y desarrollo histórico.

Scilab inicia desde los años 80 con un software llamado Blaise, creado en el Institut National de Recherche en informatique et en Automatique por los desarrolladores Francois Delebecque y Serge Steer con el fin de hacer una herramienta para el control automático de los investigadores, y tomando como referencia el software de Matlab.

Cuatro años más tarde su nombre cambio a Blaise Basile y fue distribuido por SIMULOG.

Iniciando los años 90 es donde realmente surge el nacimiento de Scilab debido a que Simulog no siguió distribuyendo Basile y es aquí donde cambia el nombre y empieza a ser desarrollado por seis investigadores.

Se empieza a distribuir Scilab como software libre y de código abierto en su versión 1.1 se podía conseguir en un sitio ftp anónimo desde 1994. Llego hasta la versión 2.7 en el año 2002 donde se distribuían el código fuentes en internet.

A principios de los años 2003 se decide crear una comunidad de Scilab para poder tener apoyo de empresas y otras organizaciones, para así asegurar que se pudiera tener futuros desarrollos y mantenimientos al software.

Por ultimo en 2010 se funda la empresa de Scilab para poder tener garantizado un buen futuro para el software.

Características del lenguaje

Sintaxis.

Scilab cuenta con un lenguaje de programación propio, enfocado en el uso de matrices y vectores.

// Para realizar comentarios

clc realiza una limpieza de la pantalla

disp (“Hola Mundo”) imprime el mensaje que está dentro de las comillas.

Para las variables no se les define su tipo ni tamaño, estas se ajustan de acuerdo al valor que se le asigne.

```
-->a=5
 a  =

    5.

-->b="hola mundo"
 b  =

 hola mundo

-->
```

Se manejan también algunas constantes entre ellas tenemos.

- ✓ %i simboliza la parte imaginaria
- ✓ %pi simboliza el valor de 3.1416
- ✓ %e simboliza el valor de Euler 2.7182
- ✓ %t simboliza el valor lógico verdadero
- ✓ %f simboliza el valor lógico falso

Entrada y Salida de Datos.

```
-->valor_numerico = input ("ingrese numero")
ingrese numero 5
 valor_numerico  =

    5.

-->valor_texto=input ("ingrese texto","s")
ingrese texto hola mundo
 valor_texto  =

  hola mundo
```

Para mostrar datos de salida existen dos funciones, disp que muestra los datos pero sin ningún tipo de formato y printf permite formatear la salida.

```
-->disp("Hola Mundo")

 Hola Mundo

-->printf('Texto=%s Numero%d\n',"hola mundo",2345)
Texto=hola mundo Numero2345
```

%s mostrar variables de tipo cadena de caracteres.

% d mostrar variables de tipo numérico.

%c mostrar variables de tipo carácter.

\n nos da un salto de línea

Operadores De Comparación.

Operador	Descripción
<	Menor que
<=	Menor o igual que
==	Igual
>	Mayor que
>=	Mayor o igual que
~= o <>	no es igual o diferente

Tabla 1. Simbología y significado de los Operadores de comparación

Operadores Lógicos

Operador	Descripción
&	Operador Y
|	Operador O
~	Negación

Tabla 2. Simbología y significado de los operadores lógicos.

Funciones Elementales

Función	Descripción
sqrt(x)	Calcula la raíz cuadrada de x
abs(x)	Calcula el valor absoluto de x
modulo(x,y)	Calcula el residuo de dividir x en y
sin(x)	Calcula el seno de x, en radianes
cos(x)	Calcula el coseno de x, en radianes
tan(x)	Calcula la tangente de x, en radianes

Tabla 3. Forma de escribir algunas de las funciones.

Estructuras de control

If.

Nos permite ejecutar ciertas instrucciones si se llega a cumplir la condición establecida dentro del if.

```
if (condicion) then
    instrucciones
end
```

```
if (condicion) then
    instrucciones
else
    instrucciones
end
```

Select

Se tiene una variable con un valor y esta es comparada con cada caso del select y si encuentra un valor igual se ejecutan las instrucciones de ese caso.

```
select variable
case 1 then
    instrucciones...
case 2 then
    instrucciones...
    
    
else
    instrucciones...
end
```

For

Se repiten las instrucciones hasta que se llegue a un límite establecido.

```
for valor inicial=limite
    instrucciones...
end
```

While.

Se repiten las instrucciones hasta que condición dada deje de ser verdadera.

```
while condicion
    instrucciones...
end
```

Vectores.

Los vectores son colecciones de datos del mismo tipo, se pueden crear de dos tipos, horizontales y verticales.

```
-->v1=[1 2 3 4 5]
 v1  =

    1.    2.    3.    4.    5.

-->v2=[1; 2; 3; 4; 5]
 v2  =

    1.
    2.
    3.
    4.
    5.
```

Para acceder a uno de sus valores se escribe el nombre del vector y el número de la posición. v1(3) = 3

Matrices

Son conjunto de datos del mismo tipo donde se necesitan índices para acceder a cada uno de sus elementos.

```
-->M=[1 2 3; 4 5 6]
 M  =

    1.    2.    3.
    4.    5.    6.
```

Para acceder a unos de sus elementos se escribe el nombre de la matriz y el número de la fila y columna en que se encuentra. M(2,1) = 4

Declarar Funciones

Para funciones que retornar algún valor.

```
function variable_retorna=nombre_funcion(argumentos)
    instrucciones de la funcion....
endfunction
```

Funciones que no retornan nada.

```
function nombre_funcion(argumentos)
    instrucciones de la funcion...
endfunction
```

Paradigmas De Programación.

Programación basada en procedimientos.

Traductor.

Maneja un lenguaje propio interpretado.

Entorno De Desarrollo.

La ventana principal de Scilab es la consola es allí donde se definen los comandos que vamos a ejecutar, se escriben y se ejecutan al presionar la tecla enter.

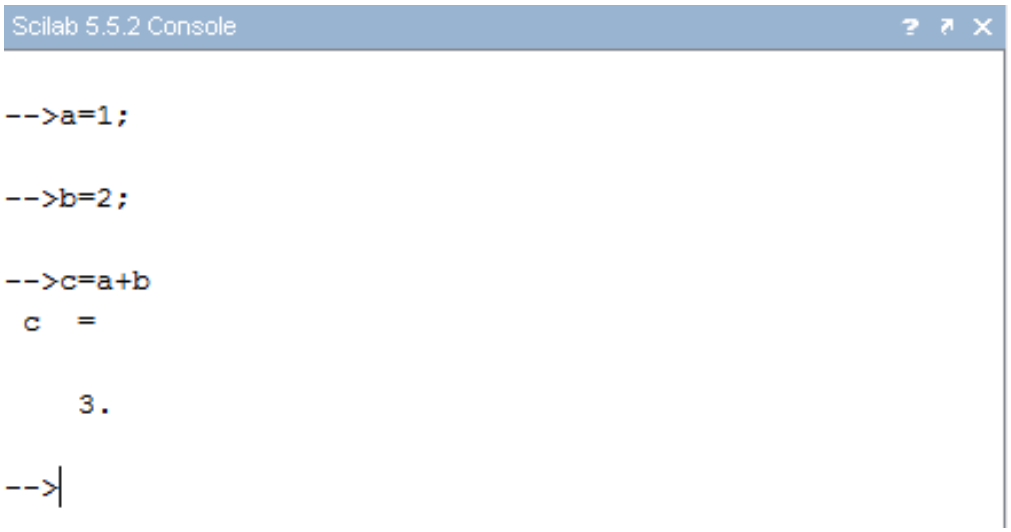


Figura 8. Consola de Scilab.

Después de esto cada ejecución queda registrada en un historial de comandos, desde esta ventana podemos volver a ejecutarlos.

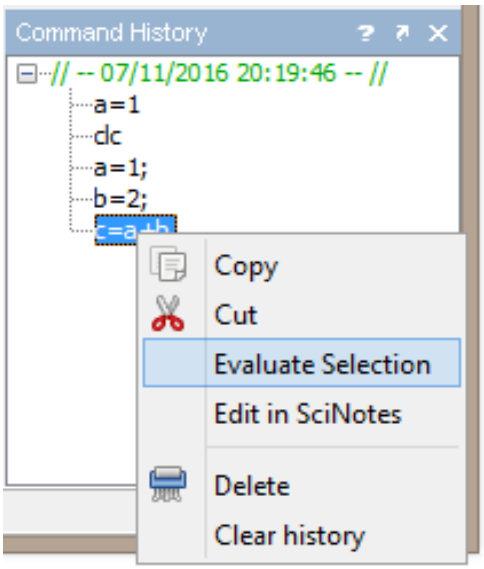


Figura 9. Ventana historial de comandos Scilab.

Otra de las ventanas es el navegador de archivos, donde podemos observar en qué lugar estamos trabajando y nos permite cambiar este sitio.

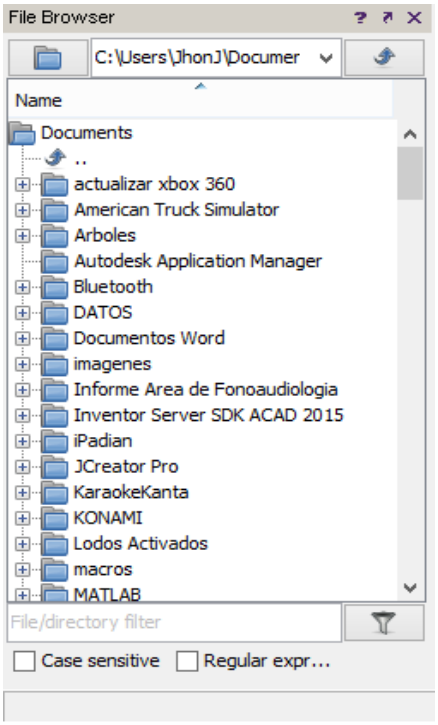


Figura 10. Ventana navegador de archivos Scilab.

Por ultimo tenemos la ventana de graficas que es invocada mediante la función plot.

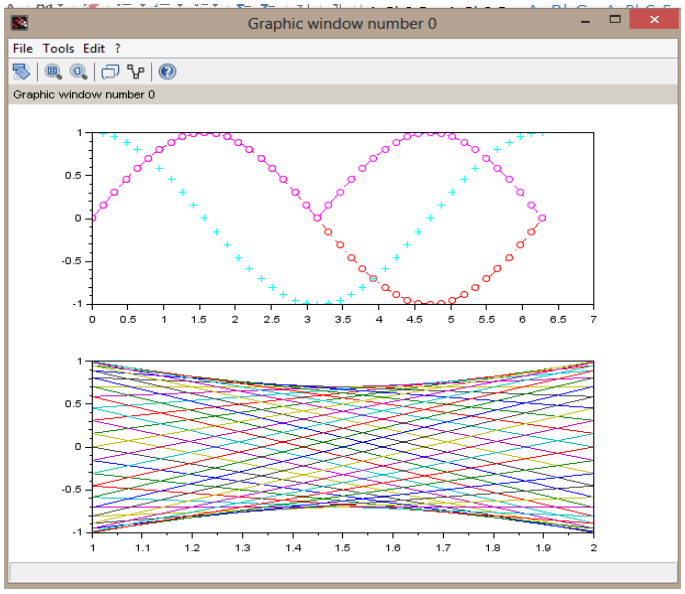


Figura 11. Ventana donde se grafican las funciones en Scilab.

Librerías y apis.

- ✓ **Guibuilder**

  Nos permite crear interfaces graficas rápidamente, y el código de cada componente de la interfaz se genera de manera automática. Se invoca en la consola de Scilab con el comando "guibuilder".

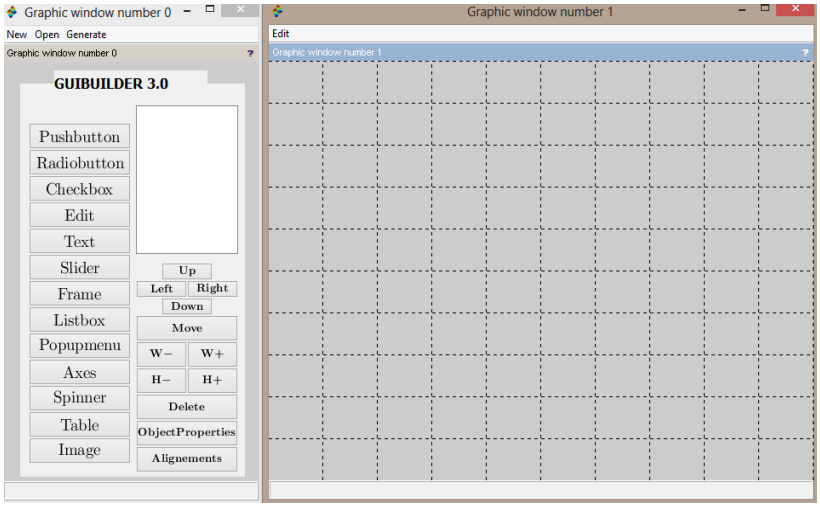


Figura 12. Ventana para crear interfaces graficas Scilab.

