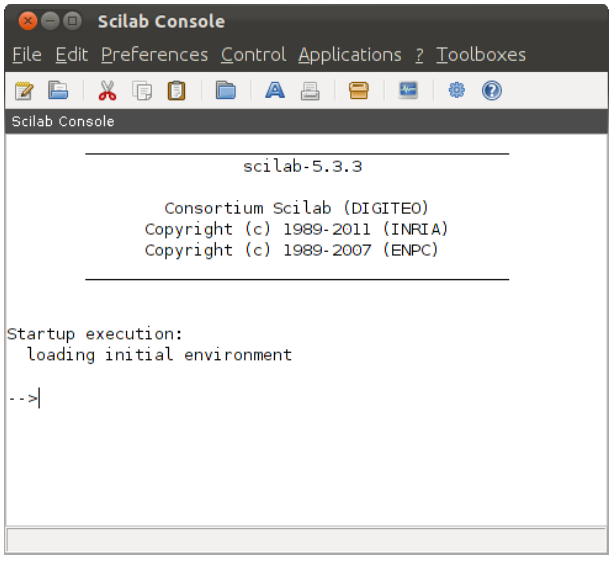
Universidad del Valle
Escuela de Ingeniería de Sistemas y Computación
Curso: Métodos Numéricos
Docente: Daniel Barragán Calderón

Clase Introducción a la Programación con Scilab (Parte 2)

Introducción

Scilab fue creado en 1990 por investigadores en INRIA (*Institut national de recherche en informatique et en automatique*) y ENPC (*École nationale des ponts et chaussées*). Scilab es un lenguaje de programación de código abierto, multiplataforma, orientado a cálculos numéricos. Puede ser usado para procesamiento de señales, análisis estadístico, tratamiento de imágenes, simulación de fluidos, optimización y modelamiento y simulación de sistemas dinámicos.

Enlace de descarga:
http://www.scilab.org/



Interfaz Gráfica Scilab

Recomendaciones

Las variables declaradas al interior de una función no se veran reflejadas en la consola. Por tanto si necesita obtener el valor de las variables al interior de la función, deberá retornar las variables de la siguiente manera:

```
function [x, aproximacion] = mifuncion()
// Digite aquí su codigo fuente que asigna un valor para x y para aproximacion
endfunction
```

Guía Básica

En esta guía se abordaran otros aspectos de la programación y visualización de funciones con Scilab.

Archivos

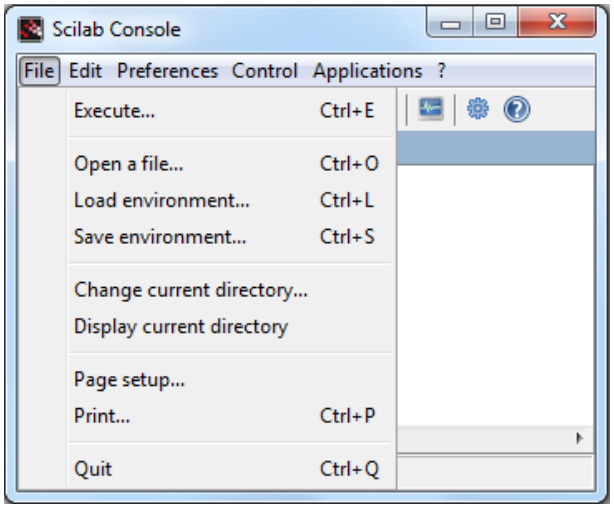
Scilab permite la creación de archivos para el almacenamiento de informacíón. Al manejar archivos en Scilab se deben tener en cuenta las siguientes consideraciones:

