

**Introduccion a la Programacion con SMath Studio**

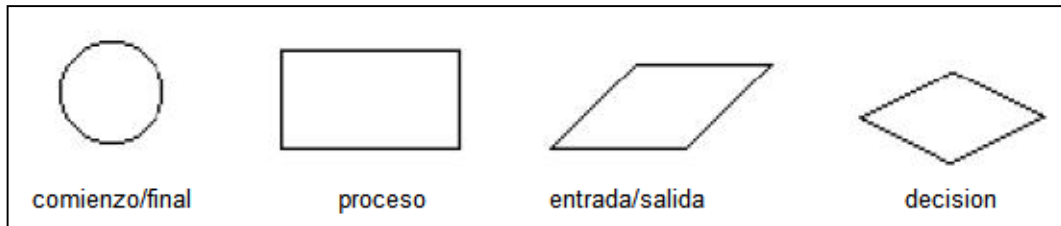
**Gilberto E. Urroz, Profesor Asociado, Departamento de Ingenieria Civil y Ambiental, Universidad Estatal de Utah, Logan, UT, EEUU, Julio 2012**

En esta seccion se presentan los conceptos basicos de la programacion para metodos numericos en SMath Studio.

**Estructuras de programacion y diagramas de flujo**

La programacion, en el contexto de aplicaciones numericas, implica el control de la computadora (ordenador), u otros equipos de calculo, para producir un resultado numerico. En este contexto, se reconocen tres estructuras principales de programacion, a seguir: (a) estructuras sequenciales; (b) estructuras de decision; y (3) estructuras de repeticion (o estructuras de lazo). Muchos calculos auxiliados por computadoras u otros equipos de calculo programables (por ejemplo, calculadoras) se pueden efectuar usando una o mas de estas estructuras, o sus combinaciones.

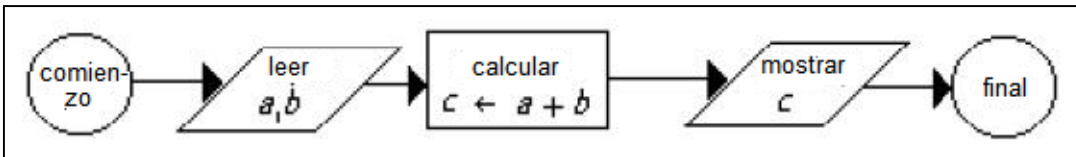
La operacion de estas estructuras de programacion se ilustraran con el uso de diagramas de flujo. El diagrama de flujo es una representacion grafica del proceso a programarse. El diagrama ilustra el flujo del proceso de programacion, de donde surge el nombre de diagrama de flujo. La figura siguiente ilustra algunos de los simbolos usados con mas frecuencia en los diagramas de flujo.



En un diagrama de flujo, estos simbolos se conectaran usando lineas y flechas que indican la direccion del flujo de procesamiento del programa.

**Estructuras sequenciales**

Un digrama de flujo completo consiste de puntos de comienzo y final, y, al menos un bloque de procesamiento entre ellos. Tal diagrama constituiria el caso mas simple de una estructura secuencial. La figura siguiente ilustra una estructura secuencial en el calculo de una simple suma.

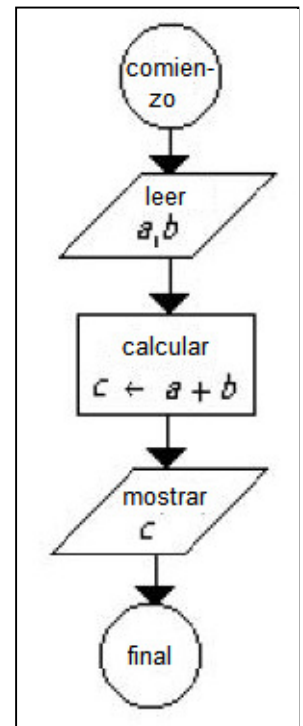


Es muy comun mostrar las estructuras sequenciales en direccion vertical como se ilustra en esta figura: ----->

La estructura secuencial mostrada en este diagrama de flujo puede ser representada usando pseudo-codigo. El pseudo-codigo consiste en escribir el proceso a programarse de forma que se asemeje al lenguaje comun, por ejemplo:

```

Inicio
  Leet a, b
  c ← a + b
  Mostrar c
Final
    
```



El diagrama de flujo, o el psudo-codigo, pueden traducirse en codigo (lenguaje de programacion) en formas diferentes, dependiendo del lenguaje de programacion utilizado. En SMath Studio, esta estructura secuencial puede traducirse como se ilustra a continuacion:

```

-----
a:= 2   b:= 3   // Input a,b
c:= a+b           // c <- a + b
c= 5           // Display c
-----

```

He aqui otro ejemplo de una estructura secuencial en SMath Studio mostrando mas de un paso de calculo. La estructura secuencial en SMath Studio no tiene que seguir una direccion vertical estricta, como se ilustra a continuacion:

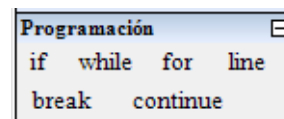
```

-----
x1:=- 10           y1:= 2
x2:= 5             y2:=- 3
Δx:= x2- x1       Δx= 15
Δy:= y2- y1       Δy=- 5
d1:=√Δx2+ Δy2    d1= 15.81
-----

```

### El comando "line" y el panel de Programacion

Los ejemplos anteriores ilustraron las estructuras secuenciales. En SMath Studio, se pueden coleccionar los pasos de calculo de un programe en una linea de programacion. La figura a la derecha -----> muestra las instrucciones para incluir una linea de programacion (line) en un cuaderno de SMath Studio. El comando "line", junto con otros comandos de programacion, estan disponibles en el panel de Programacion.

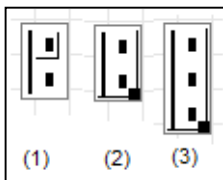


El comando "line" puede activarse de una de estas formas:

- Usando "line" en el panel de Programacion
- Escribiendo "line(" en el cuaderno

### **Creacion de una linea de programacion (line) - agregar puntos de entrada**

Por defecto, el comando "line" produce una linea de programacion vertical con dos puntos de entrada, como se ilustra en la figura (1), a continuacion:



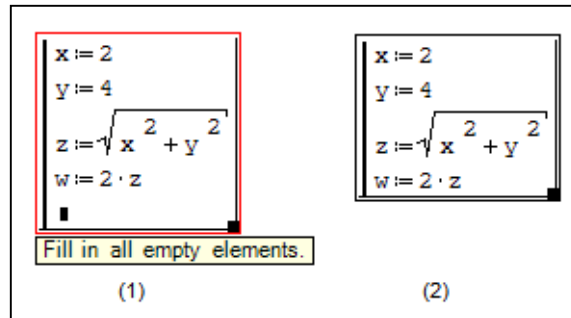
Para agregar puntos de entrada adicionales a una linea de programacion:

- Pulse entre los dos puntos de entrada, o a la izquierda de la linea de programacion
- Arrastre el boton de la esquina inferior derecha que aparece al pulsar
- Un nuevo punto de entrada sera' agregado

Repita pasos (1) y (2) para agregar cuanto puntos de entrada sean necesarios.

**Remover puntos de entrada de una linea de programacion**

Si se ha extendido la linea de programacion a mas de los puntos de entrada necesarios, se puede reducir el numero de puntos de entrada pulsando en el boton de la esquina inferior derecha, y arrastrandolo hacia arriba hasta que los puntos de entrada que no se necesitan sean removidos. La siguiente figura muestra los dos pasos descritos anteriormente:

Uso del comando "line"

El comando "line" se puede utilizar para mantener agrupados una serie de comandos de una estructura de programacion secuencial en un cuaderno de SMath Studio. Por ejemplo, los comandos que calculan  $\Delta x$ ,  $\Delta y$ , y  $d2$ , en el listado que sigue a continuacion han sido colocados dentro de una linea de programacion (line):

```
-----
xA:=- 10   yA:= 2
xB:= 5     yB:=- 3

|
| Δxx:= xA- xB
| Δyy:= yA- yB
| d2:=√ Δxx2 + Δyy2
|
d2= 15.81
-----
```

En el ejemplo anterior, el unico proposito al usar la linea de programacion (line) es mantener agrupados las tres lineas de calculo mostradas. A este punto se puede seleccionar ese grupo y moverlos como una unidad de programacion. A continuacion se muestra el uso del comando "line" para definir funciones.