Interacción con otros lenguajes.

- ✓ Python
- ✓ Perl
- ✓ C
- ✓ C++

Comunidad

Si se desea participar en el proyecto de Scilab se cuenta con varias opciones.

- ✓ **Desarrollo de código**

  Si las modificaciones que se tienen son menores en lo que tiene que ver con documentación se debe enviar esta petición a los desarrolladores por medio de correo donde estos analizan la información y si es relevante ellos proceden a aplicar las modificaciones, para esta no se necesita la firma de ningún tipo de acuerdo de contribución con Scilab.

  Por el contrario si las modificaciones ya tienen una mayor importancia que implique modificar el código fuente de Scilab se debe firmar un acuerdo de contribución con Scilab, donde este le da acceso de escritura sobre el código fuente de Scilab, para que corrija errores, añada o elimine funcionalidades, todo esto se pueden hacer después de que se haga una propuesta sobre lo que se pretende realizar y sea evaluada.

  Para pertenecer a ese grupo de colaboradores se debe enviar un correo a contribute@scilab-enterprises.com explicando porque quiere pertenecer y que ideas tiene para hacer en Scilab y ellos responderán con los pasos necesarios que se deben seguir y los documentos legales que lo acrediten como colaborador de

Scilab. Debido a que la comunidad es inglesa y francesa se debe tener dominio de estos idiomas para poder ser colaborador.

- ✓ **Traductor**

  La función como traductor de Scilab es traducir toda la interfaz de un idioma a otro ya que es utilizado por personas de todo el mundo. También puede hacerse la traducción de toda la documentación de ayuda con que cuenta Scilab.

- ✓ **Desarrollar Módulos**

  Consiste en desarrollar módulos externos para Scilab para poder así extender las áreas de aplicación de Scilab, estos módulos pueden ser alojados en ATOMOS Portal donde si cuentan con la autorización de quien los desarrollo pueden ser accedidos por usuarios que así lo deseen mediante la consola de Scilab utilizando el gestor de módulos ATOMS.

Soporte

Scilab cuenta con documentación extra sobre el uso de muchas funciones explicando cómo se hace el llamado, para que se utilizan y ejemplos, lo que lo hace un poco incómodo de entender en este caso el idioma en que está escrito (ingles) dificultando un poco el entendimiento.

Fuera de Scilab cuenta con una gran cantidad de tutoriales, videos, donde explican la mayoría de funcionalidades que posee y la solución de algunos problemas que pueda presentar.

Áreas de aplicación.

- ✓ Cálculos de ingeniería
- ✓ Análisis de señales

- ✓ Análisis estadísticos
- ✓ Análisis geométricos.

5.2.2 Python

Origen y Desarrollo Histórico.

Todo comenzó con un proyecto de desarrollar un sistema operativo llamado Amoeba para el cual se tenía un lenguaje de programación llamado ABC, quien empezó con este desarrollo fue el investigador holandés llamado Guido Van Rossum, a medida que avanzaba en este sistema operativo fue encontrando muchas fallas y limitaciones en el lenguaje ABC es por esto que decide crear un nuevo lenguaje para poder desarrollar Amoeba y es de ahí donde nace Python alrededor de los años 80, y se decide liberar el intérprete de este lenguaje bajo una licencia open source propia.

A principios de los años 2001 se decide cambiar el tipo de licencia que sea GPL y es de ahí donde nace Python Software Foundation Licence, lo que nos da la posibilidad de hacer modificaciones y desarrollar código derivado del código fuente.

El nombre del Python fue sacado de una serie de televisión llamada Monty Python´s Flying Circus, ya que su creador era amante a este programa y además quería que su lenguaje fuera divertido al momento de utilizarlo.

Características del lenguaje

Sintaxis.

Cometarios: se utiliza el carácter # antes del texto que se va a comentar.

```
>>> # Esto es un comentario.
```

Asignación: se utiliza el singo =

```
>>> a=3+2
```

Definir funciones:

```
def nombrefuncion(argumentos):
    instrucciones..
    instrucciones..
```

Las instrucciones se toman como un bloque las cuales deben de tener los mismos espacios, es decir estar alineadas para que no presente errores la función.

IF.

```
if condicion:
    instrucciones
    elif condicion:
        instrucciones
        else:
            instrucciones
```

FOR

```
for i in range(5):
    print(i)
```

WHILE.

```
cont=0
while cont<10:
    cont=cont+1
    print(cont)
```

Arreglos

En Python nos permite agregar valores de distinto tipo, para acceder a alguno de sus valores se hace de la misma manera que en otros lenguajes y es indicando la posición del arreglo.

```
>>> arreglo=[1,"hola",4.8]
>>> print(arreglo[2])
4.8
```

Paradigmas de programación.

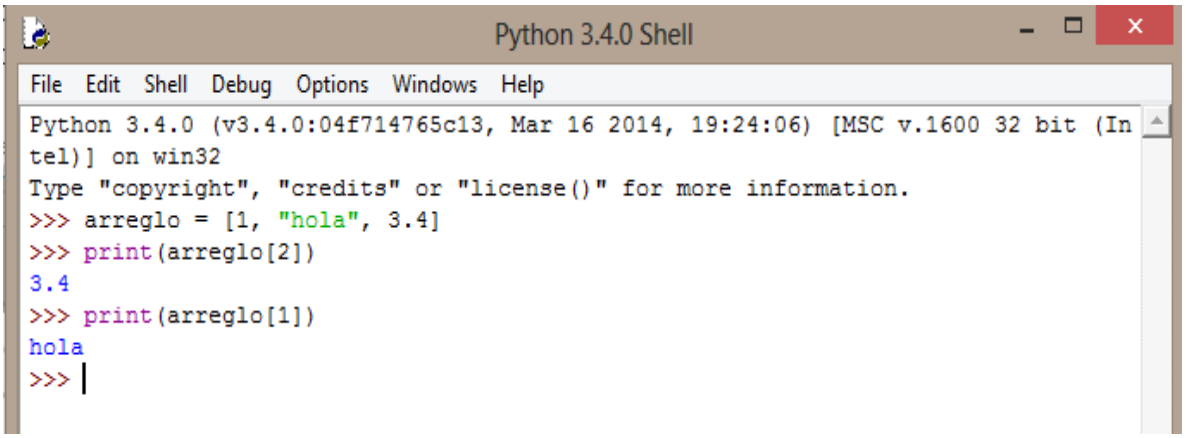
Python tiene multi-paradigmas de programación, permitiendo así la programación orientada a objetos, estructurada y funcional.

Traductor.

Maneja un lenguaje interpretado lo que nos dice que no es necesario hacer declaración de variables antes de utilizarlas y tampoco necesita de ser compilado. Cuando ejecutamos la primera vez un script este se compila y genera un bytecode el cual después es interpretado. Los errores que existan los encuentra en el tiempo de ejecución.

Entorno de Desarrollo.

Su entorno de desarrollo es muy sencillo donde existe una ventana tipo consola donde se escriben los comandos y se arrojan los resultados.



13. Consola de Python.

Desde esa primer ventana se puede abrir un editor para crear scripts, se guardan con extensión .py. Esta ventana también cuenta con la opción de chequear el código escrito y de ejecutarlo.

Figura 14. Ventana para crear scripts de Python.

Además del entorno de desarrollo que trae Python por defecto se puede hacer uso de muchos más IDEs que sirven para enlazarse con Python. Entre ellos se pueden destacar Eclipse, Netbeans, Ninja, Spyder que es un entorno de desarrollo de Python orientado a científicos, PyCharm, etc.

Librerías y Apis

- ✓ Pillow: se utiliza para trabajar con todo lo que tiene que ver con imágenes.
- ✓ Numpy: es la mayor herramienta para trabajar todo lo que tiene que ver con funcionalidades matemáticas avanzadas.
- ✓ Scipi: colección de algoritmos y herramientas matemáticas para uso de los científicos.
- ✓ Matplotlib: se utiliza para graficar similar a Matlab, muy útil para científicos que trabajan analizando datos.
- ✓ PyQt: cuenta con un conjunto de herramientas para crear interfaces graficas de Python con gran facilidad creando de manera automática el código de cada componente de la interfaz.
- ✓ SQLAlchemy: sirve para gestionar bases de datos.

- ✓ Pygame: para la creación de juegos en 2D.
- ✓ Pyglet: animación y creación de juegos en 3D, uno de sus juegos mas importantes creado es Minecraft.
- ✓ BioPython: computación biológica.
- ✓ RPy: combinar Python con el r para sistemas estadísticos.

Comunidad.

Para poder contribuir con Python se requiere de tener un sistema de control de versiones multiplataforma en este caso Mercurial, se puede obtener el código fuente de Python del siguiente repositorio https://hg.python.org/cpython, luego de esto existe una guía para configurar este código con visual studio que en este caso es el programa que utilizan para trabajar Python.

También se tiene lo que son las preguntas frecuentes, participación de conferencias sobre temas de Python para las cuales se debe realizar una previa inscripción si se desea participar de estas.

Soporte

Debido a que es un gran lenguaje y muy utilizado para diversas áreas se cuenta con una gran cantidad de tutoriales, documentación en la página principal de este, aunque en algunos casos la información está en el idioma ingles se puede decir que cuenta con un muy buen soporte.

Áreas de aplicación

- ✓ Computación científica de alto rendimiento
- ✓ Educación
- ✓ Automatización industrial

- ✓ Desarrollo web
- ✓ Biología
- ✓ Física
- ✓ Medicina
- ✓ Ingeniería civil
- ✓ Astronomía
- ✓ Matemáticas
- ✓ Química

5.2.3 Octave

Origen y desarrollo histórico.

Octave fue desarrollado en el año 1988 como software de apoyo para el estudio de un reactor químico desarrollado por James B. Rawlings, donde se desarrollaban herramientas que sirvieran para solucionar problemas del reactor, este fue creciendo agregándole más funcionalidades que fueran flexibles para el aprendizaje de estudiantes en temas de ingeniería química.

En el año 1992 octave dejo de ser una herramienta limitada que trabajaba solo con ingeniería química, se decidió complementar octave de una manera que pudiera resolver cualquier tipo de problemas y es así como hoy en día se está utilizando para temas de investigación y aplicaciones comerciales.

Características del leguaje.

Sintaxis

La sintaxis que se maneja en octave es muy similar a la utilizada por Matlab.

Las variables se definen sin necesidad de indicarles algún tipo de dato o tamaño.

Ventana de comandos

```
>> a=45
a =  45
>>
```

Se puede realizar rangos de números.

```
>> 1:10
ans =

    1    2    3    4    5    6    7    8    9   10

>> 1:2:10
ans =

    1    3    5    7    9

>>
```

Definir una matriz.

Ventana de comandos

```
>> a=[1 2; 3 4]
a =

   1   2
   3   4
```

Se escribe el nombre de la matriz y entre [] se introducen los datos, donde los datos de una misma fila se separan por espacio y para una nueva fila se utiliza el punto y coma.

```
>> a(1,1)
ans =  1
>> a(2,2)
ans =  4
>>
```

Para obtener el elemento de una matriz se indica el nombre de la matriz y entre paréntesis los índices de la fila y la columna del elemento que queremos obtener.

Sentencia if.