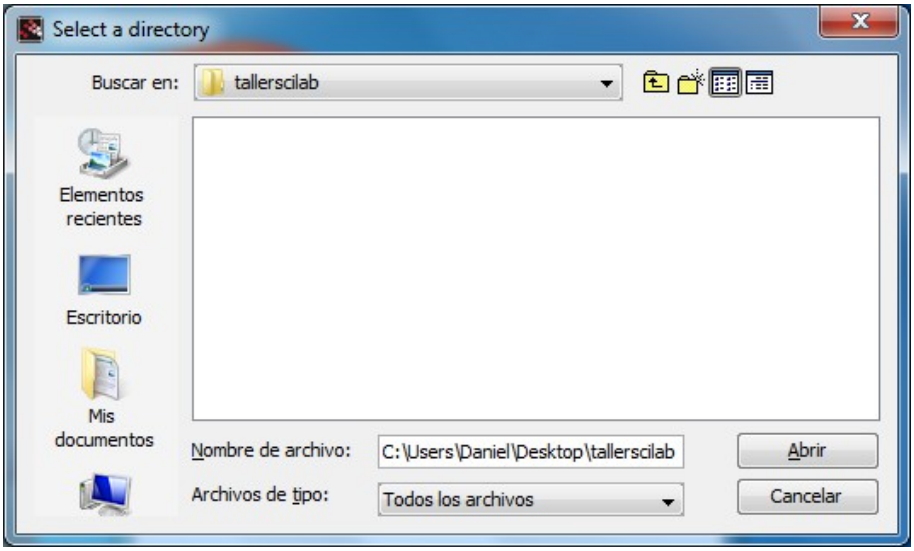
1. Al crear un archivo para escritura o abrirlo para lectura, después de realizar operaciones de lectura ó escritura, el archivo debe cerrarse.
2. Al crear un archivo para escritura o abrirlo para lectura, se debe especificar la ruta de ubicación del archivo. La ruta puede ser una ruta absoluta ó una ruta relativa.
3. Al emplear rutas relativas, Scilab almacena los archivos con relación al directorio de trabajo de la consola de Scilab. Para conocer el directorio de trabajo de la consola, digite en la consola de Scilab el comando ***pwd***.

A. Actualización del directorio de trabajo

Siga los siguientes pasos:

1. Crear una carpeta de nombre ***tallerscilab***.
2. Seleccione la carpeta ***tallerscilab*** como el directorio de trabajo haciendo clic en ***File***, ***Change current directory*** y seleccionando la carpeta en el explorador de archivos.





3. Verifique que el directorio de trabajo ha cambiado por medio del comando ***pwd***

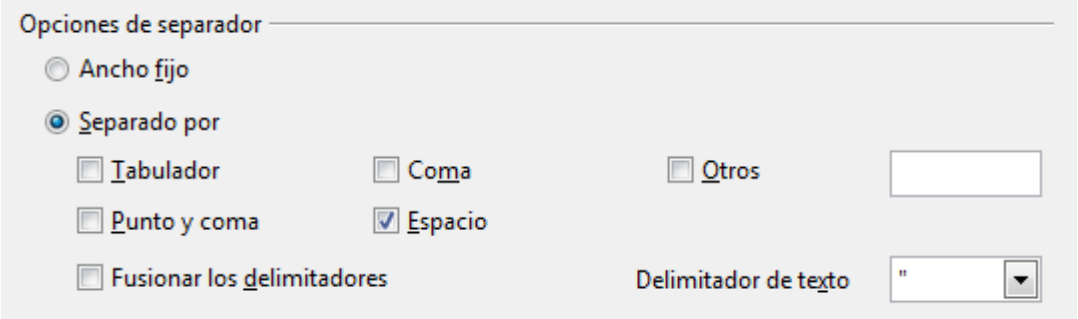
B. Escritura de Archivos

Siga los siguientes pasos:

1. Digite las siguientes instrucciones en un script de nombre ***ejemploclaseuno*** y almacenelo en la carpeta ***tallerscilab.*** Ejecute el script.

```
// Abre un archivo para escritura, si no existe lo crea
fid = mopen('tipos.dat', "w");
if (fid == -1)
   error('cannot open file for writing');
end
// Escribe en el archivo %s: string, %d entero, %f: float
mfprintf(fid, "%s %s \n", 'Tipo', 'Valor');
mfprintf(fid, "%s %d \n", 'Entero', 120);
mfprintf(fid, "%s %f \n", 'Float', 3.141592653589793115998);
mfprintf(fid, "%s %1.8f \n", 'Float', 3.141592653589793115998);
// Cierra el archivo
mclose(fid);
```

2. Verifique que el archivo **tipos.dat** ha sido creado en la carpeta ***tallerscilab***. Abra el archivo con un editor de texto.
3. Abra el archivo **tipos.dat** empleando un programa de hojas de calculo (***libreoffice calc***, ***openoffice calc***, ***microsoft excel***). En las opciones seleccione, separado por espacios.