Definicion de funciones con un comando "line"

SMath Studio permite la definicion de funciones cuyo contenido esta' definido por una secuencia de comandos de programacion. Por ejemplo, la siguiente funcion,  $f1(x,y)$ , se defined como una estructura secuencial contenida dentro de un comando "line":

```
-----
a:= 2     b:= 5

f1(x, y):= |
| r:= a·x + b·y
| s:= b·x + a·y
| √ r2 + s2
|
-----
```

Ejemplos de calculos con  $f1(x,y)$ :       $f1(-2, 4)=16.12$        $f1(-2, -4)=30$   
     $f1(2, -10)=47.07$        $f1(-2, 10)=47.07$

He aqui otro ejemplo del uso de una linea de programacion para definir una funcion. La funcion  $Rh(D, y)$  calcula el radio hidraulico de un canal abierto de seccion circular dado el diametro,  $D$ , y la profundidad de flujo,  $y$ :

$$Rh(D, y) := \begin{cases} \theta := \arccos\left(1 - 2 \cdot \frac{y}{D}\right) \\ A := \frac{D^2}{4} \cdot (\theta - \sin(\theta) \cdot \cos(\theta)) \\ P := D \cdot \theta \\ \frac{A}{P} \end{cases}$$

Calculos con  $Rh(D, y)$  :

$$\begin{aligned} Rh(10, 5) &= 2.5 & Rh(3.5, 0.5) &= 0.31 \\ Rh(5, 2) &= 1.07 & Rh(1.2, 0.1) &= 0.06 \end{aligned}$$

### Estructura de decision

La estructura de decision provee caminos alternos para el flujo de un proceso basados en el valor de "verdadero" o "falso" de una expresion logica. Como ejemplo de una estructura de decision, considerese el diagrama de flujo de la funcion mostrada a continuacion:

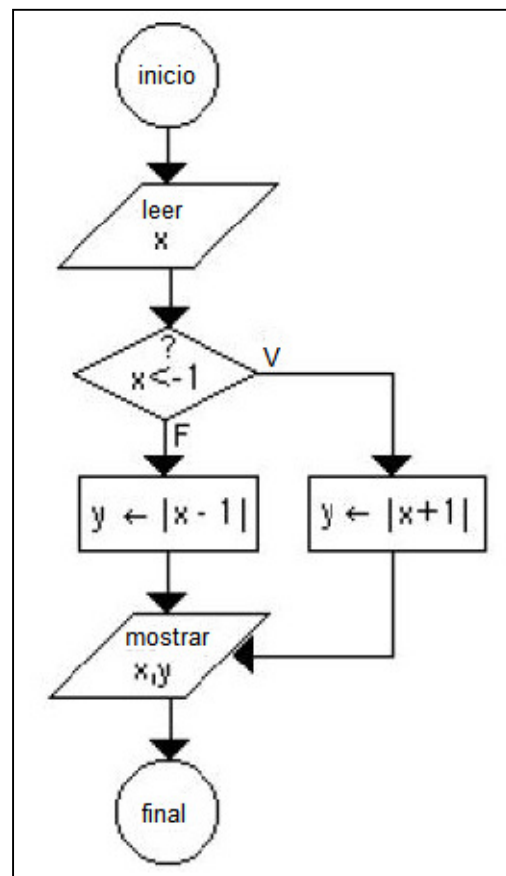
$$f(x) = \begin{cases} |x+1|, & \text{if } x < -1 \\ |x-1|, & \text{if } x \geq -1 \end{cases}$$

El diagrama de flujo se muestra a la derecha --> El pseudo-codigo correspondiente se muestra a continuacion. Mantenemos la estructura if-then-else, en ingles, pues los comandos usan esas palabras en ingles. La estructura se escribiria como "si-entonces-si no" en Espanol:

```

inicio
  leer x
  if x < -1 then
    y <- |x+1|
  else
    y <- |x-1|
  mostrar x,y
end

```



En SMath Studio la estructura de decision se activa usando el comando "if". Las instrucciones para escribir el comando "if" se presentan a continuacion:

ESTRUCTURAS DE DECISION - El comando "if":

El comando "if" se puede escribir:

- (1) Utilizando "if" en el panel de Programacion
- (2) Escribiendo "if(condicion, verdadera, falsa)"

El utilizar "if" en el panel de Programacion produce esta estructura:----->

```
if █
  █
else
  █
```

La operacion general del comando "if" es como sigue:

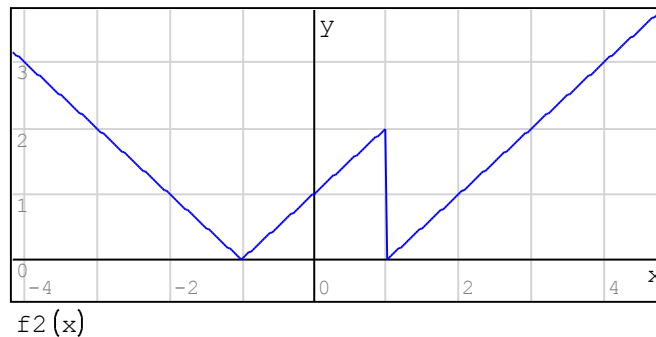
```
if condicion
  comando(s) 1
else
  comando(s) 2
```

- \* La "condicion" es una expresion logica que puede ser verdadera (true) o falsa (false).
- \* Si la "condicion" es verdadera se ejecutan el o los "comando(s) 1"
- \* Si la "condicion" es falsa se ejecutan el o los "comando(s) 2", asociados con la palabra "else", por defecto

Para ilustrar el uso del comando "if" en SMath Studio, definase la funcion  $f(x)$ , indicada anteriormente, como se muestra a continuacion:

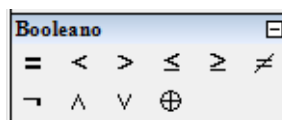
```
-----
f2(x):= if x<1
        |x+1|
        else
        |x-1|
-----
```

La grafica de esta funcion se muestra a continuacion:

Operadores de comparacion y operadores logicos - El panel "Booleano"

Las estructuras de decision requieren de una condicion que active una decision en el flujo del programa. Tal condicion se representa por una expresion logica. Dentro del contexto de la programacion de calculos numericos, una expresion logica es una expresion matematica que puede ser verdadera o falsa, por ejemplo,  $3 > 2$ ,  $5 < 2$ , etc. En SMath Studio los resultados logicos "verdadero" y "falso" se representan por los numeros 1 y 0, respectivamente.

Entre las expresiones logicas mas elementales se encuentran aquellas que resultan al comparar numeros reales. Los operadores de comparacion y los operadores logicos se encuentran disponibles en el panel "Booleano" en SMath Studio:



La línea superior del panel "Booleano" contiene los operadores de comparación, mientras que la línea inferior del mismo panel contiene los operadores lógicos. Los operadores de comparación son: igual (=), menor que (<), mayor que (>), etc. Los operadores lógicos son: negación (¬), conjunción (and), disjunción (or), y el o exclusivo (xor).

### Ejemplos de operaciones de comparación

Los siguientes ejemplos muestran operaciones que resultan en los valores lógicos de 0 o 1:

Desigualdades:  $3 > 2 = 1$  // verdadero  $3 < 2 = 0$  // falso

Igualdad booleana y la no-igualdad:

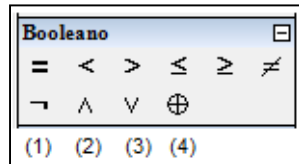
$3 = 2 = 0$  // falso  $3 \neq 2 = 1$  // verdadero

Menor que o mayor que:

$5 \geq \pi = 1$  // verdadero  $5 \leq \pi = 0$  // falso

### Los operadores lógicos y las tablas de verdad

el panel "Booleano" en SMath Studio incluye los siguientes cuatro operadores lógicos:



(1) negacion (no):	$\neg(3 < 2) = 1$	$\neg(3 > 2) = 0$
(2) conjuncion (y):	$(3 > 2) \wedge (4 > 3) = 1$	$(3 < 2) \wedge (4 > 3) = 0$
(3) disjuncion(o):	$(3 > 2) \vee (5 < 2) = 1$	$(3 < 2) \vee (5 < 2) = 0$
(4) o exclusivo (xor):	$(3 < 2) \oplus (2 > 1) = 1$	$(3 > 2) \oplus (2 > 1) = 0$

Al operador "negacion" se le conoce como un operador "unario" (unitario?) porque aplica a una sola expresión lógica exclusivamente. A los otros tres operadores (conjunción, disjunción, y o exclusivo) se les conoce como operadores "binarios", porque requieren de dos expresiones lógicas para poder activarse.

El término "tablas de verdad" se refiere a los resultados de los diferentes operadores lógicos. Por ejemplo, la negación de una expresión verdadera es falsa, y vice versa. Dado que en SMath Studio 1 significa "verdadero" y 0 "falso", se muestran a continuación las tablas de verdad de los cuatro operadores lógicos:

<u>Negacion (no):</u>	<u>Conjuncion (y):</u>
$\neg 1 = 0$	$1 \wedge 1 = 1$
$\neg 0 = 1$	$1 \wedge 0 = 0$
	$0 \wedge 1 = 0$
	$0 \wedge 0 = 0$
<u>Disjuncion (o):</u>	<u>O exclusivo (xor):</u>
$1 \vee 1 = 1$	$1 \oplus 1 = 0$
$1 \vee 0 = 1$	$1 \oplus 0 = 1$
$0 \vee 1 = 1$	$0 \oplus 1 = 1$
$0 \vee 0 = 0$	$0 \oplus 0 = 0$

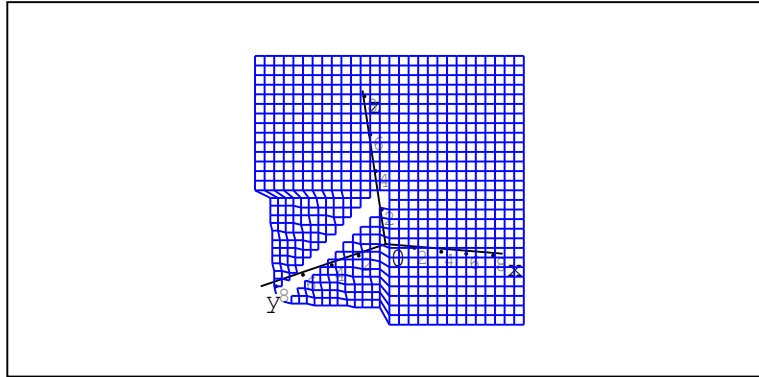
### Ejemplos del comando "if" con operaciones lógicas

La función  $g1(x,y)$ , mostrada a continuación, se define usando las expresiones lógicas " $(x < 0)$  and  $(y < 0)$ ". Evaluaciones de esa función, y una gráfica de la misma, siguen:

```
g1(x, y) := if (x < 0) ^ (y < 0)
           sqrt(x^2 + y^2)
           else
           0
```

$$g1(2, 0) = 0$$

$$g1(-3, -2) = 3.61$$



$g1(x, y)$

### Comandos "if" anidados

Si un diagrama de decision incluye mas de una condicion, puede ser necesario "anidar" un comando "if" dentro de otra como se ilustra en la definicion de la funcion  $s(x, y)$ , a continuacion:

```
s(x, y) := if x > 0
           if y > 0
           sqrt(x + y)
           else
           sqrt(2 * x + y)
           else
           sqrt(3 * x + y)
```

En este caso, si la condicion " $x > 0$ " es verdadera, es necesario entonces verificar el comando "if" mas interno, es decir:

```
if y > 0
sqrt(x + y)
else
sqrt(2 * x + y)
```

cuya operacion se discutió anteriormente.