Ventana de comandos

```
>> a=3
a =  3
>> if a=3
disp("es igual")
end
es igual
>>
```

Sentencia while.

```
>> a=3
a =  3
>> while a<6
disp(a)
a=a+1
endwhile
 3
a =  4
 4
a =  5
 5
a =  6
>>
```

Sentencia for

```
>> for a=1:5
disp(a)
endfor
 1
 2
 3
 4
 5
```

Definir una función.

```
function variable_retornada=nombre_funcion(argumentos)
instrucciones de la funcion...
endfunction
```

Para graficar se utiliza la función plot.

```
>> plot (sin (0:90))
>>
```

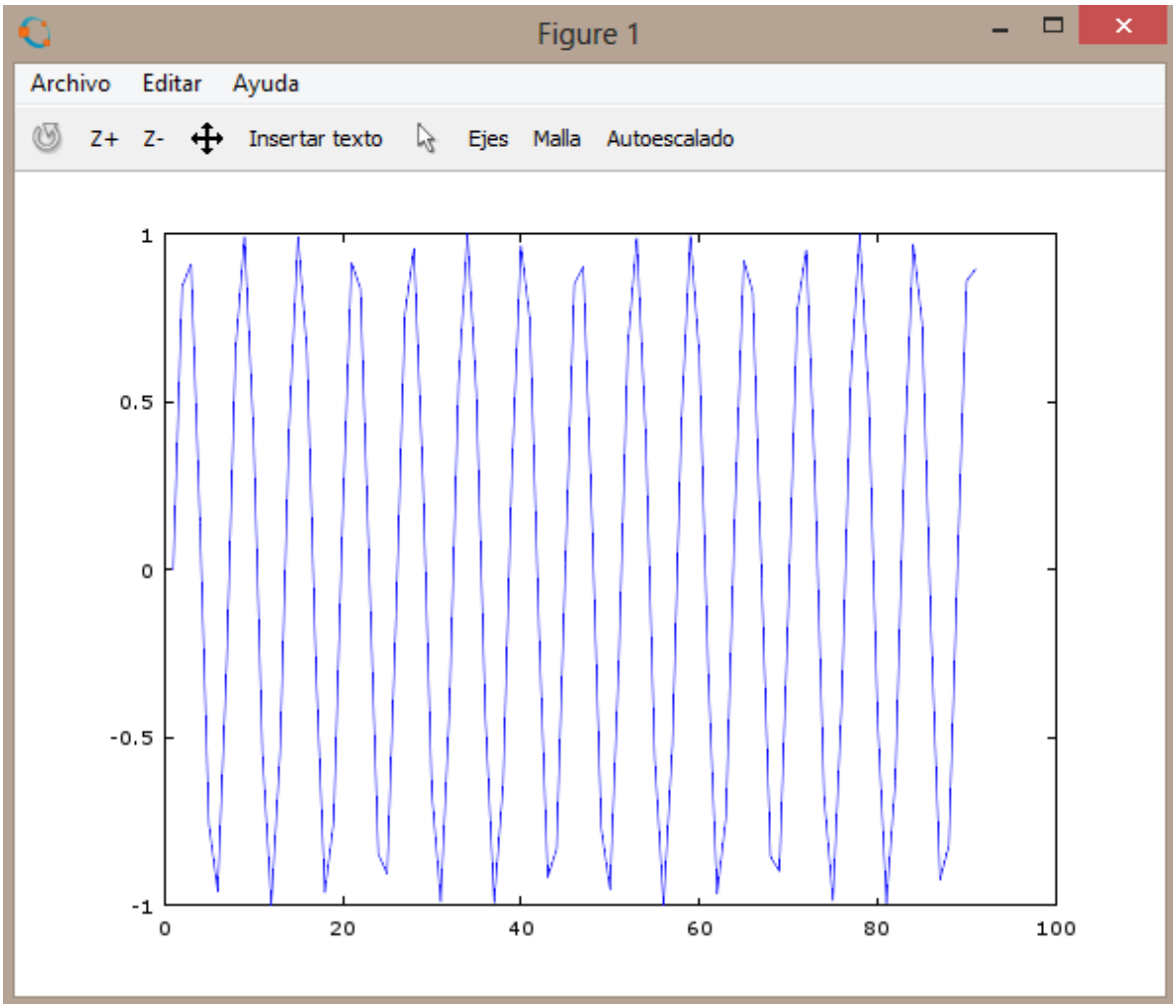


Figura 15. Ventana donde se grafican las funciones de octave.

Paradigmas de programación.

Utiliza la programación estructurada, utilizando funciones de c++, selección de if y switch e interacción de bucles for y while.

Traductor.

Utiliza un intérprete que es propio de su lenguaje, permite una ejecución interactiva o por lotes.

Entorno de Desarrollo.

Cuenta con la ventana de comandos donde se escriben las ordenes y se ejecutan con enter, otra ventana es la de historial de comando la cual almacena todos los comando que hemos ejecutando permitiendo también volver a ejecutarlos, está la ventana de espacio de trabajo donde quedan almacenada las variables que hemos creado mostrando el nombre, tipo, y valor que tiene, por último el explorador de archivos donde nos indica en que ruta estamos trabajando.

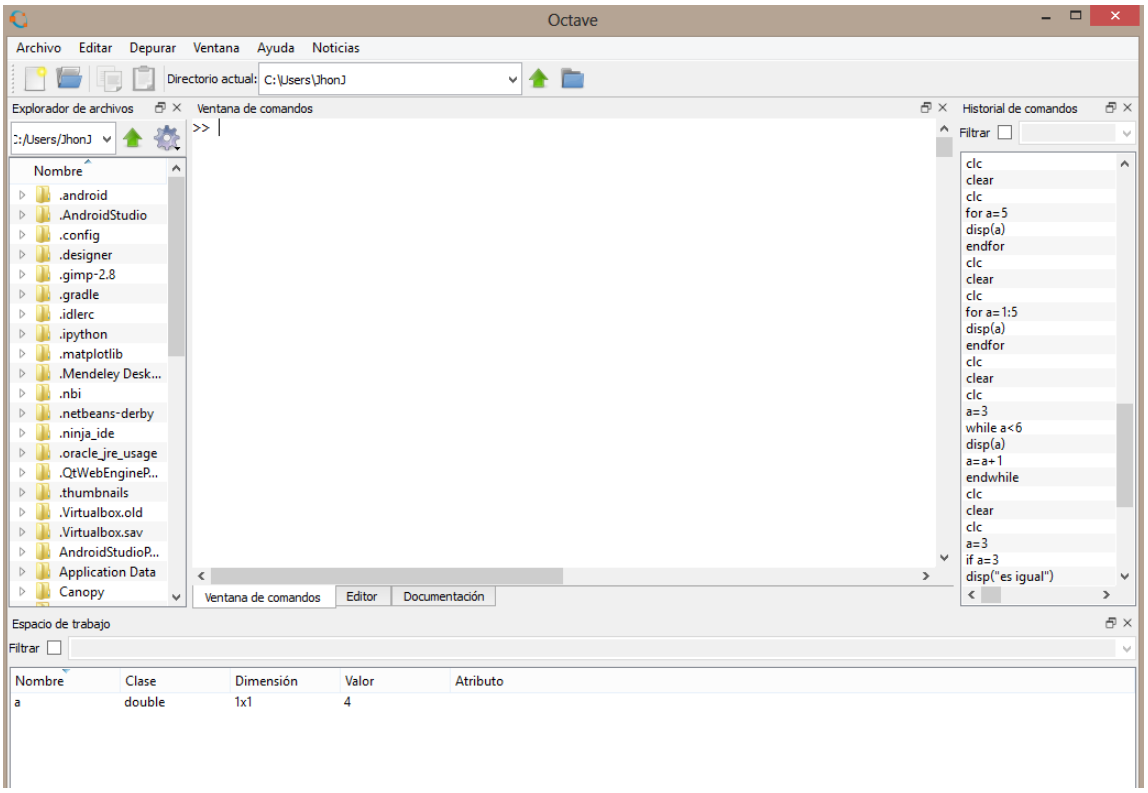


Figura 16. La interfaz que trae octave para interactuar.

Librerías y Apis

- ✓ Database: interfaz para trabajar bases de datos sql, solo postgreSQL.
- ✓ Dicom: comunicación entre archivos dicom utilizados en medicina.
- ✓ Financial: para manipulación financiera, funciones de gráficas.

- ✓ Image: funciones para el procesamiento de imágenes.
- ✓ Optics: funciones para aspectos de óptica.
- ✓ Parallel: funciones para la ejecución en paralelo
- ✓ Vrml: graficos en 3D utilizando VRML(lenguaje para modelado de realidad virtual)

Comunidad.

Esta comunidad cuenta con varias opciones de apoyo para el continuo desarrollo de octave entre estas encontramos la de reportar fallos donde primero se debe ver si el fallo no ha sido reportado por otra persona o si ya tiene solución, después de este primer paso se procede a llenar un formulario donde se reporta el fallo y si se tiene una solución posible a este.

Otra manera de contribuir con octave es ingresar al repositorio que encontramos en http://www.octave.org/hg/octave con que cuentan a través de mercurial donde se hace una subversión o copia del código fuente original de octave, se cuenta con una guía de como poder hacer un correcto uso de este código fuente, nos permite realizar módulos adicionales que creamos le sirven a octave y estos son registrados en las subversiones que manejan con mercurial y los desarrolladores observan los módulos y deciden si son de gran ayuda o no para mejorar el funcionamiento de octave.

Soporte

Cuenta con gran cantidad de material de ayuda para sus usuarios y su principal idioma es español.

Áreas de aplicación.

- ✓ Procesamiento de imágenes

- ✓ Algebra lineal
- ✓ Calculo con variables complejas
- ✓ Electrónica
- ✓ Control automático
- ✓ Medicina
- ✓ Circuitos
- ✓ Señales
- ✓ Comunicaciones.
- ✓ Estadística
- ✓ Economía

5.2.4 R

Origen y desarrollo histórico.

Empieza en los años 90 cuando dos profesores de estadística Ross Ihaka y Robert Gentleman, de donde proviene su nombre por las iniciales de estos dos profesores, se deciden a desarrollar un software de estadística para enseñarles a sus estudiantes.

En 1995 se tenía la primera versión donde se decidió que este fuera un software gratuito y de código abierto para que su código fuente estuviera al alcance de cualquier usuario permitiéndole hacer modificaciones de acuerdo a las preferencias de cada quien.

Después de que esta primera versión fuese utilizada por varios estadísticos y expertos en computación y con el apoyo de ellos en el año 2000 es donde aparece la primera versión oficial para el público en general (v.1.0).

Características del lenguaje

Sintaxis.

- Se distinguen las mayúsculas de las minúsculas.
- Para asignar un valor a una variable se utiliza <-. también se pude utilizar el =.

```
> x=2
> x
[1] 2
> s<-2
> s
[1] 2
```

- Para saber que contiene una variable se escribe el nombre de la variable.
- Para crear vectores se puede hacer uso del comando c donde este combina varios objetos en uno solo.

```
> X=c(3, 5, 20)
> X
[1]  3  5 20
>
```

- Para imprimir datos de utiliza el comando print.

```
> print ("hola mundo")
[1] "hola mundo"
```

- Estructuras de programación.

  Ciclo For.

```
> for(i in 1:10){
+ print(i)}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
>
```

Ciclo while.

```
> a=0
> while(a<5){
+ print(a)
+ a=a+1
+ }
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
>
```

Condicional If.

```
> a=5
> if(a==5)
+ print(a)
[1] 5
>
```

Crear funciones.

```
> nombre=function(argumentos)
+ comandos
>
```

Paradigmas de programación.

Multiparadigma (imperativo, procedural, orientado a objetos, vectorial, funcional.)

Traductor.

Maneja un lenguaje interpretado.

Entorno de Desarrollo.

- ✓ Cuenta con un manejo eficiente de los datos.
- ✓ Colección de operadores para cálculos de matrices.
- ✓ Gran colección de herramientas con gran coherencia para el análisis de datos.

- ✓ Funciones gráficas para el análisis y visualización de los datos.
- ✓ Lenguaje de programación simple y eficiente cuenta con condicionales, ciclos repetitivos, permite funciones de recursividad y entrada y salida de datos.

Se puede trabajar la programación R cuenta con dos opciones una de ella es la consola de R donde se escriben los comandos, solo se puede ejecutar una línea de comando al mismo tiempo por esto se utiliza para hacer cosas muy simples.

Para crear ya funciones o programas completos existe un editor de scripts donde se pueden agregar múltiples líneas de comandos, y este cuenta con un icono de ejecución del código con la abreviatura CTRL+R. los resultados al ejecutar estos script son mostrados en la ventana de consola de R, por tanto se deben mantener abiertas estas dos.

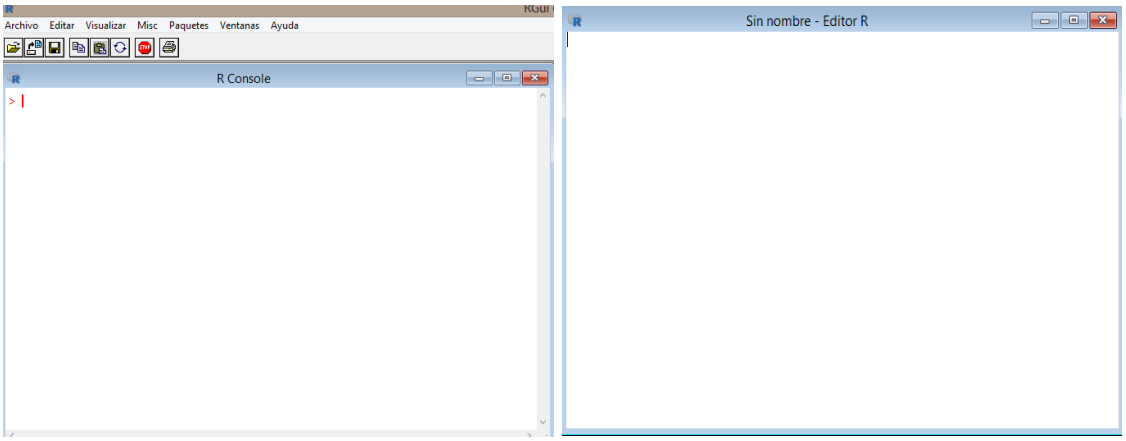


Figura 16. Consola y editor de scripts R.

Librerías y Apis.

- ✓ Bayesian: Estadística bayesiana
- ✓ ChemPhys: Quimiometria y Física Computacional
- ✓ ClinicalTrials: Diseño, monitorización y análisis de ensayos clínicos.
- ✓ Envirometrics: Modelos para la Ecologia.