El archivo creado desde Scilab puede ser abierto por un programa de hojas de cálculo. En el desarrollo de programas de ingeniería es útil la creación de archivos para el almacenamiento y presentación de resultados, en donde el cliente puede usar herramientas convencionales para la visualización (hojas de cálculo)

Siga los siguientes pasos:

1. Digite las siguientes instrucciones en un script de nombre ***ejemploclasedos*** y almacenelo en la carpeta ***tallerscilab.*** Ejecute el script.

```
// Abrir un archivo para escritura, si no existe se crea automáticamente
fid = mopen('exp.dat', "w");
if (fid == -1)
   error('cannot open file for writing');
end
// Escribir en el archivo %f: float
t = 0:0.05:1
y = exp(t);
for i=1:length(t)
   mfprintf(fid, "%f %f \n", t(i), y(i));
end
// Cerrar el archivo
mclose(fid);
```

2. Verifique que el archivo ***exp.dat*** ha sido creado en la carpeta ***tallerscilab***. Abra el archivo con un editor de texto.
3. Abra el archivo ***exp.dat*** empleando un programa de hojas de calculo (***libreoffice calc***, ***openoffice calc***, ***microsoft excel***). En las opciones seleccione, separado por espacios.

El archivo creado desde Scilab puede ser abierto por un programa de hojas de cálculo. Los programas de hoja de cálculo permiten obtener información adicional a partir de los datos como por ejemplo estimación de promedios, desviación estándar, entre otras.

C. Lectura de Archivos

Una vez haya realizado el ejercicio anterior y habiendo comprobado el contenido del archivo ***exp.dat*** en una hoja de cálculo, digite las siguientes instrucciones en un script de nombre ***ejemploclasetres*** y almacenelo en la carpeta ***tallerscilab***. Ejecute el script.

```
// Abrir un archivo para escritura, si no existe se crea automáticamente
fid = mopen('exp.dat', "r"); // El parametro r indica que se abrira el archivo para lectura
if (fid == -1)
   error('cannot open file for writing');
end
// Leer el archivo
[n,t,y]=mfscanf(fid,'%e %e') // lee una linea del archivo
l1=mfscanf(fid,'%e %e')     //lee la siguiente linea del archivo
l2=mfscanf(5,fid,'%e %e')  //lee 5 lineas del archivo
l3=mfscanf(-1,fid,'%e %e')  // lee hasta el final del archivo
// Cerrar el archivo
mclose(fid);
```

Las instrucciones anteriores permiten leer la información del archivo ***exp.dat***. La información leída puede ser asignada a variables de Scilab. Verifique los valores que han sido almacenados en n, t, y, l1, l2 y l3 tras la ejecución del script.

Nota: Si usa una función recuerde devolver los parámetros en la forma ***[n, t, y, l1, l2, l3] = mifuncion()***

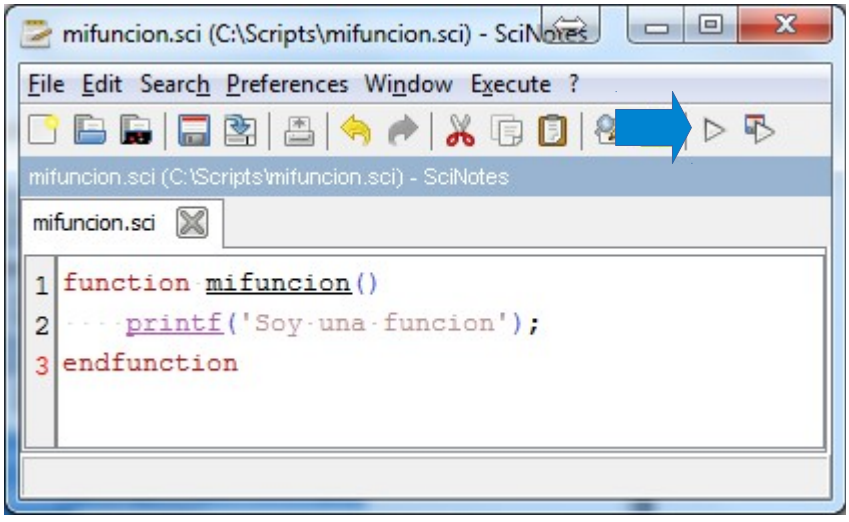
Funciones
Scilab permite la creación de funciones. Para crear una función se sigue el mismo procedimiento para la creación de un script.

Cree una carpeta de nombre **Funciones** en el escritorio.