Por lo tanto, para la funcion  $s(x, y)$ , su evaluacion procede como se muestra a continuacion:

\* si  $x > 0$  y si  $y > 0$ , entonces  $s = \sqrt{x + y}$  , e.g.,  $s(2, 2) = 2$

\* si  $x > 0$  y si  $y < 0$ , entonces  $s = \sqrt{2 \cdot x + y}$  , e.g.,  $s(3, -2) = 2$

\* si  $x < 0$ , entonces  $s = \sqrt{3 \cdot x + y}$  , e.g.,  $s(-2, 10) = 2$  , or,  $s(-2, -2) = 2.83 \cdot i$

Combinacion del comando "if" con el comando "line"

Las condiciones "verdadero" o "falso" en un comando "if" pueden resultar en la ejecucion de mas de una expresion como se ilustra en los siguientes ejemplos, en los cuales se utilizan comandos "line" para ejecutar mas de una operacion:

caso (a):  $xx < yy$ ,  
intercambia  $xx$  and  $yy$ :

$xx := 3$     $yy := 4$

```
if xx < yy
| temp := xx
| xx := yy
| yy := temp
else
| xx := -xx
| yy := -yy
```

$xx = 4$     $yy = 3$

caso (b):  $x > y$ , cambiar el  
signo de  $x$  y  $y$ :

$x := 4$     $y := 3$

```
if x < y
| t := x
| x := y
| y := t
else
| x := -x
| y := -y
```

$x = -4$     $y = -3$

Si quisieramos convertir este comando "if" en una function, debemos recordar que una function siempre produce un solo valor. Para poder producir mas de un valor, tenemos que poner nuestros resultados,  $x$  y  $y$ , en un vector columna de dos lineas (filas), por ejemplo:

```
f3(x, y) := if x < y
| t := x
| x := y
| y := t
else
| x := -x
| y := -y
| (x)
| (y)
```

Ejemplos de evaluacion de esta function son:

Caso (a),  $x < y$ :

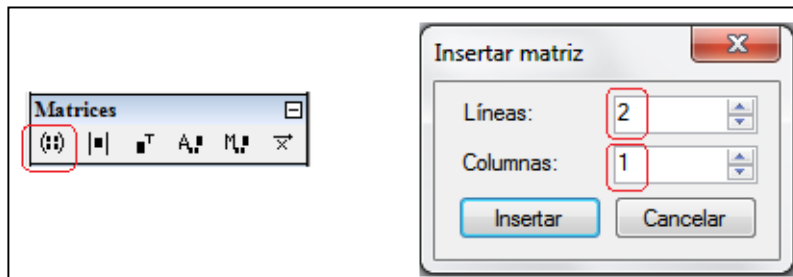
$$f3(3, 4) = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

Caso (b),  $x > y$ :

$$f3(4, 3) = \begin{pmatrix} -4 \\ -3 \end{pmatrix}$$

NOTE: To enter a vector you need to use the "Matrix (Cntl+M)" icon in the "Matrices" palette. Then, enter the number of rows and columns in the resulting entry form, and press [Insert]. (See figure below)

NOTA: Para escribir un vector utilicese el icono "Matriz 3x3 (Cntl+M)" en el panel de Matrices. Despues, escriba el numero de lineas y de columnas en los campos correspondientes, y presione el boton [Insertar].



Una vez hecho esto, escriba las componentes del vector o matriz en los puntos de entrada apropiados.



Estructuras de repeticion (o de lazo)

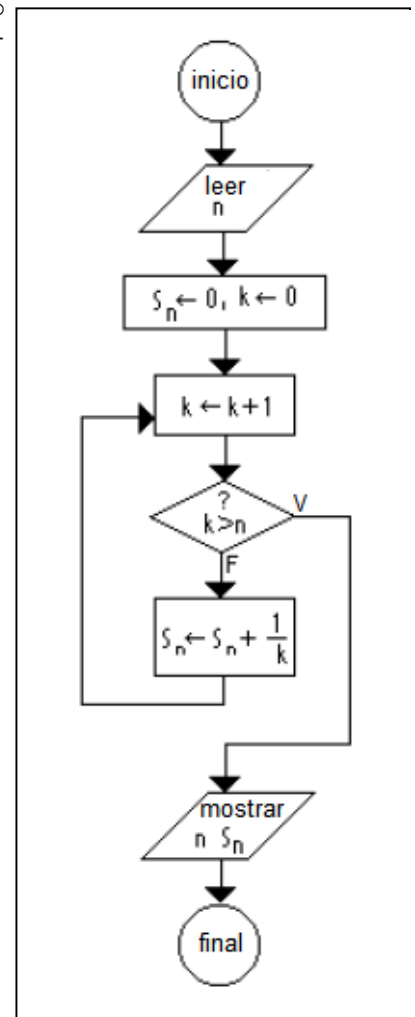
En una estructura de lazo el flujo de proceso se repite un numero finito de veces antes de salir del lazo. La parte media del diagrama de flujo a la derecha muestra una estructura de lazo. El diagrama de flujo en cuestion representa el calculo de una sumatoria, es decir,

$$S_n = \sum_{k=1}^n \frac{1}{k}$$

La sumatoria  $S_n$  se inicializa como  $S_n \leftarrow 0$ , y se inicializa asi mismo un indice,  $k$ , como  $k \leftarrow 0$ , antes de pasar el control al lazo. El lazo comienza a incrementar  $k$  y luego verifica si el indice  $k$  es mayor que su valor maximo  $n$ . La suma se incrementa dentro del lazo,  $S_n \leftarrow S_n + 1/k$ , y el proceso se repite hasta que la condicion  $k > n$  se satisface ( $V = \text{verdadero}$ ), el control, entonces, se envia fuera del lazo hacia el bloque de salida.

En terminos de comandos de programacion existen dos tales comandos en SMath Studio para programar un lazo: `while` y `for`. The pseudo-codigo para un lazo "while", que interpreta el diagrama de flujo de la derecha, se muestra a continuacion:

```
start
input n
Sn <- 0
k <- 0
do while ~(k>n)
  k <- k + 1
  Sn <- Sn + 1/k
end loop
display n, Sn
end
```



Dado que un comando de lazo "while" verifica la condicion de salida al inicio del lazo, la condicion dada en el pseudo-codigo y en el diagrama de flujo, es decir,  $k > n$ ?, fue convertida a  $\neg(k > n)$ ?, es decir,  $\text{no}(k > n) = k \leq n$

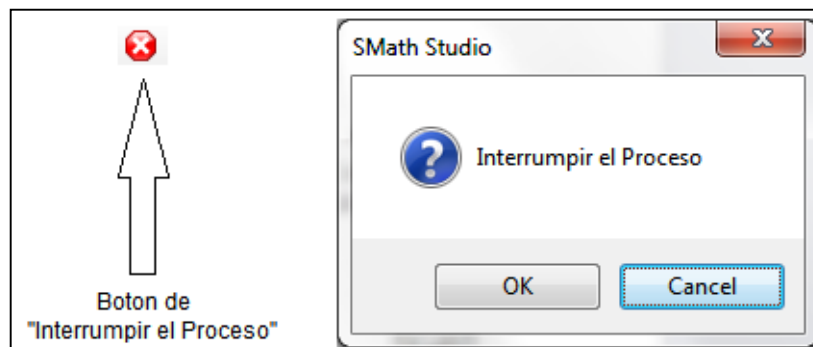
El comando "while" ("mientras") en SMath Studio

El comando "while" se puede agregar ya sea:

- (1) Usando "while" en el panel de Programacion
- (2) Escribiendo "while(condicion, contenido)"

```
while [
  ]
```

La expresion que define la "condicion" debe modificarse dentro del lazo "while" de manera que se provea una salida del lazo. De otra manera, se puede caer en un lazo infinito que solo puede detenerse usando el boton de "Interrumpir el proceso" disponible en el menu de SMath Studio (vease la figura a continuacion):



Ejemplo: sumense los numeros pares del 2 al 20

```

S00:= 0           //Inicializar suma (S00)
k:= 2            //Inicializar indice (k)

while k≤20
  S00:= S00+ k    //Lazo "while" con
  k:= k+ 2       comando "line"

k= 22   S00= 110   //Resultados
    
```

Esta operacion se puede calcular usando el simbolo de sumatoria en el panel de Funciones:

$$S := \sum_{k=1}^{10} (2 \cdot k) \qquad S = 110$$

Lazos "while" anidados

Los lazos "while" se pueden "anidar" como se ilustra en el ejemplo mostrado a continuacion (en el lado izquierdo). La doble sumatoria correspondiente se muestra en el lado derecho:

Doble sumatoria con lazos "while" anidados:

```

S01:= 0   k:= 1   j:= 1

while k≤5
  j:= 1
  while j≤5
    S01:= S01+ k·j
    j:= j+ 1
  k:= k+ 1

S01= 225
    
```

El simbolo de doble sumatoria calcula la suma:

$$S_{05} := \sum_{k=1}^5 \sum_{j=1}^5 (k \cdot j)$$

S05 = 225

Diagrama de flujo para el lazo "for"