- ✓ Finance: Análisis financiero.
- ✓ Genetics: Modelos para genética.
- ✓ MedicalImaging: Análisis de imágenes médicas.

Interacción con otros lenguajes.

- ✓ Python

Comunidad.

Se puede participar en el proyecto de R de las siguientes maneras.

- ✓ **Reportar Fallos.**

  Se debe tener bien claro de lo que se encontró como error lo sea teniendo, realizar un informe detallado de este y proceder a enviarlo al correo R-core@r-project.org donde debe ir como asunto del mensaje maintairner y entre paréntesis sobre que es el error por ejemplo maintainer(“graphics”).

  Si se tiene un parche para solucionar el problema este puede ser agregado al proyecto de R utilizando un control de versiones SVN, una vez instalado se debe ingresar el comando svn checkout https://svn.r-project.org/R/trunk/ R-devel donde se crea un directorio R-devel y este contendrá el código fuente de la versión más actualizada de R.

  Luego ya puede hacer los cambios que considera solucionan el problema de error tratando de tener el mismo estilo de codificación, luego de terminado se debe ejecutar el comando svn update y svn diff > patch.diff para que se actualicen los cambios y se cree un archivo patch.difff que es donde estarán los cambios realizados.

- ✓ **Conferencias.**

  Cuenta con dos tipos de conferencias una de estas es para usurios de R, para poder participar tiene que enviar una previa solicitud para ser tenido en cuenta al correo R-conferences@r-project.org.

  El otro tipo de conferencias está dirigida a desarrolladores de software estadístico.

- ✓ **Desarrollar Código.**

  Para participar como desarrollador de R se tiene que configurar la herramienta de subversiones SVN para acceder a los repositorios y allí obtener el código fuente y así poder realizar algunas modificaciones las cuales estarán sujetas a revisiones.

Soporte.

Se puede decir que cuenta con suficiente información relacionada a la manera de desarrollar con esta herramienta, variedad de tutoriales, videos, etc.

Áreas de aplicación.

- ✓ Medicina
- ✓ Matemática
- ✓ Física
- ✓ Genética
- ✓ Ecología
- ✓ Estadística

5.2.5 Julia

Origen y desarrollo histórico.

Su primera versión publicada fue en 2012 fue ideado por Stefan Karpinski, su nombre se debe al matemático francés Gaston Julia. Se desarrolló con la idea de ser una alternativa gratuita a las herramientas Matlab y Mathematica y poder resolver cualquier tipo de tarea.

Desde su lanzamiento ha tenido más de 1.3 millones de visitantes, más de 9000 visitas en su plataforma de desarrollo, es una herramienta que no ha sido muy promocionada pero a pesar de eso cuenta con más de 10000 usuarios, cuenta con unos 600 paquetes adicionales, ya que es muy poco el tiempo que lleva de publicado ha tenido buena aceptación y se espera que siga avanzando.

Características del lenguaje.

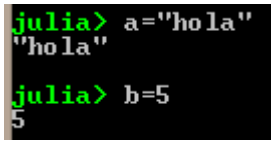
Sintaxis.

Es muy parecida a la utilizada en Octave o Matlab.

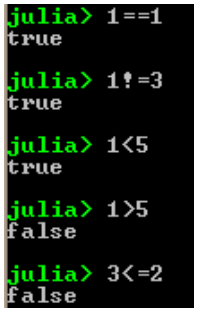
- Comentarios:

-

- Variables:



- Operadores lógicos:



- Imprimir:

```
julia> println("Esto es una impresion")
Esto es una impresion
```

- Vectores:

```
julia> v=[1, 2, 3]
3-element Array{Int64,1}:
 1
 2
 3
```

- Sentencia if:

```
julia> a=3
3

julia> if a > 10
       println("a es mayor que 10")
       else
       println("a es menor que 10")
       end
a es menor que 10
```

- Ciclo for:

```
julia> for i = 1:5
       println(i)
       end
1
2
3
4
5
```

- While:

```
julia> while x < 3
       println(x)
       x=x+1
       end
0
1
2
```

Paradigmas de programación.

Multiparadigma de tipo dinámico de alto nivel y alto desempeño.

Traductor.

Lo que caracteriza a Julia es que no utiliza un traductor intermedio, compila directamente al código de la máquina.

Entorno de desarrollo.

Utiliza una consola de comandos para ejecutar sus funciones, se puede enlazar con otras herramientas para tener un mejor entorno de desarrollo.

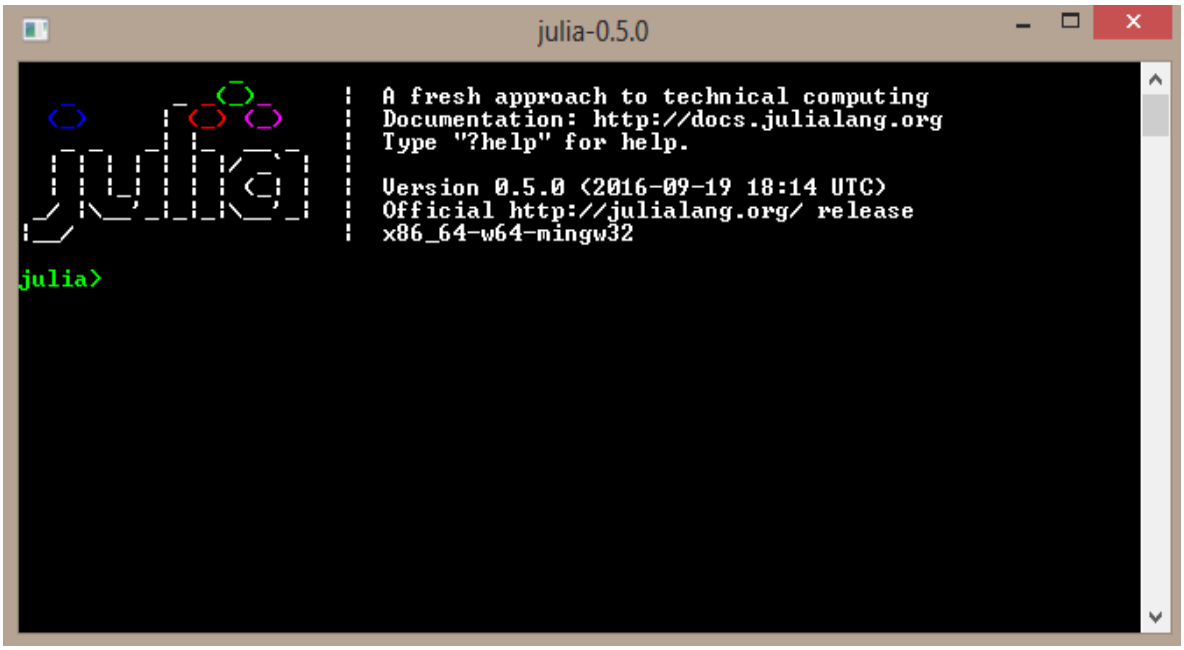


Figura 17. Ventana principal de Julia.

Librerías y Apis.

- ✓ DSP: procesamiento digital de señales.
- ✓ NLsolve: desarrollo sistemas de ecuaciones no lineales.
- ✓ SymPy: desarrollo funciones simbólicas.
- ✓ SerialPorts: enviar o recibir datos utilizando puerto serial.
- ✓ PyCall: llamar funciones de Python

Interacción con otros lenguajes.

- ✓ Python
- ✓ Php
- ✓ Nodo
- ✓ Js
- ✓ Perl

Comunidad.

- ✓ **Lista de correos.**

  Cuenta con varios grupos de correos para poder enterarse de las últimas noticias sobre nuevas versiones, hacer preguntas, discusiones sobre el desarrollo de julia, discusión sobre programación matemática, funcionamiento de julia en la nube y discusiones del uso de julia en idioma español.

- ✓ **Desarrollo.**

  Manejo de un repositorio para tener acceso al código fuente y poder utilizarlo para modificación o encontrar errores, que serán notificados mediante la plataforma de subversión que se debe manejar.

Soporte.

Debido a que es un lenguaje relativamente nuevo no está muy documentado, la información es escasa.

Áreas de aplicación.

- ✓ Estadística.
- ✓ Biología.
- ✓ Astronomía

- ✓ Finanzas
- ✓ Matemáticas.

En resumen a la caracterización de cada una de las herramientas se tiene que dos de estas herramientas tratadas son de licencia comercial las cuales se incluyeron para para tener una referencia de sin son mejores con respecto a las de licenciamiento libre lo cual su punto más visible es el soporte que brindan estas a sus usuarios, en cuando a los lenguajes de programación todas poseen uno propio, los paradigmas que se manejan son multiparadigma a diferencia de Scilab y Octave que manejan funcional y programación estructurada respectivamente, para interfaces graficas de usuario la única herramienta que no cuenta con esta opción hasta el momento es octave, la participación que se tiene en la comunidad de las herramientas libres funcionan de manera similar en todas permitiendo trabajar como desarrollador y reporte de errores, sus últimas versiones creadas son recientes lo cual indica que están en constante mantenimiento y crecimiento, todas pueden ser utilizadas en las plataformas de Windows, Linux y Mac, por último se tiene que las áreas de aplicación son muy amplias en todas las herramientas, donde se tienen algunas en común pero también cada una se inclina más a cierta área.

6. Vectorización y programación paralela.

6.1 Matlab & octave

Debido a que Matlab y Octave están pensados para trabajar con matrices toda variable o todo dato introducido será almacenado como una matriz no importa si es un solo valor ya que quedaría una matriz de tamaño 1x1.

```
>> A=50;
>> whos A
  Name      Size            Bytes  Class     Attributes

  A         1x1                 8  double
```

No se tiene en cuenta el tipo de dato que sea introducido, su almacenamiento sigue en forma de matriz, en este caso una cadena de caracteres, tamaño de 1x10, el 10 representa cada carácter guardado individualmente.

```
>> B='hola mundo';
>> whos B
  Name      Size            Bytes  Class    Attributes

  B         1x10               20  char
```

La manera de acceder a sus elementos se consigue con indicar el nombre de la matriz y la posición o índice de la matriz, en este caso por ser de una solo fila no se indica el primer valor que corresponde a la fila basta con indicar solo la columna.

```
>> B(7)

ans =

u
```

A partir de esta manera de trabajar en Matlab y Octave, existe un tipo de optimización de código llamado vectorización.

Vectorización.

Cuando se trabaja con vectores es necesario hacer un recorrido a los mismos mediante ciclos los cuales por lo general son muy lentos, es por esto que cuando se tengan aplicaciones que requieran de un alto rendimiento utilicemos la vectorización, lo cual se refiera a tratar de quitar los ciclos y reemplazarlos por funciones y operaciones semejantes.

Al utilizar la vectorización el código puede quedar más parecido a expresiones matemáticas tal y como aparecen en los libros, haciendo que sea más fácil de entender. Por otra parte la vectorización reduce las líneas de código, y entre menos líneas menos posibilidades de crear errores de programación y mayor velocidad en la ejecución del código.

Mostraremos un ejemplo donde se imprimen los números del 1 al 100.000 utilizando un ciclo for, y vectorizando donde veremos el tiempo que demora la ejecución de cada uno.

Ciclo FOR.

```
tic
for t=0:1:100000
    disp(t)
end
toc
```

```
       99998

       99999

      100000

Elapsed time is 2.914498 seconds.
>>
```

Utilizando la vectorización.

```
tic
t=0:1:100000;
disp(t)
toc
```

```
      99995       99996       99997

  Column 100001

      100000

Elapsed time is 0.737990 seconds.
>>
```

Si nos fijamos en el número de líneas de código no es muy significativa la diferencia, pero al observar el tiempo trascurrido en cada ejecución si se puede ver la rapidez que se tiene al momento de vectorizar con respecto a utilizar los ciclos repetitivos.

Existen algunas funciones para utilizar en la vectorización.

all	Para saber si todos los elementos de una matriz son diferentes de cero.
any	Para saber si algún elemento es diferente de cero.
cumsum	Suma acumulativa.
diff	Diferencia entre el número siguiente y el actual.
find	Nos muestra los índices de los valores diferentes de cero.
sort	Organizar los elementos de una matriz de menor a mayor.
sum	Sumar los elementos de una matriz

Tabla 4. Funciones para trabajar vectorización.

Programación Paralela Matlab.

Se trata de realizar varios cálculos de manera simultánea, donde se note el ahorro de tiempo en la distribución de las tareas y ejecución de las mismas. Nos permite solucionar problemas que manejen una gran cantidad de datos haciendo uso de los recursos informáticos (núcleos o procesadores).

Matlab tiene por defecto utilizar solo un procesador pero cuenta con la opción de poner a f

```
>> matlabpool(2)
Starting matlabpool using the 'local' profile ... connected to 2 workers.
```

ncionar los demás procesadores que tengamos en nuestro equipo.