Cree un nuevo archivo (script), asignele como nombre **mifuncion** y almacénelo dentro de la carpeta **Funciones** que creo anteriormente. Digite las siguientes instrucciones.

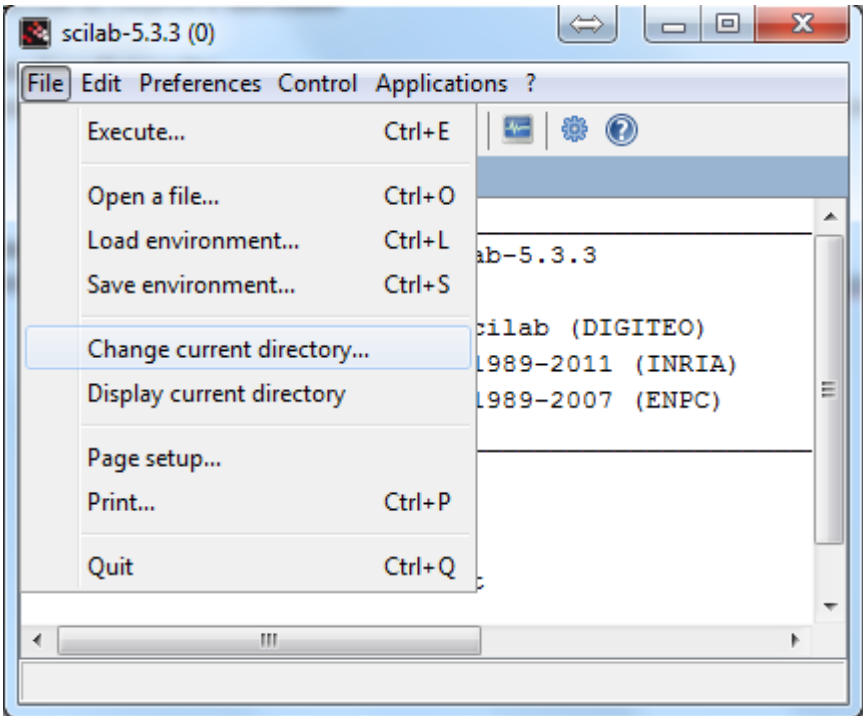
```
function mifuncion()
   printf('Soy una funcion');
endfunction
```

Haga clic en el icono del triangulo

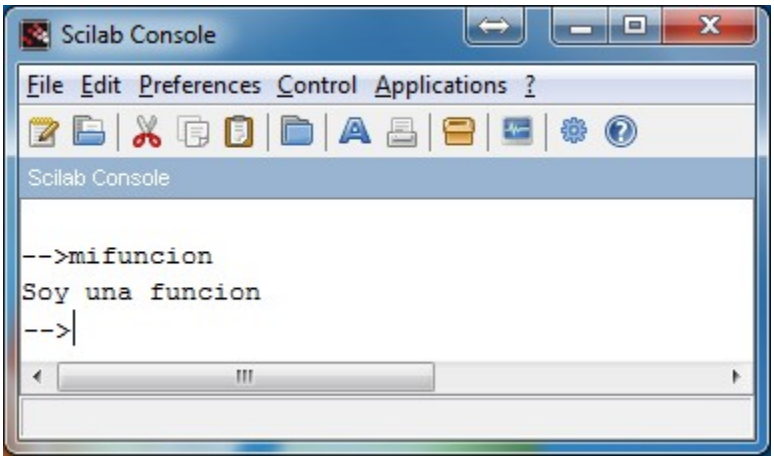


Nota: En versiones anteriores de Scilab deberá hacer click en **Execute**, **Load into Scilab.**

En la consola de scilab selecione el menú File → Change Current Directory y seleccione como directorio actual la carpeta **Funciones**



Digite en la consola el nombre de la función que creo anteriormente (**mifuncion**):



Nota: Al digitar el comando **pwd** en la consola puede consultar el directorio de trabajo actual

Parámetros de una función
Scilab permite crear funciones que reciben uno o más datos.

Digite las siguientes instrucciones en un nuevo script y almacénelo con el nombre **sumatoria**. Haga clic en el simbolo del triangulo, si es necesario cambie el directorio actual de trabajo y ejecutelo desde

la consola con el comando sumatoria(1, 2, 5), donde los números 1, 2 y 5 son entradas a la función.

```
function sumatoria(valor1, valor2, valor3)
  sumatoria = valor1 + valor2 + valor3;
  printf('%d',sumatoria);
endfuncion
```

Scilab permite crear funciones que retornan uno o más datos.