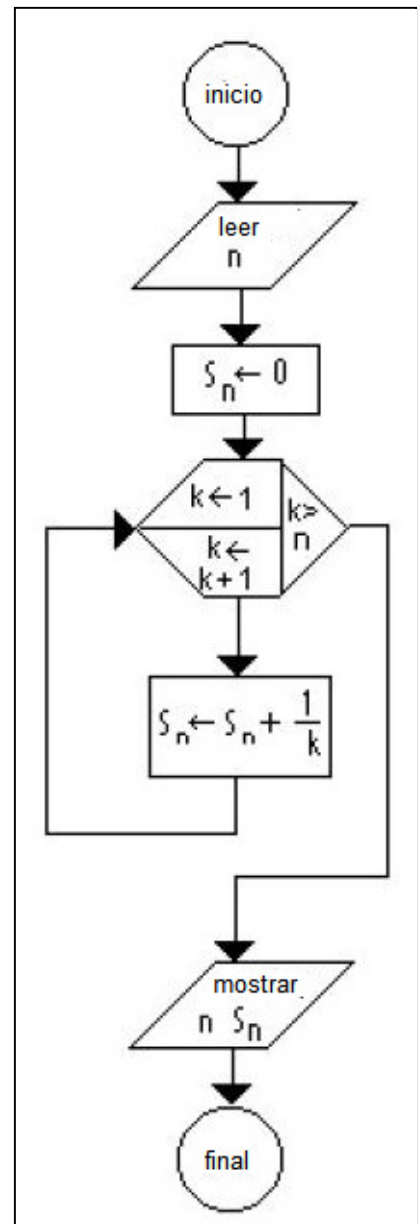
La figura a la dercha muestar un diagrama de flujo alternativo para calcular la sumatoria:

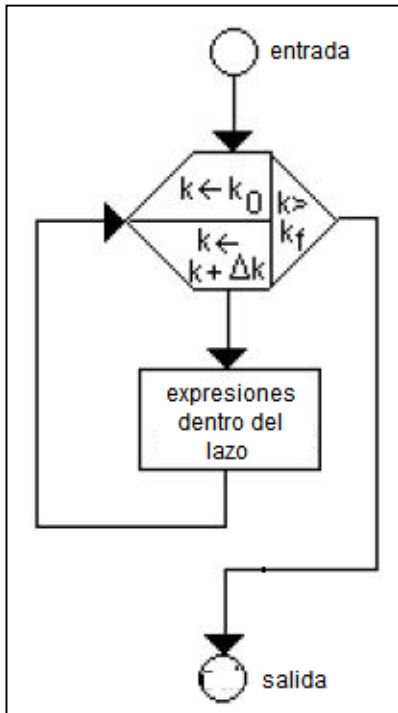
$$S_n = \sum_{k=1}^n \frac{1}{k} \quad \text{----->}$$

El simbolo hexagonal en el diagrama de flujo muestra tres elementos:

- (1) la inicialization del indice,  $k \leftarrow 1$ ;
- (2) el incremento en el indice,  $k \leftarrow k + 1$ ; y,
- (3) la condicion a verificar para salir del lazo,  $k > n$ .

Este simbolo hexagonal representa el comando "for" en el diagrama de flujo de la sumatoria en cuestion.





El indice toma los valores:

$$k = k_0, k_0 + \Delta k, k_0 + 2 \cdot \Delta k, \dots, k_{end},$$

de manera que  $k_{end} \leq k_f$  por una cantidad menor que  $\Delta k$ .

#### El comando "range" (extension) en SMath Studio

El comando "range" produce un vector de indices necesario para establecer un lazo "for". El comando "rango", por lo tanto, se define aqui primero.

El vector generado por el comando "range" tiene elementos que siguen una formula particular. Para activar el comando "range" usese:

- (1) range(valor inicial, valor final)  
resulta en: valor inicial..valor final (incremento = 1)
- (2) range(valor inicial, valor final, inicial+incremento)  
resulta en: valor inicial, inicial+incremento..valor final

El comando "range" genera un vector columna. A continuacion se usan vectores transpuestos para mostrar el resultado del comando "range" como vectores fila:

#### **Ejemplos del comando "range" con incremento unitario:**

//Escribir "range(2,5)" produce: `r1:= 2 .. 5`       $r1^T = (2 \ 3 \ 4 \ 5)$

//Escribir "range(10,18)" produce: `r2:= 10 .. 18`       $r2^T = (10 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16 \ 17 \ 18)$

#### **Ejemplos del comando "range" con incremento positivo:**

// Escribir "range(2,18,4)" produce: `r3:= 2 , 4 .. 18`       $r3^T = (2 \ 4 \ 6 \ 8 \ 10 \ 12 \ 14 \ 16 \ 18)$

// Escribir "range(20,300,80)" produce: `r4:= 20 , 80 .. 300`       $r4^T = (20 \ 80 \ 140 \ 200 \ 260)$

#### **Ejemplos del comando "range" con incremento negativo (decremento):**

// Escribir "range(100,20,80)" produce: `r5:= 100 , 80 .. 20`       $r5^T = (100 \ 80 \ 60 \ 40 \ 20)$

// Escribir "range(5,1,4)" produce: `r6:= 5 , 4 .. 1`       $r6^T = (5 \ 4 \ 3 \ 2 \ 1)$

#### El comando "for" (para) en SMath Studio

El comando "for" se puede escribir:

- (1) Usando "for" en el panel de Programacion
- (2) Escribiendo "for(indice,range(extension),bloque de expresiones)"

for  $\mathbf{k} \in \mathbf{r}$   
 $\mathbf{e}$

He aqui un ejemplo del comando "for" en SMath Studio usando la extension (range) 1..10 para calcular una sumatoria:

$$S03 = \sum_{k=1}^{10} 2 \cdot k$$

```
-----
S03:=0           // inicializar una sumatoria (S03)

for k∈ 1 .. 10   //lazo "for", escribir la extension
  S03:= S03+ 2·k //como: 'range(1,10) '

S03= 110        // valor final de S03
-----
```

Lazos "for" anidados

Como en el caso de los lazos "while", los lazos "for" tambien pueden escribirse anidados, como se ilustra en el siguiente ejemplo en el que se calcular una doble sumatoria:

$$S_{04} = \sum_{j=1}^5 \sum_{k=1}^5 (k \cdot j)$$

```
-----
S04:= 0           // inicializar S04

for j ∈ 1 .. 5   // lazos "for " anidados
  for k ∈ 1 .. 5 // de la misma extension en i y k:
    S04:= S04+ k·j // 'range(1,5) '

S04= 225         // valor final de S04
-----
```

**Ejemplo de programacion usando estructuras secuenciales, de decision, y de lazo**

Este ejemplo demuestra el uso de las tres estructuras basicas de programacion en SMath Studio. Este es el algoritmo clasico para ordenar numeros en orden ascendente, conocido como "el algoritmo de burbujas." En este algoritmo, dado un vector the valores, el valor menor de todos (es decir, el mas ligero, el de menor peso) sube hacia la parte superior del vector como lo haria una burbuja en un liquido. El programa mostrado a continuacion usa un vector fila rS, y se refiere a sus elementos usando subindices numericos, por ejemplo, S[1,k], etc. El ejemplo muestra como escribir subindices numericos. El resultado del programa se guarda de vuelta en el vector rS.

```
-----
rS:=(5.4 1.2 3.5 10.2 -2.5 4.1) // Dado un vector "rS"
nS:= length(rS)    nS= 6      // Calculese la longitud del vector

for k ∈ 1 .. nS-1 // Lazo doble que efectua
  for j ∈ k+1 .. nS // el ordenamiento del vector rS
    if rS [1 k] > rS [1 j] // Para escribir subindices use,
      // por ejemplo, rS[1,k]
      temp:= rS [1 j]
      rS [1 j] := rS [1 k]
      rS [1 k] := temp
    else
      0

rS:=(-2.5 1.2 3.5 4.1 5.4 10.2) // Resultado: vector en orden ascendente
-----
```

Este ordenamiento puede efectuarse en SMath Studio usando la funcion "sort":

```
rT:=(5.4 1.2 3.5 10.2 -2.5 4.1)
```

$$\text{sort}(rT^T) = \begin{pmatrix} -2.5 \\ 1.2 \\ 3.5 \\ 4.1 \\ 5.4 \\ 10.2 \end{pmatrix}$$

Funcion para ordenar terminos usando el algoritmo de burbuja

El algoritmo del ordenamiento de burbuja se puede escribir como una funcion:

```

Burbuja(rS):=
nS:= length(rS)
for k∈ 1..nS-1
  for j∈ k+1..nS
    if rS_1 k > rS_1 j
      temp:= rS_1 j
      rS_1 j := rS_1 k
      rS_1 k := temp
    else
      0
rS

```

He aqui una aplicacion de esta funcion:

```

rS:=(5.4 1.2 3.5 10.2 -2.5 4.1)
Burbuja(rS)=(-2.5 1.2 3.5 4.1 5.4 10.2)

```

Los comandos "break" (salida abrupta) y "continue" (continuar proceso)

Estos comandos estan disponibles en la segunda linea del panel de Programacion.

\* El comando "break" provee una salida abrupta de un lazo "for", si se detecta una cierta condicion dentro del lazo, antes de realizar todas las repeticiones del lazo requeridas por la definicion del indice.

\* El comando "continue" basicamente deja el proceso de un programa pasar por el punto donde se encuentra este comando sin tomar accion alguna.