Para este caso se utilizó matlabpool (2), el 2 se refiera a la cantidad de procesadores, puede ser 4 en el caso de contar con un procesador quadcore.

Para finalizar el uso de varios procesadores se utiliza el siguiente comando.

```
>> delete(gcp)
Parallel pool using the 'local' profile is shutting down.
```

Esto hace que se ejecute más rápido nuestro código pero dependiendo de cómo está escrito, ya que no todo código puede ser ejecutado en paralelo.

En el siguiente ejemplo se imprime 10 millones de veces la palabra hola, donde se utiliza un ciclo for.

FOR

```
tic
for i=1:10000000
    disp('hola');
end
toc;
```

```
hola
hola
Elapsed time is 163.223908 seconds.
```

En el siguiente ejemplo se imprime 10 millones de veces la palabra hola, donde se utiliza un ciclo parfor.

PARFOR

```
tic
parfor i=1:10000000
    disp('hola');
end
toc;
```

```
hola
hola
Elapsed time is 104.108049 seconds.
```

En estas dos ejecuciones se puede ver como existen dos tipos de for uno para calculo simple con un solo procesador y el parfor para trabajar programación paralela, se ve claramente la diferencia entre tiempos de ejecución, casi con un 40% de mayor rapidez.

6.2 Python.

Python cuenta con varios paquetes entre esos Numpy, el principal para desarrollo científico y manejo de matrices n-dimensionales. En cuanto al manejo de matrices esta una clase llamada numpy.vectorize, utilizada para vectorizar funciones que solo aceptan como datos de entrada escalares, y como su nombre indica se trata de poder cambiar esas entradas escalares por vectores o numpyarray y retornando el mismo tipo.

Vectorización.

Una de las funciones básicas de Python es valor absoluto de un número.

```
>>> print(abs(-3))
3
```

Si aplicamos esta función a un vector nos va a genera error.

```
>>> print(abs([-3, -5, -7]))
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    print(abs([-3, -5, -7]))
TypeError: bad operand type for abs(): 'list'
```

Es por esto que se puede utilizar la vectorización para poder pasar un vector como dato de entrada a la misma función de valor absoluto.

```
>>> import numpy as np
>>> vecabs=np.vectorize(abs)
>>> print(vecabs([-3, -5, -7]))
[3 5 7]
```

Como se puede observar se importa el paquete numpy y se le agrega un alias np, se vectoriza la función abs ahora tomando el nombre de vecabs al cual ya se le puede dar como entrada un vector.

La vectorización también consiste en reemplazar los ciclos repetitivos con funciones que cumplan con lo mismo, pero mejorando su rapidez de ejecución.

```
from time import time
import numpy as np
aleatorios=np.random.randn(10000000)
ti=time()
np.array([abs(i) for i in aleatorios])
tf=time()
te=tf-ti;
print('tiempo de ejecucion',te)
```

```
tiempo de ejecucion 3.202881097793579
```

```
from time import time
import numpy as np
aleatorios=np.random.randn(10000000)
ti=time()
vecabs=np.vectorize(abs)
vecabs(aleatorios)
tf=time()
te=tf-ti;
print('tiempo de ejecucion',te)
```

```
tiempo de ejecucion 1.5569989681243896
```

Se puede notar que el tiempo de ejecución es mucho menor utilizando la vectorización.

Programación paralela.

Se cuenta con tres maneras de realizar la programación paralela, manejo de Threads/hilos o tareas, mensajes, y la combinación de las dos.

- ✓ Threads

  Es usado con arquitecturas de memoria compartida, comunicación entre los threads del procesador, utiliza OpenMP, CUDA, OpenCL.

- ✓ Mensajes.

  Es usado con arquitecturas de memoria distribuida, comunicación entre los procesadores del sistema, los datos de las tareas son compartidas en el mensaje, utiliza MPI.

- ✓ Combinación de las dos anteriores.

  Utiliza ambas arquitecturas, usa OpenMp, CUDA, OpenCL+MPI

Python maneja una librería llamada GIL (Global Interpreter Lock) la cual impide que sean ejecutadas varias tareas de manera simultánea. Para solucionar este problema se cuenta con un módulo Multiprocessing, que permite ejecutar distintos procesos en distintos procesadores, solo necesita ser importado a Python.

7. Análisis de software

7.1 Descripción de la actividad

Lo que se pretende hacer en este trabajo es explorar varias herramientas de programación las cuales tengan algún enfoque en el tratamiento de imágenes, donde se caracterizaran cada una para determinar que alcances, ventajas, desventajas, etc. poseen estas en cuanto al tratamiento de imágenes, para así poder seleccionar dos herramientas y utilizarlas en la implementación de un aplicación simple donde se trabaje algunas de las funciones de tratamiento de imágenes y así poder observar que diferencias existen entre las mismas para formar un cuadro donde se muestre lo evidenciado.

8. Prueba de aplicación.

8.1 Softwares seleccionados

Después de haber realizado un análisis detallado de cada uno de los softwares donde se evidenciaron sus ventajas, desventajas, aplicación y alcance en cuanto al tratamiento de imágenes se decidió seleccionar los dos mejores que se acomodan a nuestras necesidades de tratar imágenes.

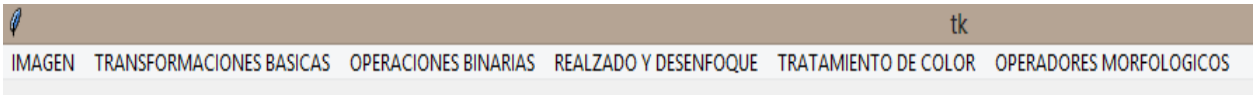
Es por esto que se ha seleccionado la herramienta Python y Scilab, debido a que Matlab y Mathematica son comerciales, octave no cuenta con la manera de crear interfaces gráficas, r está orientado más a la parte estadística y julia por ser la más nueva dentro de estas herramientas no cuenta con mucha documentación que nos sirva como base para trabajar este tema.

8.2 Objetivo de la Prueba.

Se desarrolló una aplicación implementando una interfaz amigable para el usuario donde este pueda realizar algunas de las funciones a las cuales pueden ser sometidas las imágenes, para esta aplicación se utilizó las herramienta de Python y Scilab, en la primera se manejó el tratamiento de imágenes con una biblioteca llamada OpenCV, se utilizaron algunos módulos externos como pyplot para mostrar las imágenes, numpy que es quien trabaja con matrices, su interfaz gráfica fue creada con la biblioteca por defecto que trae Python Tkinter, todo está implementación se hizo dentro de un script.

Para el caso de Scilab el tratamiento de imágenes se hizo utilizando SIVP (procesamiento de video e imagen Scilab.), IPD (diseño de procesamiento de imágenes.), para su interfaz se manejó GUI Builder la cual permite crear interfaces rápidamente y genera el código automáticamente. , todo esto también dentro de un script.

Python



Se tiene un menú con los siguientes ítems.

Figura 18. Menú realizado en Python.

Imagen.

Se tiene la opción de seleccionar una imagen y cargarla para tratarla.

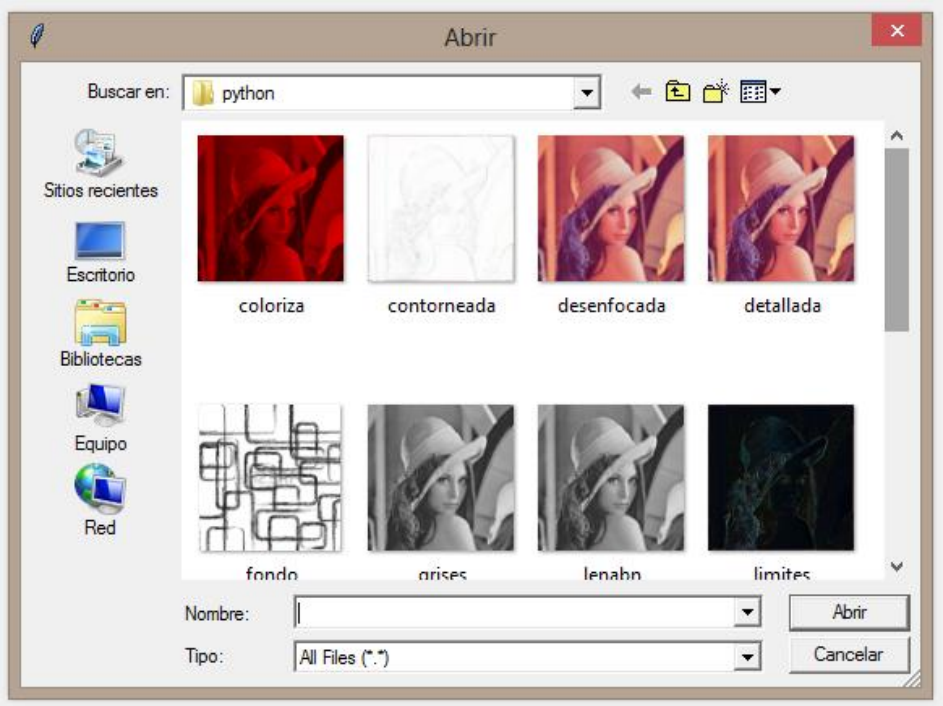


Figura 19. Cuadro de dialogo para seleccionar la imagen y cargarla.

Transformaciones básicas.

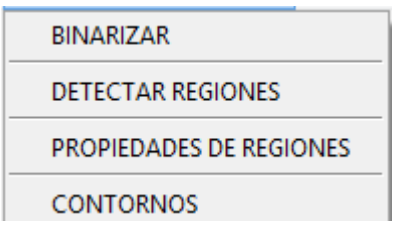
Ecualización: la ecualización del histograma de una imagen nos permite mejorarle el contraste y brillo dependiente si es demasiado oscura o brillante.

Desplazamiento: se utiliza para aclarar u oscurecer una imagen.

Figura 20. Imagen Ecualizada.

Operaciones binarias.

Son aquellas donde las imágenes donde su información cambia manejando solo dos valores posibles que son ceros y unos que corresponden a los colores blanco y negro, esto se hace con el fin de poder separar regiones u objetos para ser analizados.



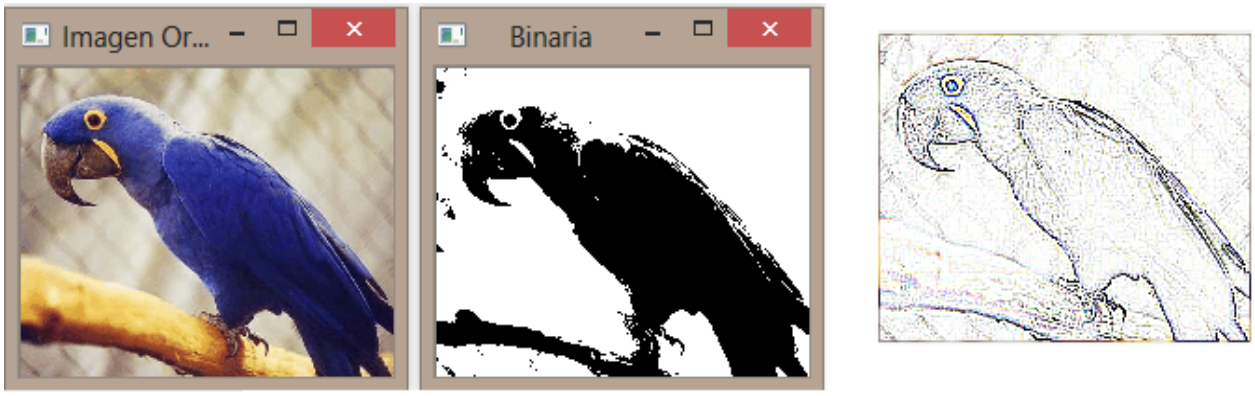
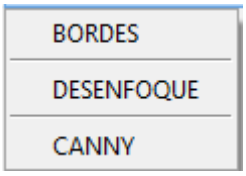


Figura 21. Imagen binarizada y con contornos.

Realzado y desenfoque.



Canny: es un filtro para la detección de bordes el cual se basa en la derivada gaussiana.

Desenfoque: consiste en disminuir la nitidez de la imagen.

Bordes: detectar bordes utilizando métodos diferentes al de canny como son prewitt, sobel.