Digite las siguientes instrucciones en un nuevo script y almacenelo con el nombre **operaciones**. Haga clic en el símbolo del triangulo, si es necesario cambie el directorio actual de trabajo y ejecútelo desde la consola con el comando [suma, mult] = operaciones(2, 5), donde los números 2 y 5 son entradas a la función.

```
function [suma, mult] = operaciones(valor1, valor2)
   suma = valor1 + valor2;
   mult = valor1*valor2;
endfunction
```

Estructuras Condicionales
Scilab permite el empleo de estructuras condicionales (if, if-else, switch) en la creación de scripts

Digite las siguientes instrucciones en un nuevo script y almacenelo con el nombre **espar**. Haga clic en el símbolo del triangulo, si es necesario cambie el directorio actual de trabajo y ejecútelo desde la consola con el comando espar(8), donde el número 8 es una entrada a la función.

```
function espar(numero)
residuo = modulo(numero, 2);
if residuo == 0 then
   printf('%d es par',numero);
else
   printf('%d es impar', numero);
end
endfunction
```

Digite las siguientes instrucciones en un nuevo script y almacénelo con el nombre **clasificarUmbral**. Haga clic en el símbolo del triangulo, si es necesario cambie el directorio actual de trabajo y ejecútelo desde la consola con el comando clasificarUmbral(40), donde el número 40 es una entrada a la función.

```
function clasificarUmbral(umbral)
if  umbral < 0 then
   disp('nivel bajo')
elseif umbral >= 0 & umbral <= 10 then
   disp('nivel medio')
elseif umbral > 10 & umbral <= 20 then
   disp('nivel alto')
else
```

```
   disp('nivel crítico')
end
endfunction
```

Digite las siguientes instrucciones en un nuevo script y almacenelo con el nombre **leerOpcion**. Haga clic en el símbolo del triangulo, si es necesario cambie el directorio actual de trabajo y ejecútelo desde la consola con el comando leerOpcion(1), donde el número 1 es una entrada a la función.

```
function leerOpcion(opcion)
  select opcion,
     case 1 then
        disp('Opcion 1');
     case 2 then
        disp('Opcion 2');
     case 3 then
        disp('Opcion 3');
     else
        disp('Opcion no valida');
  end
endfunction
```

Estructuras de Ciclos
Scilab permite el empleo de estructuras de ciclos (for, while) en la creación de scripts

Digite las siguientes instrucciones en un nuevo script y almacenelo con el nombre **mostrarlistado**. Haga clic en el símbolo del triangulo, si es necesario cambie el directorio actual de trabajo y ejecútelo desde la consola con el comando listado(10), donde el número 10 es una entrada a la función.

```
function mostrarlistado(limite)
   for i = 1:2:limite
      printf('%d\n',i);
   end
endfunction
```

Digite las siguientes instrucciones en un nuevo script y almacenelo con el nombre **solucioneuler**. Haga clic en el símbolo del triangulo, si es necesario cambie el directorio actual de trabajo y ejecútelo desde la consola con el comando solucioneuler(12.5, 68.1, 2, 14), donde el número 12.5 corresponde al coeficiente de arraste, el número 68.1 a la masa, el número 2 al stepsize, y el número 14 al tiempo final a evaluar.

```
function solucioneuler(cd, mass, stepsize, tfinal)
// Esta funcion encuentra una solucion aproximada al problema de caida libre visto en clase.
g = 9.81;
v(1) = 0;
t = 0:stepsize:tfinal;
for i=2:length(t)
   v(i) = v(i-1) + (g-(cd/mass)*v(i-1))*(t(i)-t(i-1));
```

```
end
plot(t,v,'color','red','marker','>');
xlabel("t, s");
ylabel("v, m/s");
set(gca(),"grid",[1 1]);
endfunction
```

Problemas

1. Realice una función que reciba dos matrices y un valor de opción. Si la opción es 1, retorne la suma de las matrices, si la opción es dos retorne la resta de las matrices y si la opción es tres retorne la multiplicación de las matrices, si la opción es distinta retorne una matriz de ceros y muestre un mensaje de opción invalida.
2. Realice una función para encontrar una solución aproximada por medio del método de euler empleando una relación de segundo orden para FU = cd*v^2.
3. Seleccione e implemente un ejercicio del anexo problemaCap3.pdf