Para ilustrar el uso de los comandos "break" y "continue", considerese el programa escrito aqui:--->  
La variable XS se inicializa primero con el valor de cero, despues un comando "for" con indice k = 1,2, ...,10, activa un lazo, en el cual XS se incrementa por 1 en cada ciclo del lazo. En este programa, sin embargo, se ha incluido un comando "if" que saca el control del lazo en cuanto k > 5. El comando "if" automaticamente incluye una opcion "else", sin embargo, no requerimos ninguna accion en caso de que k<5 en este comando "if". Por lo tanto, colocamos un comando "continue" para esa opcion "else" en este programa.

-----  
XS:= 0

```

for k∈ 1..10
  XS:= XS+1
  if k>5
    break
  else
    continue

```

XS= 6  
-----

NOTA: Este ejemplo representa un lazo muy ineficiente, dado que el indice se define inicialmente con una extension de 1 a 10, pero el comando "if" efectivamente reduce la extension del indice a los numeros 1 al 4. El proposito de este ejemplo fue, por lo tanto, el de ilustrar el uso de los comandos "break" y "continue", solamente. Una forma mas eficiente de programar este calculo, sin el uso de "if", "break", o "continue", se muestra a continuacion:

-----  
XS:= 0

```

for k∈ 1..4
  XS:= XS+1

```

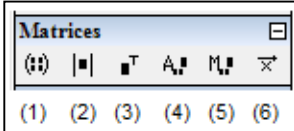
XS= 4  
-----

Muchos programas numericos requieren el uso de vectores y matrices, cuyo operacion en SMath Studio, se presenta a continuacion.

## Uso del panel de Matrices en SMath Studio + Operaciones matriciales

A continuacion se presentan ejemplos de creacion y manipulacion de matrices utilizando el panel de Matrices en SMath Studio, asi como otras funciones matriciales. La programacion usualmente requiere el uso de arreglos unidimensionales (vectores) y bidimensionales (matrices). Por lo tanto, es esencial conocer las funciones de vectores y matrices disponibles en SMath Studio para utilizarlas en la escritura de programas.

El panel de Matrices se muestra a continuacion:

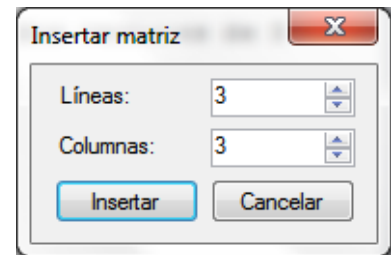


Este panel consiste de 6 iconos identificados de esta manera:

- (1) Matriz 3x3 (Cntl+M): escribir una matriz especificando el numero de lineas y columnas
- (2) Determinante: calcula el determinante de una matriz
- (3) Matriz transpuesta (Cntl+1): para obtener la transpuesta de una matriz
- (4) Cofactor de una matriz: similar al menor de una matriz
- (5) Menor: calcula el determinante de una matrix menor (o reducida)
- (6) Producto cruz (o producto vectorial): calcula el producto cruz de dos vectores de 3 elementos

### Escritura de matrices con el icono "Matriz 3x3"

Supongase que se quieren definir dos vectores columna de 3 elementos, u y v, usando el icono "Matriz 3x3". Primero definase el vector u al pulsar en una seccion vacia del cuaderno y escribir "u:". Despues, pulse el boton "Matriz 3x3" en el panel de Matrices para producir lo siguiente ----> en el cual, por defecto, se crea una matrix de 3 lineas y 3 columnas.



Cambiese el numero de "Columnas" a 1 y pulse el boton [Insertar]. Esto resulta en lo siguiente ----->

$$u := \begin{pmatrix} \blacksquare \\ \blacksquare \\ \blacksquare \end{pmatrix}$$

El siguiente paso consiste en escribir las componentes del vector al pulsar en los diferentes puntos de entrada y escribir los valores correspondientes. Los vectores u y v, escritos de esta manera, se muestran a la derecha: ----->

$$u := \begin{pmatrix} 3 \\ 5 \\ -2 \end{pmatrix} \quad v := \begin{pmatrix} 3 \\ -5 \\ 7 \end{pmatrix}$$

Utilizando el icono "Matriz 3x3", o el comando "Cntl+M", se pueden definir otras matrices tales como la matriz A, 5x4, y la matrix B, 6x6, que se muestran a continuacion:

$$A := \begin{pmatrix} 5 & -2 & 3 & 8 \\ -4 & -7 & 0 & 3 \\ 7 & 2 & 3 & -5 \\ -2 & -8 & 3 & 5 \\ 6 & 2 & -4 & 9 \end{pmatrix} \quad B := \begin{pmatrix} 2 & -8 & -3 & 5 & -6 & 3 \\ 3 & 1 & -5 & 4 & 4 & 2 \\ 6 & 8 & 6 & -1 & 8 & 8 \\ 9 & -3 & -5 & -6 & 8 & 7 \\ -3 & 8 & 0 & 9 & -7 & 8 \\ 6 & 0 & -7 & 4 & -1 & 2 \end{pmatrix}$$

### Calculo del determinante de una matriz

Para calcular el determinante de una matriz que ha sido definida anteriormente, como las matrices A y B, pulse en el cuaderno, y despues pulse el icono "Determinante" en el panel de Matrices. Escriba el nombre de la matriz en el punto de entrada del determinante y presione el signo igual (=) en su teclado. He aqui un ejemplo:

$$|B| = -6.1881 \cdot 10^5$$

Notese que el determinante de A, que no es una matriz cuadrada, no esta definido. Por lo tanto, el intentar calcular el determinante de A produce un error. ----->  $|A| = \blacksquare$

Transpuesta de una matriz

Para obtener la transpuesta de una matriz que ha sido definida anteriormente, pulse en el cuaderno, y presione el icono "Matriz transpuesta" en el panel de Matrices. Escriba el nombre de la matriz en el punto de entrada de la transpuesta y presione el signo igual (=) en su teclado. He aquí algunos ejemplos:

$$u^T = (3 \ 5 \ -2) \quad v^T = (3 \ -5 \ 7)$$


$$A^T = \begin{pmatrix} 5 & -4 & 7 & -2 & 6 \\ -2 & -7 & 2 & -8 & 2 \\ 3 & 0 & 3 & 3 & -4 \\ 8 & 3 & -5 & 5 & 9 \end{pmatrix} \quad B^T = \begin{pmatrix} 2 & 3 & 6 & 9 & -3 & 6 \\ -8 & 1 & 8 & -3 & 8 & 0 \\ -3 & -5 & 6 & -5 & 0 & -7 \\ 5 & 4 & -1 & -6 & 9 & 4 \\ -6 & 4 & 8 & 8 & -7 & -1 \\ 3 & 2 & 8 & 7 & 8 & 2 \end{pmatrix}$$

Iconos de Cofactor de una matriz (4) y Menor (5)

La operación de estos dos iconos es muy similar: Pulse en el cuaderno, presione el icono (4) o el icono (5), lo que produce ya sea el símbolo A o el símbolo M con dos puntos de entrada para subíndices y un punto de entrada entre parentesis. Escriba valores numericos enteros en los puntos de entrada de subíndices, digamos i y j. Asi mismo, escriba el nombre de una matriz cuadrada en el punto de entrada entre parentesis, digamos, B. El simbolo A produce el valor absoluto del determinante de la matriz menor que resulta al remover la linea i y la columna j de la matriz B. Por otro lado, el simbolo M produce el determinante de la misma matriz menor. Por ejemplo, para la matriz B, 6x6, definida anteriormente, resulta:

$$A_{2 \ 3}(B) = 22763 \quad M_{2 \ 3}(B) = -22763$$

Extracting una matriz menor (esta funcion no se encuentra en el panel de Matrices)

Para extraer una matriz menor utilicese la funcion "vminor()", disponible bajo la categoria "Matriz y Vector" en el icono de Funcion en la barra del ----->  menu en SMath Studio. Esta funcion produce un simbolo M muy similar al que se produce con el icono (5) en el panel de Matrices, pero resulta en la matriz menor, en lugar del determinante de esa matriz, es decir,

$$M_{2 \ 3}(B) = \begin{pmatrix} 2 & -8 & 5 & -6 & 3 \\ 6 & 8 & -1 & 8 & 8 \\ 9 & -3 & -6 & 8 & 7 \\ -3 & 8 & 9 & -7 & 8 \\ 6 & 0 & 4 & -1 & 2 \end{pmatrix}$$

Calculo de un producto cruz o producto vectorial

El producto cruz, o producto vectorial, opera solamente en vectores columna de tres elementos, tales como los vectores u y v definidos anteriormente.