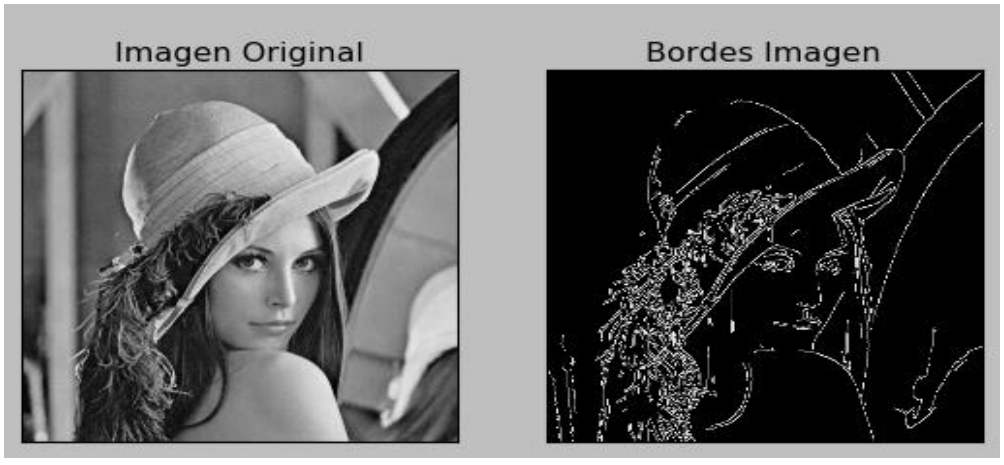


Figura 22. Bordes de una imagen utilizando el método canny.

Tratamiento de color.

Transformar imágenes que están en el sistema RGB y pasarlos a HSV donde nos muestra la matiz, saturación y brillo que tiene la imagen, en detección de color consiste en buscar colores que estén dentro de un rango establecido y mostrar la parte que corresponde a ese color, y falso color para poder hacer más visibles ciertas áreas.

Figura 23.detectar cada uno de los colores de la imagen.

Operadores morfológicos.

Son aquellos que simplifican imágenes y conservan las características principales de la forma del objeto. La dilatación consiste en agregar pixeles al objeto para hacerlo más grande, erosión extrae pixeles del objeto para hacerlo más pequeño, en la apertura se aplica una erosión seguida de una dilatación con el fin de abrir pequeños huecos, y cierre viceversa a la apertura.

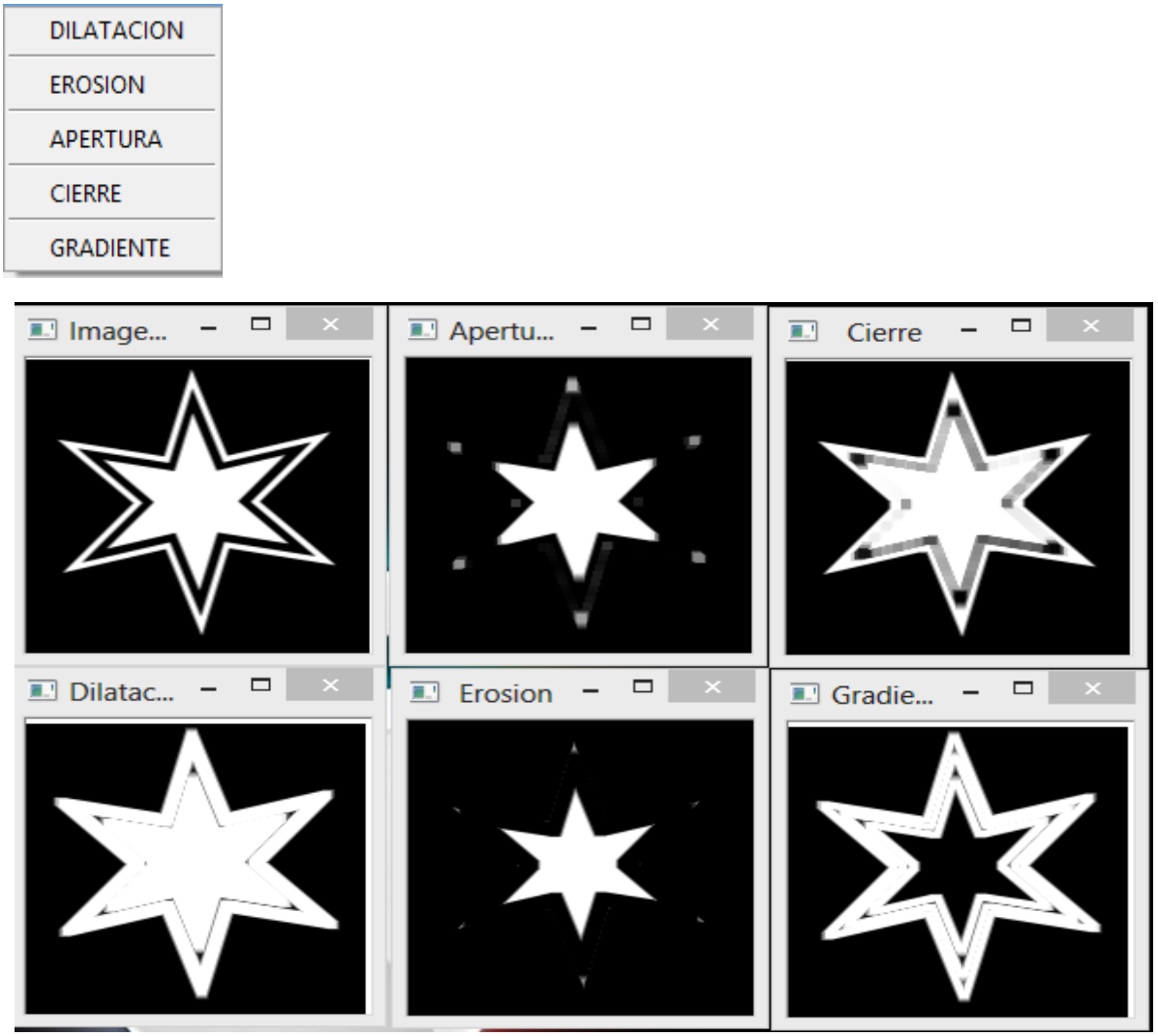


Figura 24. Aplicación de los operadores morfológicos (apertura-cierre-dilatación-erosión-gradiente)

Scilab.

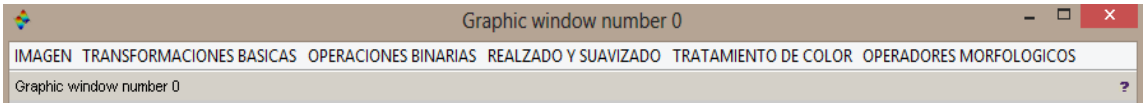


Figura 25. Menú hecho en Scilab.

Imagen.

Se tiene la opción de seleccionar una imagen y cargarla para tratarla.

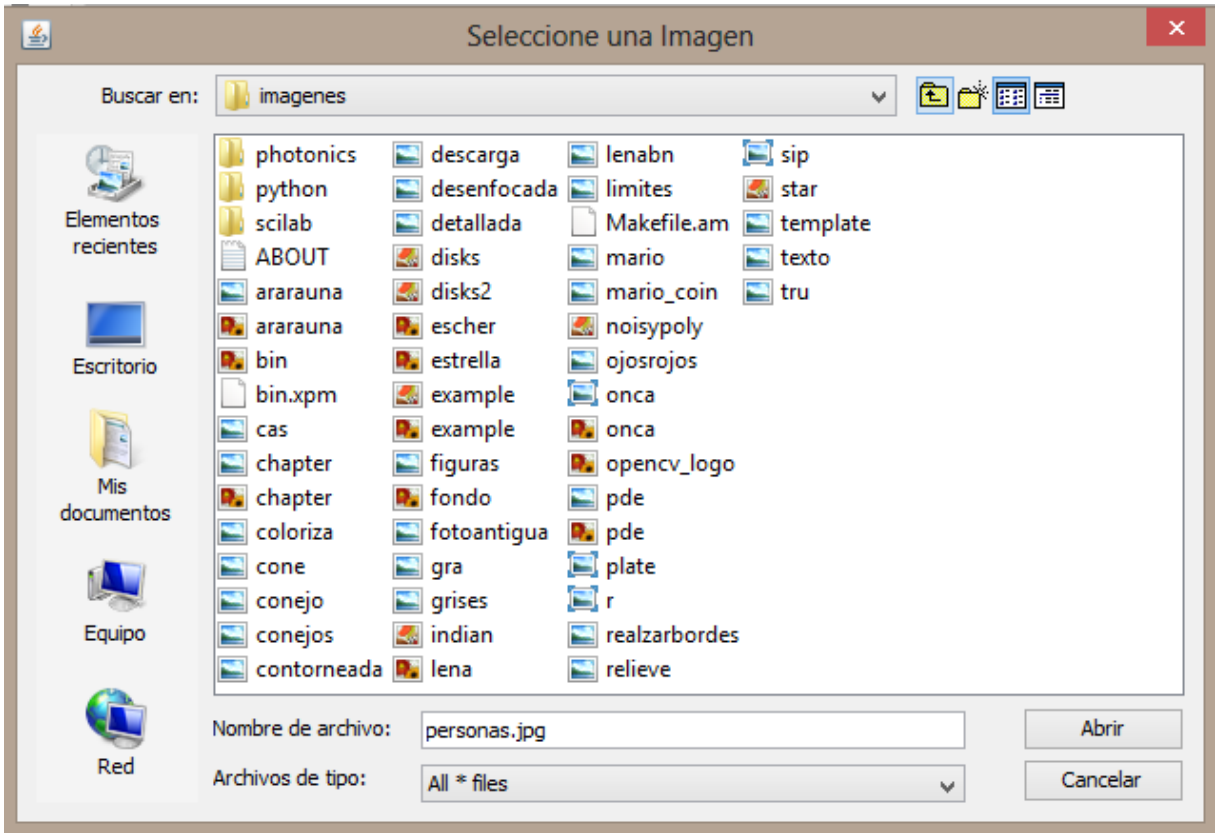
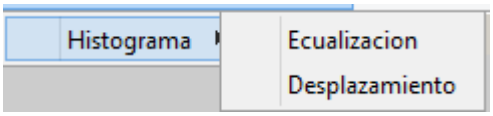


Figura 26. Cuadro de dialogo para seleccionar una imagen y cargarla

Transformaciones básicas.



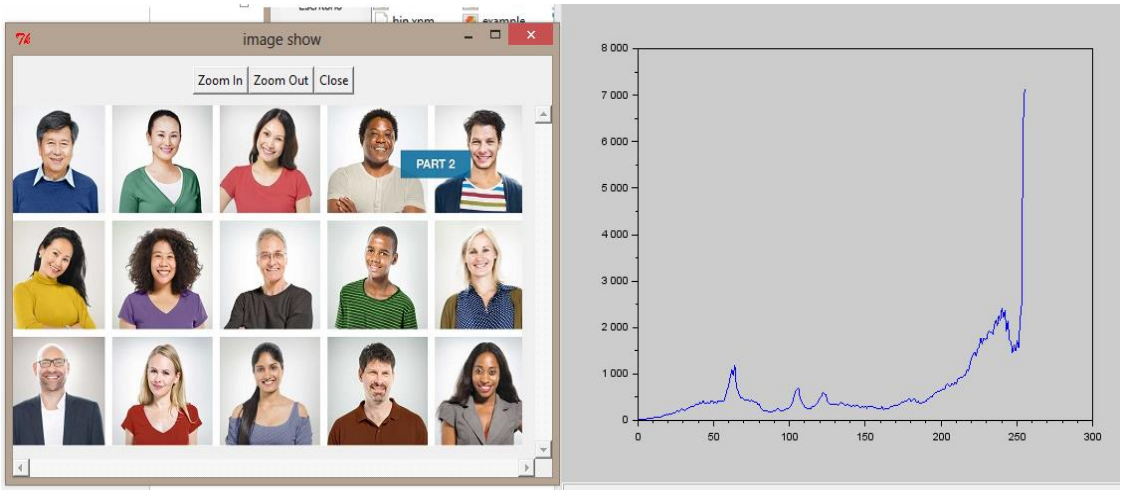
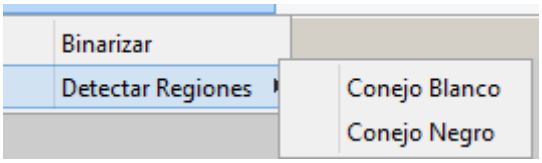


Figura 27. Histograma de una imagen.

Operaciones binarias.



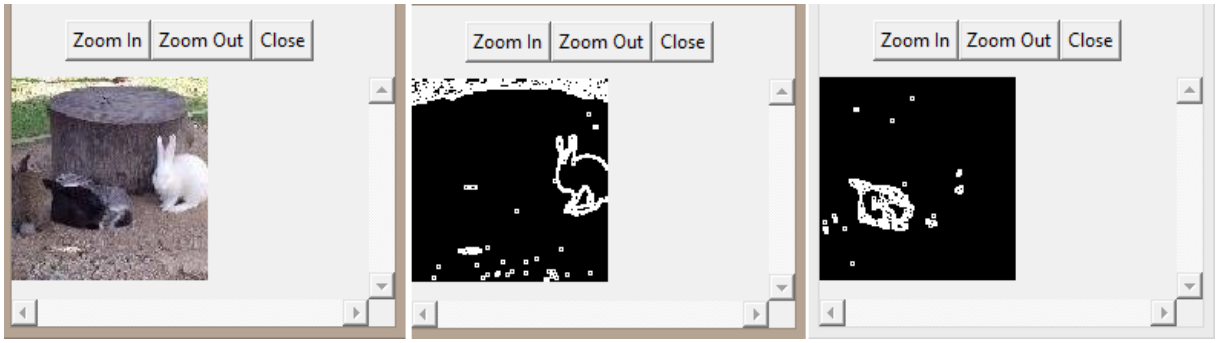


Figura 28. Detectar donde está un conejo de color blanco y el otro de color negro.