Para calcular un producto cruz, pulse en un lugar vacio del cuaderno, y pulse el icono "Producto cruzado" en el panel de Matrices. Esta funcion produce dos puntos de entrada separados por el simbolo de multiplicacion (x). Escriba, en los puntos de entrada, los nombres de los vectores previamente definidos, o escriba nuevos vectores usando el icono "Matriz 3x3 (Cntl+M)" en el panel de Matrices. Finalmente, escriba el signo igual (=). He aquí algunos ejemplos:

$$u \times v = \begin{pmatrix} 25 \\ -27 \\ -30 \end{pmatrix} \quad v \times u = \begin{pmatrix} -25 \\ 27 \\ 30 \end{pmatrix}$$

Otras operaciones matriciales comunes (no contenidas en el panel de Matrices)

- 1 - Calculo de la INVERSA de una matriz cuadrada: escriba el nombre de la matriz y elevele a la potencia (-1), por ejemplo:

$$B^{-1} = \begin{pmatrix} 0.02 & -0.1 & 0.08 & -0.05 & -0.06 & 0.17 \\ -0.09 & -0.05 & 0 & -0.01 & 0.03 & 0.08 \\ 0.07 & -0.04 & 0.09 & -0.07 & -0.05 & -0.02 \\ 0.07 & 0.13 & 0.05 & -0.08 & -0.03 & -0.02 \\ 0.01 & 0.16 & 0.01 & 0 & -0.03 & -0.11 \\ 0.03 & 0.01 & -0.01 & 0.08 & 0.07 & -0.09 \end{pmatrix}$$

- 2 - Adicion, substracion, y producto de matrices: use los mismos operadores aritmeticos (+, -, \*) que se usan con cantidades escalares, por ejemplo:

$$B + B^T = \begin{pmatrix} 4 & -5 & 3 & 14 & -9 & 9 \\ -5 & 2 & 3 & 1 & 12 & 2 \\ 3 & 3 & 12 & -6 & 8 & 1 \\ 14 & 1 & -6 & -12 & 17 & 11 \\ -9 & 12 & 8 & 17 & -14 & 7 \\ 9 & 2 & 1 & 11 & 7 & 4 \end{pmatrix} \quad B^T - B = \begin{pmatrix} 0 & 11 & 9 & 4 & 3 & 3 \\ -11 & 0 & 13 & -7 & 4 & -2 \\ -9 & -13 & 0 & -4 & -8 & -15 \\ -4 & 7 & 4 & 0 & 1 & -3 \\ -3 & -4 & 8 & -1 & 0 & -9 \\ -3 & 2 & 15 & 3 & 9 & 0 \end{pmatrix}$$

$$B \cdot B^T = \begin{pmatrix} 147 & 15 & -99 & 0 & 41 & 65 \\ 15 & 71 & 40 & 71 & 23 & 69 \\ -99 & 40 & 265 & 126 & 45 & -2 \\ 0 & 71 & 126 & 264 & -105 & 71 \\ 41 & 23 & 45 & -105 & 267 & 41 \\ 65 & 69 & -2 & 71 & 41 & 106 \end{pmatrix} \quad B \cdot B^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$C := 3 \cdot B - 5 \cdot B^T \quad , \quad \text{or,}$$

$$C = \begin{pmatrix} -4 & -39 & -39 & -30 & -3 & -21 \\ 49 & -2 & -55 & 27 & -28 & 6 \\ 33 & 49 & -12 & 22 & 24 & 59 \\ 2 & -29 & -10 & 12 & -21 & 1 \\ 21 & 4 & -40 & -13 & 14 & 29 \\ 3 & -10 & -61 & -23 & -43 & -4 \end{pmatrix}$$

- 3 - Escritura de tipos especificos de matrices: use las siguientes funciones bajo la categoria "Matriz y Vector" que resulta al usar el icono de "Funcion":----->



diag(v): produce un matriz tal que los elementos de su diagonal principal son las componentes de un vector columna v, mientras que los elementos fuera de la diagonal principal son cero, por ejemplo,

$$\text{diag}(u) = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -2 \end{pmatrix} \quad \text{diag}(v) = \begin{pmatrix} 3 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & 7 \end{pmatrix}$$

identity(n): produce una matrix identidad de orden nxn, por ejemplo,

$$\text{identity}(3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{identity}(4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



`matrix(n,m)` : produce una matriz de n líneas y m columnas con todos los elementos iguales a cero, por ejemplo,

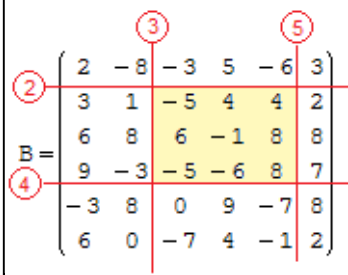
$$\text{matrix}(3, 4) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{matrix}(4, 4) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

`reverse(matriz)` : revierte el orden de las líneas en matrices o vectores, por ejemplo,

$$u = \begin{pmatrix} 3 \\ 5 \\ -2 \end{pmatrix} \quad \text{reverse}(u) = \begin{pmatrix} -2 \\ 5 \\ 3 \end{pmatrix}$$

$$B = \begin{pmatrix} 2 & -8 & -3 & 5 & -6 & 3 \\ 3 & 1 & -5 & 4 & 4 & 2 \\ 6 & 8 & 6 & -1 & 8 & 8 \\ 9 & -3 & -5 & -6 & 8 & 7 \\ -3 & 8 & 0 & 9 & -7 & 8 \\ 6 & 0 & -7 & 4 & -1 & 2 \end{pmatrix} \quad \text{reverse}(B) = \begin{pmatrix} 6 & 0 & -7 & 4 & -1 & 2 \\ -3 & 8 & 0 & 9 & -7 & 8 \\ 9 & -3 & -5 & -6 & 8 & 7 \\ 6 & 8 & 6 & -1 & 8 & 8 \\ 3 & 1 & -5 & 4 & 4 & 2 \\ 2 & -8 & -3 & 5 & -6 & 3 \end{pmatrix}$$

`submatrix(matriz, is, ie, js, je)` : extrae una submatrix de "matriz" incluyendo líneas is a ie, and columnas js a je, por ejemplo,

$$\text{submatrix}(B, 2, 4, 3, 5) = \begin{pmatrix} -5 & 4 & 4 \\ 6 & -1 & 8 \\ -5 & -6 & 8 \end{pmatrix} \quad , \text{i.e.},$$


`col(matriz, j)` : extrae la columna j de "matriz" como un vector columna, por ejemplo,

$$\text{col}(A, 3) = \begin{pmatrix} 3 \\ 0 \\ 3 \\ 3 \\ -4 \end{pmatrix} \quad \text{col}(B, 2) = \begin{pmatrix} -8 \\ 1 \\ 8 \\ -3 \\ 8 \\ 0 \end{pmatrix}$$

`row(matriz, i)` : extrae la línea i de "matriz" como un vector línea, por ejemplo,

$$\text{row}(A, 2) = (-4 \ -7 \ 0 \ 3) \quad \text{row}(B, 3) = (6 \ 8 \ 6 \ -1 \ 8 \ 8)$$

`augment(m1,m2)` : produce una nueva matriz al agregar las columnas de la matriz m2 a la matriz m1. Las matrices m1 y m2 deben tener el mismo número de líneas, por ejemplo,

$$u = \begin{pmatrix} 3 \\ 5 \\ -2 \end{pmatrix} \quad v = \begin{pmatrix} 3 \\ -5 \\ 7 \end{pmatrix} \quad \text{augment}(u, v) = \begin{pmatrix} 3 & 3 \\ 5 & -5 \\ -2 & 7 \end{pmatrix}$$

`stack(m1,m2)`: produce una nueva matriz al agregar las líneas de la matriz `m2` a la matriz `m1`. Las matrices `m1` y `m2` deben tener el mismo número de columnas, por ejemplo,

$$u^T = (3 \ 5 \ -2) \quad v^T = (3 \ -5 \ 7) \quad \text{stack}(u^T, v^T) = \begin{pmatrix} 3 & 5 & -2 \\ 3 & -5 & 7 \end{pmatrix}$$

- 4 - Funciones que caracterizan matrices: estas son funciones que producen cantidades que representan alguna característica de una matriz. Una de esas cantidades es el determinante de una matriz, el cual puede calcularse usando el icono (2) en el panel de Matrices, o usando la función "det".

Las siguientes son funciones de interés disponibles bajo la categoría "Matriz y Vector" en el icono de "Función" en la barra de menú de SMath Studio:



```
* cols(matriz): determina el número de columnas en una "matriz"
* det(matriz): calcula el determinante de una "matriz"
* el(matriz,i,j): extrae elemento i,j de una "matriz"
* length(matriz): determina el número de elementos en una "matriz"
  (o vector)
* max(matriz): determina el valor máximo en una "matriz" (o vector)
* min(matriz): determina el valor mínimo en una "matriz" (o vector)
* norm1(matriz): determina la norma L1 de una "matriz"
* norme(matriz): determina la norma Euclidiana de una "matriz"
* normi(matriz): determina la norma infinita de una "matriz"
* rank(matriz): determina el rango de una "matriz"
* rows(matriz): determina el número de líneas en una "matriz"
* trace(matriz): determina la traza (suma de elementos en la diagonal)
  de una "matrix" cuadrada
```

Algunos ejemplos de estas funciones se muestran a continuación:

```
cols(A)=4      rows(A)=5      B2 3 = -5  <-- this is "el(B,2,3)"

length(B)=36   max(B)=9      min(B)=-8

norm1(A)=30    norme(B)=■     normi(A)=21

rank(A)=4      tr(B)=-2
```

- 5 - Operaciones que involucran ordenar una matriz por columnas o líneas, u ordenar un vector: Estas operaciones están también disponibles bajo la categoría "Matriz y Vector" del icono de Función en la barra del menú de SMath Studio: ----->



`csort(matrix,j)`: ordena los elementos de la columna `j` en orden ascendente mientras llevan con ellos los elementos correspondientes de otras columnas, por ejemplo, ordenar la matriz `B` por la columna 3:

$$B = \begin{pmatrix} 2 & -8 & -3 & 5 & -6 & 3 \\ 3 & 1 & -5 & 4 & 4 & 2 \\ 6 & 8 & 6 & -1 & 8 & 8 \\ 9 & -3 & -5 & -6 & 8 & 7 \\ -3 & 8 & 0 & 9 & -7 & 8 \\ 6 & 0 & -7 & 4 & -1 & 2 \end{pmatrix} \quad \text{csort}(B, 3) = \begin{pmatrix} 6 & 0 & -7 & 4 & -1 & 2 \\ 3 & 1 & -5 & 4 & 4 & 2 \\ 9 & -3 & -5 & -6 & 8 & 7 \\ 2 & -8 & -3 & 5 & -6 & 3 \\ -3 & 8 & 0 & 9 & -7 & 8 \\ 6 & 8 & 6 & -1 & 8 & 8 \end{pmatrix}$$

`rsort(matrix, j)`: ordena los elementos de la línea  $i$  en orden ascendente mientras llevan con ellos los elementos correspondientes de otras líneas, por ejemplo, ordenar la matriz  $B$  por la columna 3:

$$B = \begin{pmatrix} 2 & -8 & -3 & 5 & -6 & 3 \\ 3 & 1 & -5 & 4 & 4 & 2 \\ 6 & 8 & 6 & -1 & 8 & 8 \\ 9 & -3 & -5 & -6 & 8 & 7 \\ -3 & 8 & 0 & 9 & -7 & 8 \\ 6 & 0 & -7 & 4 & -1 & 2 \end{pmatrix} \quad \text{rsort}(B, 4) = \begin{pmatrix} 5 & -3 & -8 & 3 & -6 & 2 \\ 4 & -5 & 1 & 2 & 4 & 3 \\ -1 & 6 & 8 & 8 & 8 & 6 \\ -6 & -5 & -3 & 7 & 8 & 9 \\ 9 & 0 & 8 & 8 & -7 & -3 \\ 4 & -7 & 0 & 2 & -1 & 6 \end{pmatrix}$$

`sort(vector)`: ordena los elementos de un vector columna en orden ascendente, por ejemplo,

$$v = \begin{pmatrix} 3 \\ -5 \\ 7 \end{pmatrix} \quad \text{sort}(v) = \begin{pmatrix} -5 \\ 3 \\ 7 \end{pmatrix} \quad u = \begin{pmatrix} 3 \\ 5 \\ -2 \end{pmatrix} \quad \text{sort}(u) = \begin{pmatrix} -2 \\ 3 \\ 5 \end{pmatrix}$$

### Pasos típicos en la programación

Los siguientes pasos se recomiendan en la preparación de programas eficientes:

- (1) Defina claramente el problema a resolverse
- (2) Defina los valores de entrada y salida del programa
- (3) Diseñe el algoritmo utilizando diagramas de flujo o pseudo-código
- (4) Programe el algoritmo en un lenguaje de programación (por ejemplo, comandos de programación en SMath Studio)
- (5) Pruebe el código que resulta con un conjunto de valores ya conocidos

### Errores en la programación

Típicamente existen tres tipos principales de errores que aparecen al programar:

- (1) Errores de sintaxis: ocurre cuando un comando no sigue la sintaxis del lenguaje de programación. Estos errores son fáciles de detectar pues el lenguaje mismo avisa al programador de las violaciones a la sintaxis.
- (2) Errores de ejecución: Estos son errores que típicamente involucran inconsistencias matemáticas, por ejemplo, una división por cero. Estos pueden ser detectados por el usuario cuando se corre un programa.
- (3) Errores lógicos: Estos son errores del algoritmo mismo. Estos errores son, a veces, difíciles de detectar, por eso la necesidad de probar programas con valores conocidos. Verifique cada paso de un algoritmo para asegurarse que el programa está operando como se diseñó originalmente.

### EJEMPLO DE PROGRAMACION No. 1 - El método de Newton-Raphson para resolver $f(x) = 0$

El método de Newton-Raphson que se utiliza para resolver una ecuación de la forma  $f(x) = 0$  requiere que se conozca la derivada  $f'(x)$ . Esta última puede conseguirse fácilmente en SMath Studio usando la opción "Derivada" (5to ícono, segunda línea) en el panel de Funciones:

$$fp(x) = \frac{d}{dx} f(x)$$

Dado un estimado inicial de la solución,  $x = x_0$ , la solución puede ser aproximada por el cálculo iterativo:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

para  $k = 0, 1, \dots$

Las iteraciones continúan hasta que la solución converja, es decir,  $|f(x_{k+1})| < \varepsilon$ , o hasta que se alcance un número máximo de iteraciones sin conseguir convergencia, es decir,  $k > n_{\max}$

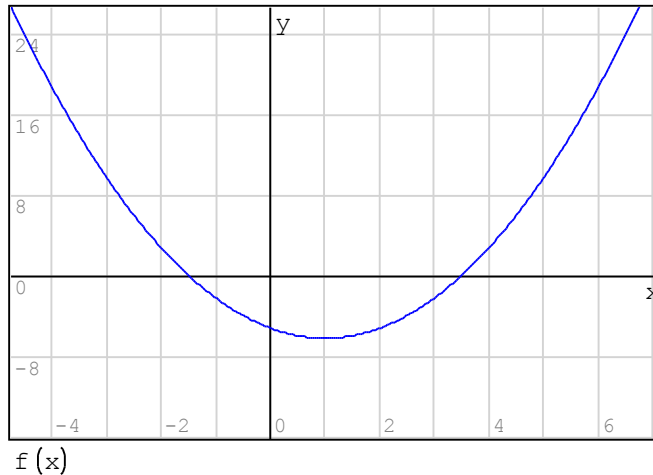
Ejemplo: Resuelva la ecuación:  $x^2 - 2 \cdot x - 5 = 0$

Solución: La gráfica de la función nos da una idea sobre donde se encuentran las soluciones:

Defina la función:

$$f(x) := x^2 - 2 \cdot x - 5$$

Produzca la gráfica de  $f(x)$ :



La gráfica muestra soluciones cerca de  $x = -1$  y  $x = 3$ . Podemos implementar la solución usando el método de Newton-Raphson como se indica a continuación:

$$f_p(x) := \frac{d}{dx} f(x) \quad f_p(x) = -10$$

Los parámetros de la solución son:  $\varepsilon := 1.0 \cdot 10^{-6}$   $n_{\max} := 100$

Cálculo de una solución:

Comenzando con un estimado inicial:  $x_G := -2.5$  se encuentra una solución usando el procedimiento iterativo:

```

k:= 0
while ((k ≤ nmax) ∧ (|f(xG)| > ε))
  xGp1:= xG - f(xG)/fp(xG)
  k:= k + 1
  xG:= xGp1

```

$x_G = -1.45$

Esta es la solución

$k = 4$

Después de este número de iteraciones

$f(x_G) = 2.14 \cdot 10^{-11}$

La función en el valor de la solución

Escritura del programa de Newton-Raphson

Los pasos indicados anteriormente para calcular la solución de la ecuación  $f(x) = 0$  se pueden agrupar en una función como se indica a continuación:

$$f_{\text{Newton}}(f, x_0, \varepsilon, N_{\text{max}}) := \begin{array}{l} f_p(x) := \frac{d}{dx} f(x) \\ x_G := x_0 \\ k := 0 \\ \text{while } ((k \leq n_{\text{max}}) \wedge (|f(x_G)| > \varepsilon)) \\ \quad \left| \begin{array}{l} x_{Gp1} := x_G - \frac{f(x_G)}{f_p(x_G)} \\ k := k + 1 \\ x_G := x_{Gp1} \end{array} \right. \\ \left( \begin{array}{c} x_G \\ f(x_G) \\ k \end{array} \right) \end{array}$$

NOTA: El resultado se da en la forma de un vector columna de 3 elementos: (1) la solución,  $x_G$ , (2) la función en el valor de la solución,  $f(x_G)$ , y (3) el número de iteraciones,  $k$ , requeridas para converger a una solución.

Solución para un caso particular:

$$ff(x) := x^2 - 2 \cdot x - 5 \quad \varepsilon := 1.0 \cdot 10^{-6} \quad N_{\text{max}} := 100$$

Solución 1:

$$x_0 := 2.5 \quad f_{\text{Newton}}(ff(x), x_0, \varepsilon, N_{\text{max}}) = \begin{pmatrix} 3.45 \\ 2.99 \cdot 10^{-9} \\ 4 \end{pmatrix}$$

Solución 2:

$$x_0 := -2.2 \quad f_{\text{Newton}}(ff(x), x_0, \varepsilon, N_{\text{max}}) = \begin{pmatrix} 3.45 \\ 2.99 \cdot 10^{-9} \\ 4 \end{pmatrix}$$

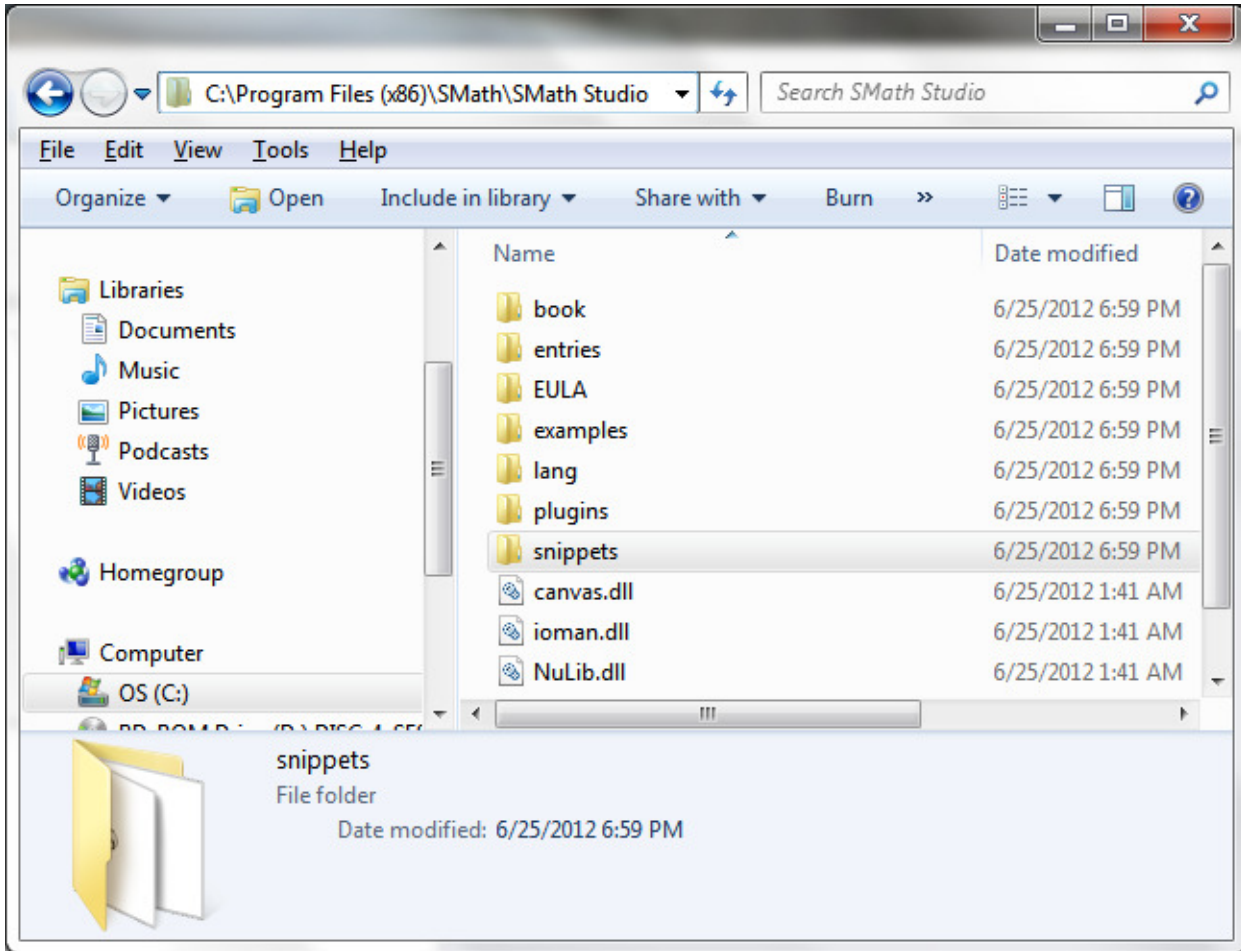
Compare with solution given by function "solve":