Realzado y suavizado.

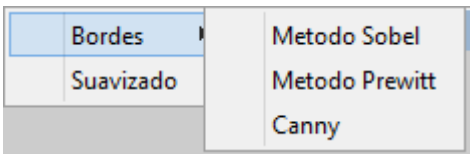


Figura 29. Imagen de un tigre para tratarla con los métodos de sobel, prewitt, canny para detectar bordes.(2016,Dreamstime, https://es.dreamstime.com/stock-de-ilustracin-funcionamiento-del-tigre-animal-salvaje-en-el-fondo-blanco-image65930531)

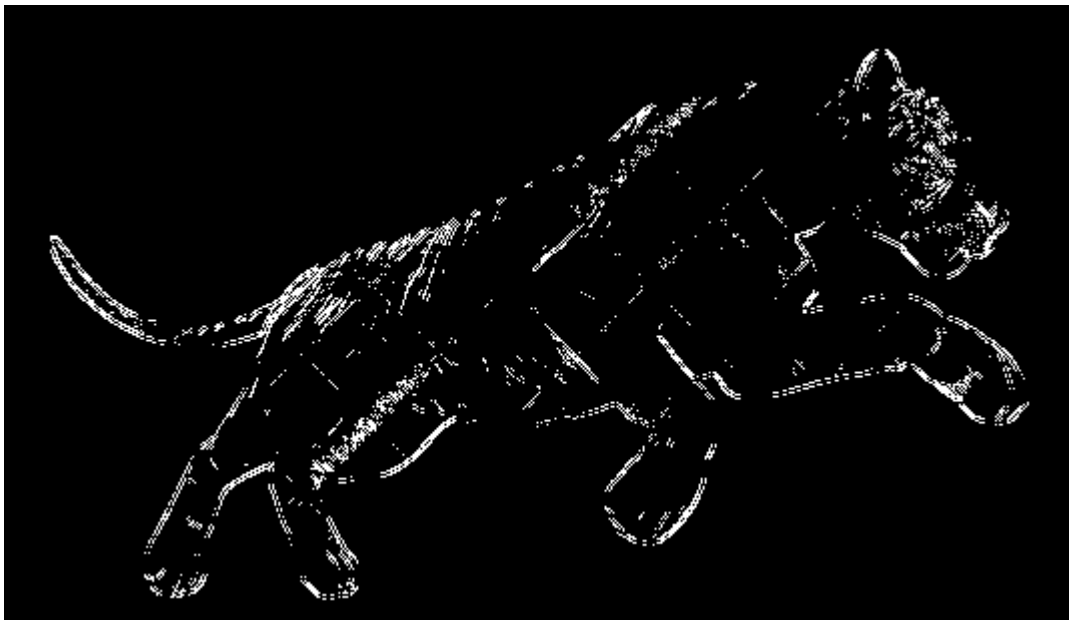


Figura 30. Imagen aplicándole el método sobel.

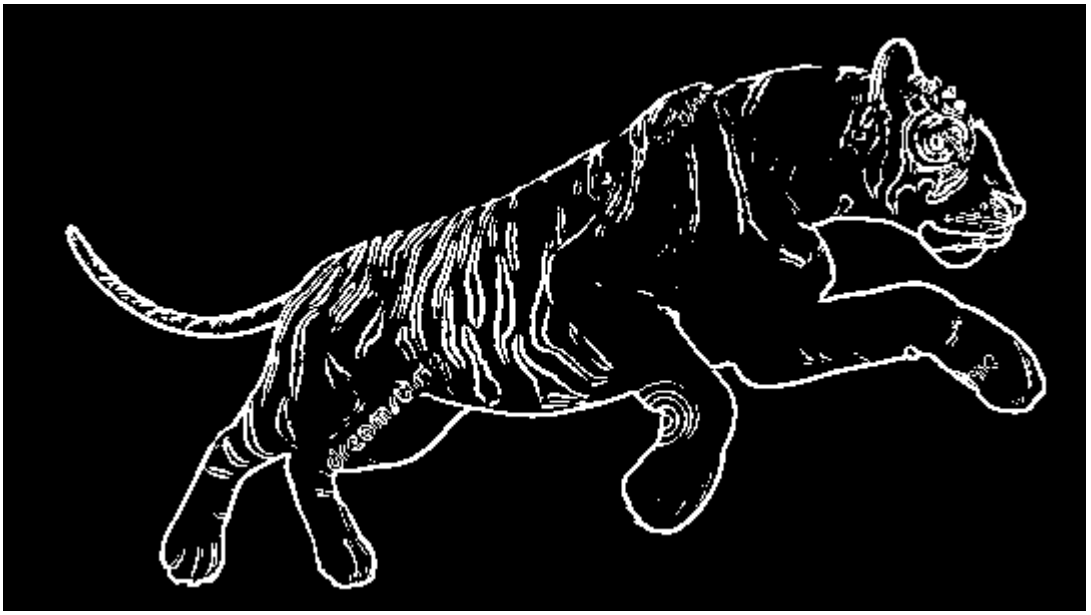


Figura 31. Imagen aplicándole el método prewitt.

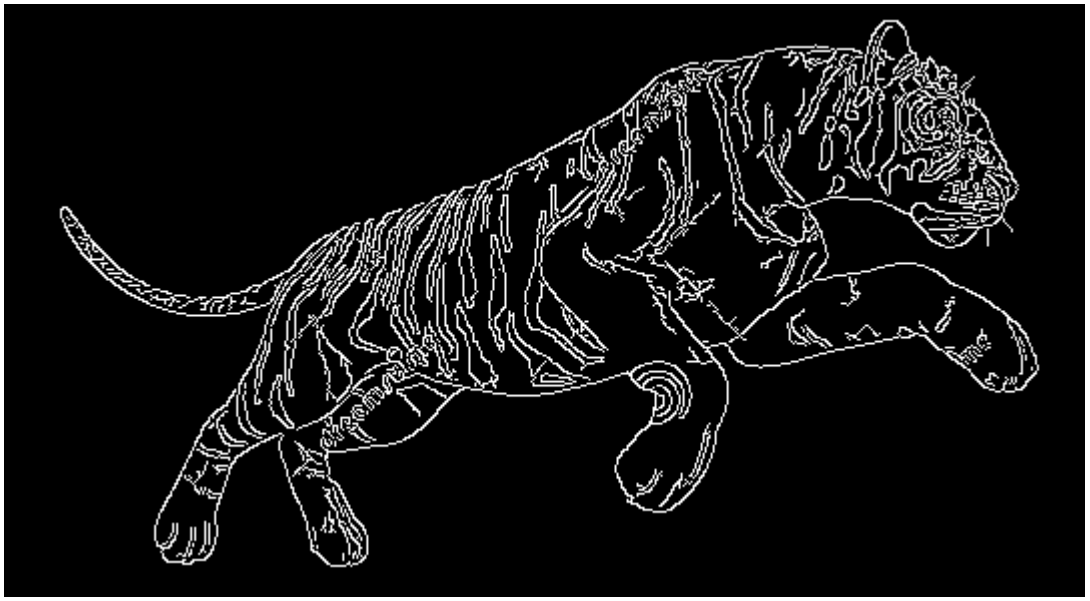


Figura 32. Imagen aplicándole el método canny.

Tratamiento de color.

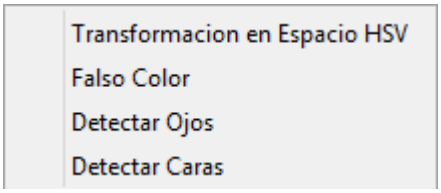


Figura 33. Imagen para aplicarle la detección de caras. (Garther, 2016, http://blogs.gartner.com/smarterwithgartner/files/2015/07/SWG_part2_personas_1.jpg)

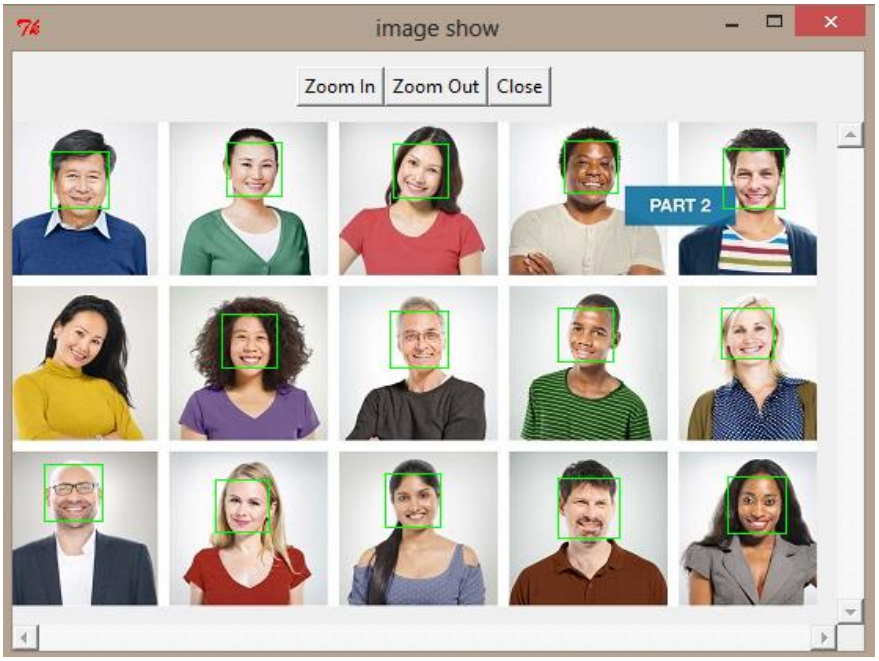


Figura 34. Aplicación de la detección de las caras.

Operadores morfológicos.

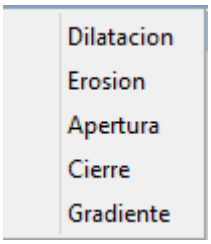


Figura 35. Método de dilatación aplicado.

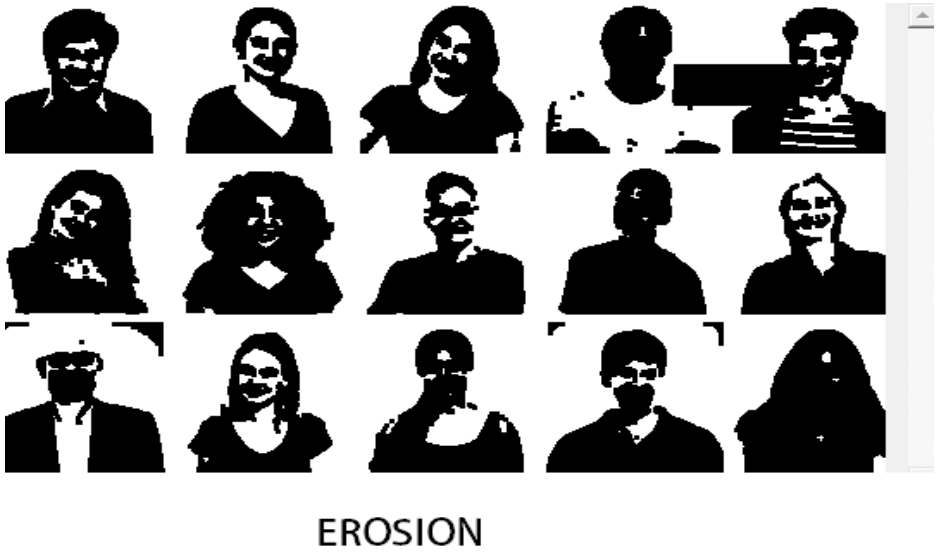


Figura 36. Método de erosión aplicado.

APERTURA

Figura 37. Método de apertura aplicado.

CIERRE

Figura 38. Método de cierre aplicado.

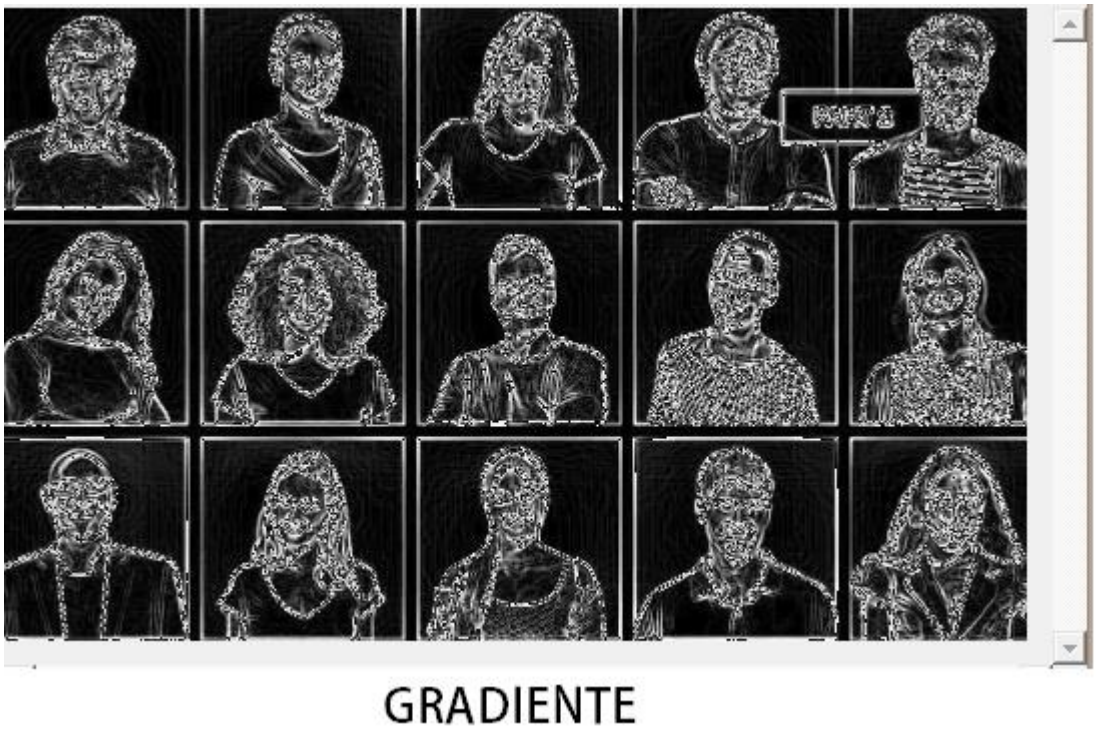


Figura 39. Método de gradiente aplicado.

Se puede observar los resultados que se obtuvieron al realizar la aplicación en estas dos herramientas, al momento de implementarla se tuvo más facilidad en Python, la librería que se utilizo fue OpenCV, en el caso de Scilab se manejó con SIVP (procesamiento de imagen y video Scilab). La instalación de ambas es muy sencilla, la ejecución de los scripts también son de fácil manejo, para la creación de interfaces graficas Scilab cuenta con una caja de herramientas que permite crear los objetos simplemente con arrastrarlos a la ventana principal, en Python se puede utilizar otra herramienta adicional que nos permita hacer eso mismo sin embargo gracias a la facilidad de crear interfaces utilizando solo código se hizo de esta manera.

9. Análisis de resultados.

En la siguiente tabla se realiza un comparativo que resume las principales características de las herramientas analizadas. En dicho cuadro se pueden apreciar las ventajas que puede tener una herramienta sobre otra, lo cual puede ser útil para orientar la decisión respecto al tipo de herramienta que se puede elegir, dependiendo de las características del problema que se desea abordar.

	Matlab	**Mathematica**	**Python**	**Scilab**
Sistemas operativos	Mac Linux Windows	Mac Linux Windows	Mac Linux, Windows	Mac Linux Windows
Lenguaje programación.	Propio.	Propio.	Propio.	Propio.
Paradigmas de programación.	Multiparadigma.	Multiparadigma.	Multiparadigma.	Funcional.
Interfaces.	Si	Si	Si	Si
Programación Paralela	Si	Si	Si	No
Vectorización	Si	No	Si	No
Software y	Comercial	Comercial	Libre	Libre

Licencia.	Privada.	Privada.	Publica	publica
Comunidad de desarrollo.	Alta	Alta	Alta	Alta
Documentación.	Ingles.	Ingles.	Ingles.	Ingles.
Áreas de aplicación.	Sistemas de control, mecatrónica, análisis financiero, procesamiento de señales, procesamiento de imagen y video, comunicaciones inalámbricas, biología computacional, robótica, etc.	Química, sistemas de control, eléctrica, procesamiento de imágenes, industrial, mecánica, óptica, bioinformática, economía, estadística, etc.	Computación científica de alto rendimiento, automatización industrial, biología, física, medicina, astronomía, química, matemáticas, procesamiento de imágenes, etc.	Cálculos de ingeniería, análisis de señales, análisis estadístico, análisis geométricos, procesamiento de imágenes, etc.
Interacción con otras herramientas.	Visual Basic .NET, java, c/c++, Python.			Python, Perl, c, c++.
Última versión.	Matlab R2016a, marzo 2016	Mathematica 11.0.1,	Python 3.5.2, junio 2016	Scilab 5.5.2, marzo 2015

		septiembre 2016		

Tabla 4. Cuadro comparativo.

	Octave	R	Julia
Sistemas operativos	Mac Linux Windows	Mac Linux Windows	Mac Linux, Windows
Lenguaje de programación.	Propio.	Propio.	Propio.
Paradigmas de programación.	Estructurada.	Multiparadigma.	Multiparadigma.
Interfaces.	No	Si	Si
P. Paralela	Si	Si	Si
Vectorización	Si	Si	No
Software y Licencia.	Libre Publica	Libre Publica	Libre Publica

Comunidad de desarrollo.	Media	Alta	Media
Documentación.	Ingles.	Ingles.	Multilenguaje.
Áreas de aplicación.	Procesamiento de imágenes, algebra lineal, electrónica, control automático, medicina, circuitos, señales, comunicaciones, estadística, economía, etc.	Medicina, matemática, física, genética, ecología, estadística, etc.	Estadística, biología, astronomía, finanzas, matemáticas, etc.
Interacción con otros lenguajes.		Python	Python, php, nodo, js, Perl
Última versión.	Octave 4.2.0 noviembre 2016	R 3.3.1, junio 2016	Julia 0.5.0

Tabla 5. Cuadro comparativo.

Teniendo en cuenta el cuadro comparativo se puede observar que las herramientas Matlab y Mathematica son muy completas para el desarrollo de cualquier aplicación incluyendo las de tipo

científico, sin embargo hay una característica que hace que no se tengan en cuenta para ser seleccionadas y esto corresponde a su tipo de licenciamiento ya que el costo de su licencia es muy elevado.

En cuanto a su última versión que se tiene de cada herramienta se puede observar que son muy recientes lo que significa que están en constante desarrollo para así tenerlas muy completas y con la menor cantidad de errores, para brindar una buena solución a los problemas de programación.

Basados en el análisis que se hizo de cada una de las herramientas se puede decir que Python fue una de las seleccionadas para el desarrollo del aplicativo sobre tratamiento de imágenes gracias a que el tiempo de aprendizaje es muy corto, su sintaxis entendible, cuenta con una gran cantidad de paquetes lo que hace que el tiempo de desarrollo sea corto, y algo muy importante es la demanda alta que tiene en campo laboral.

Python debería ser uno de los primeros lenguajes que se manejen en la carrera de sistemas, ya que es muy fácil de entender y aprender para poder empezar a tener la lógica de programador y así ser más fácil empezar a utilizar otros lenguajes más avanzados, pero esto no quiere decir que Python no sea una herramienta muy completa para desarrollo de software, para tener una noción de lo potente que es se sabe que es utilizado por la NASA.

Para el desarrollo de las aplicaciones sobre tratamiento de imágenes se tuvieron en cuenta Python por ser fácil de aprender y contar con paquetes que facilitan este proceso, Scilab por tener algunas semejanzas con Matlab, Octave no se tuvo en cuenta por no tener la capacidad de crear interfaces de usuario, R por estar orientado más a la parte estadística, Julia por su poco soporte que se tiene al ser algo nuevo su aparición y Matlab y Mathematica por ser comerciales.

10. Conclusiones

Se ha realizado una revisión bibliográfica para determinar las áreas en las cuales se desarrolla software científico, el cual tiene ciertas características especiales, las cuales se describen en el marco teórico. También se ha determinado cuales son las herramientas para su desarrollo más destacadas en la bibliografía, incluso se ha encontrado una metodología específicamente utilizada para aplicar en el desarrollo de este tipo de aplicaciones, lo cual indica que la comunidad científica y del área de Ingeniería del Software han encontrado que este tipo de aplicaciones tiene un tratamiento especial, y en su desarrollo es importante tener en cuenta dichas circunstancias.

Se ha realizado un estudio comparativo sobre algunas herramientas de software libre y comercial para el desarrollo de aplicaciones científicas. Se seleccionaron un total de 7 herramientas habitualmente citadas en la bibliografía: Matlab, Mathematica, Scilab, Python, Octave, R, Julia. De cada una de las herramientas se han estudiado sus características, de forma tal que se pueda encontrar sus diferencias, afinidades y especialidades. Dicha caracterización se ha resumido en una tabla que permite a los interesados tener en cuenta las particularidades de cada herramienta para decidir cuál puede adecuarse de mejor manera a sus desarrollos.

Finalmente se han seleccionado dos de las herramientas analizadas para implementar una sencilla aplicación en tratamiento de imágenes, con el objetivo de analizar de primera mano su uso. En particular se pudo observar que ambas cuentan con un gran número de funciones para el

tratamiento de imágenes, sin embargo en el caso de Python su implementación es mucho más flexible, así mismo para la implementación de las interfaces graficas Scilab tiene presente algunos inconvenientes al momento de querer agregar más objetos, ya que vuelve y genera todo el código de estos objetos haciendo que lo que se tenga implementado se pierda, algo que no es nada favorable para la programación.

Se puede destacar que una de las herramientas más versátiles de las analizadas es Python, es un lenguaje da fácil aprendizaje, una sintaxis que facilita la programación incluyendo una disminución en el tiempo de desarrollo, un número importante de librerías para uso específico en diferentes áreas y una comunidad amplia y creciente que facilita el autoaprendizaje y el soporte. El estudio de dicho lenguaje podría incluso incluirse en los primeros cursos de programación, facilitando una profundización en su manejo, dada la demanda actual de programadores de Python en diferentes frentes de la industria.

11. Bibliografía.

Arabnejad, H., & Barbosa, J. G. (2016). Maximizing the completion rate of concurrent scientific applications under time and budget constraints. *Journal of Computational Science*. https://doi.org/10.1016/j.jocs.2016.10.013

Barrett, R. F., Crozier, P. S., Doerfler, D. W., Heroux, M. A., Lin, P. T., Thornquist, H. K., … Vaughan, C. T. (2015). Assessing the role of mini-applications in predicting key performance characteristics of scientific and engineering applications. *Journal of Parallel and Distributed Computing*, *75*, 107–122. https://doi.org/10.1016/j.jpdc.2014.09.006

Castañè, G. G., Núneñ, A., Filgueira, R., & Carretero, J. (2012). Dimensioning Scientific Computing Systems to Improve Performance of Map-Reduce based Applications. *Procedia Computer Science*, *9*, 226–235. https://doi.org/10.1016/j.procs.2012.04.024

Heaton, D., & Carver, J. C. (2015). Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology*, *67*, 207–219. https://doi.org/10.1016/j.infsof.2015.07.011

Herniter, M. E. (2001). Programming in MATLAB. *Subband Adaptive Filtering*, 486. https://doi.org/10.1002/9781118033319.ch4

Herrero, P. A. V. (n.d.). Programación Paralela y Distribuida. Retrieved from http://2013.es.pycon.org/media/programacion-paralela.pdf

John W. Eaton. (1996). GNU Octave. Retrieved from https://www.gnu.org/software/octave/doc/interpreter/

Kanewala, U., & Bieman, J. M. (2014). Testing scientific software: A systematic literature

review. *Information and Software Technology*, *56*(10), 1219–1232. https://doi.org/10.1016/j.infsof.2014.05.006

Kelly, D. (2009). Determining factors that affect long-term evolution in scientific application software. *Journal of Systems and Software*, *82*(5), 851–861. https://doi.org/10.1016/j.jss.2008.11.846

Kelly, D. (2015). Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software. *Journal of Systems and Software*, *109*, 50–61. https://doi.org/10.1016/j.jss.2015.07.027

La Implementación de la adaptación modular de software Científico. (n.d.). Retrieved from http://www.sciencedirect.com/science/article/pii/S1877750312000099

Li, Y., Guzman, E., Tsiamoura, K., Schneider, F., & Bruegge, B. (2015). Automated Requirements Extraction for Scientific Software. *Procedia Computer Science*, *51*, 582–591. https://doi.org/10.1016/j.procs.2015.05.326

Nazarenko, A. M., & Prokhorov, A. A. (2015). Hierarchical Dataflow Model with Automated File Management for Engineering and Scientific Applications. *Procedia Computer Science*, *66*, 496–505. https://doi.org/10.1016/j.procs.2015.11.056

Read the Docs. (n.d.). Julia Documentation. Retrieved from http://docs.julialang.org/en/release-0.5/

Scilab Enterprises S.A.S. (2015). Documentation. Retrieved from http://www.scilab.org/resources/documentation

The MathWorks, I. (1994). Documentation. Retrieved from

https://es.mathworks.com/help/matlab/index.html

The Scipy community. (2008). numpy.vectorize. Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.vectorize.html