$$\text{solve}(ff(x) = 0, x, -5, 0) = -1.45$$

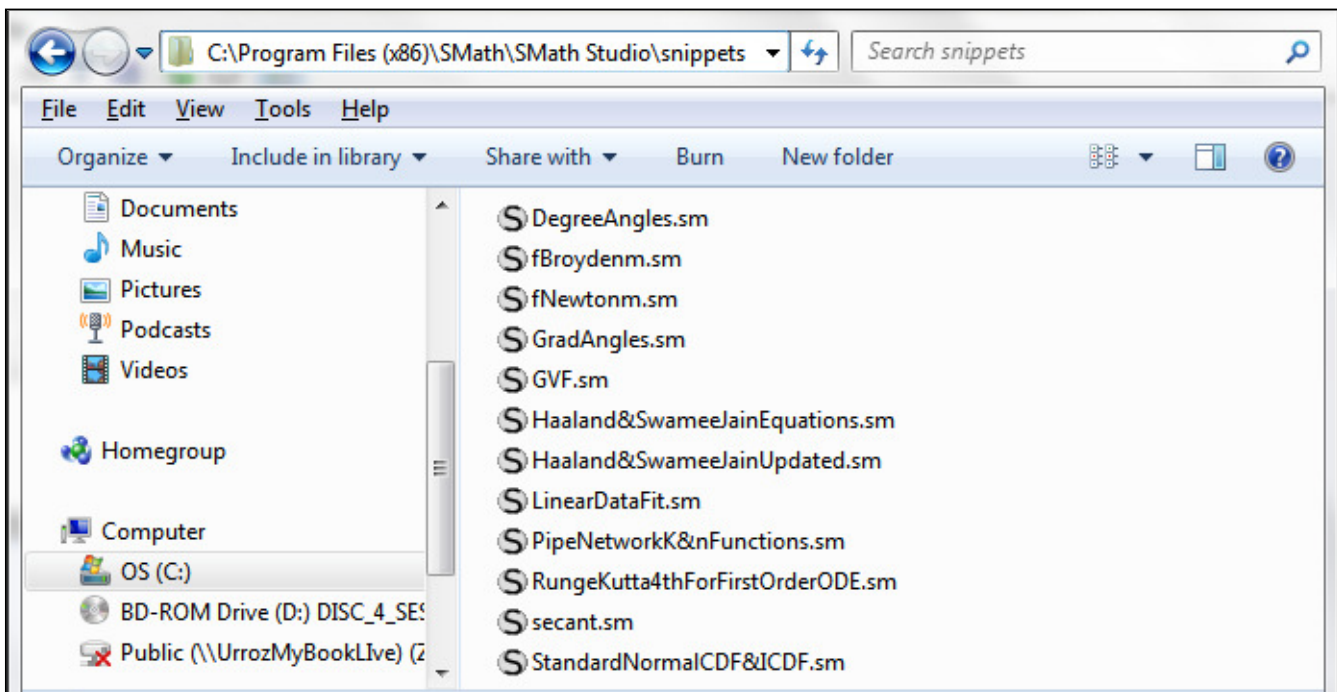
$$\text{solve}(ff(x) = 0, x, 0, 5) = 3.45$$

## Uso de los Fragmentos de Código

Un Fragmento de Código es una función definida por el usuario en un archivo de SMath Studio que se coloca en la carpeta "snippets" en la instalación de SMath Studio en su computadora (u ordenador). Por ejemplo, la figura siguiente muestra la localización de la carpeta de fragmentos "snippets" en una instalación en el sistema operativo Windows 7.



En mi computadora, por ejemplo, tengo los siguientes fragmentos de código que uso para mis cursos:



La siguiente figura muestra un archivo de SMath Studio titulado "StandardNormalCDF&ICDF.sm" que muestra programas utilizados para estimar la Funcion de Distribucion Cumulativa (CDF) y la Funcion de Distribucion Cumulativa Inversa (ICDF) de la distribucion normal estandar de probabilidades. Este archivo es un ejemplo de un Fragmento de Codigo que define 2 funciones:  $\Phi(z)$ , la CDF de la distribucion normal estandar, y  $\Phi^{-1}(p)$ , la ICDF de la distribucion normal estandar.

The screenshot shows the SMath Studio interface with a window titled "StandardNormalCDF&ICDF.sm". The code defines two functions:  $\Phi(z)$  and  $\Phi^{-1}(p)$ . The code is as follows:

```

Standard normal CDF  $\Phi(z)$  and its inverse  $\Phi^{-1}(p)$ :
Function  $\Phi(z)$  and its inverse  $\Phi^{-1}(p) = \Phi^{-1}(p)$ ,  $0 < p < 1$ , are
defined in the following SMath Studio programs:

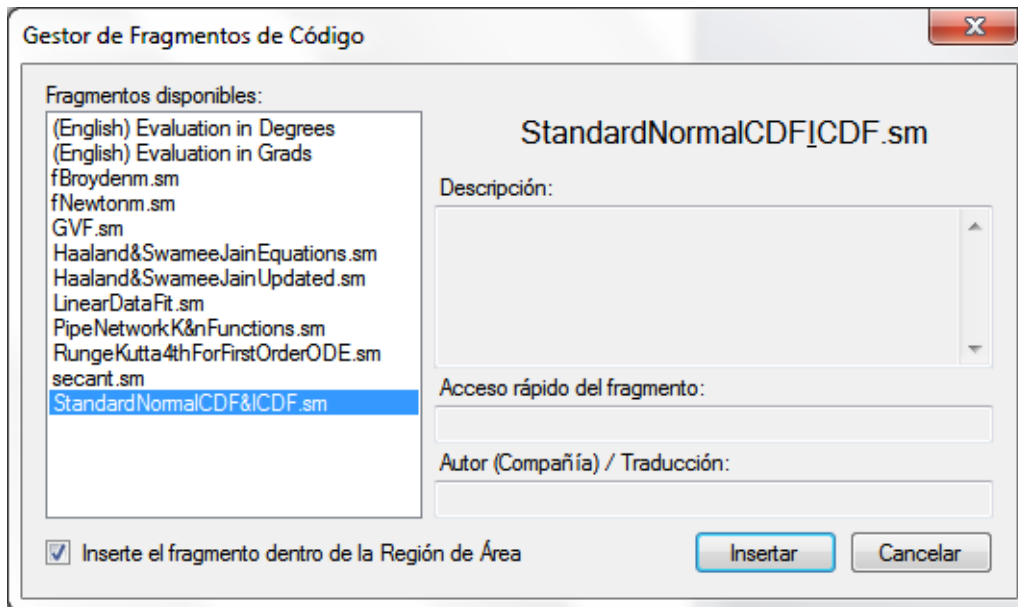
 $\Phi(z) :=$ 
s := |z|
b :=
  ( 0.2316419
    0.319381530
   -0.356563782
    1.781477937
   -1.821255978
    1.330274429 )
r :=  $\frac{1}{\sqrt{2 \cdot \pi}} \cdot \exp\left(-\frac{s^2}{2}\right)$ 
t :=  $\frac{1}{1 + b_1 \cdot s}$ 
w :=  $1 - r \cdot \sum_{k=1}^5 (b_{k+1} \cdot t^k)$ 
if z ≥ 0
  w
else
  1 - w

 $\Phi^{-1}(p) :=$ 
F(x) :=
  ( 0.2316419
    0.319381530
   -0.356563782
    1.781477937
   -1.821255978
    1.330274429 )
b :=
r :=  $\frac{1}{\sqrt{2 \cdot \pi}} \cdot \exp\left(-\frac{x^2}{2}\right)$ 
t :=  $\frac{1}{1 + b_1 \cdot x}$ 
 $1 - r \cdot \sum_{k=1}^5 (b_{k+1} \cdot t^k)$ 
if p < 0.5
  q := 1 - p
else
  q := p
s := solve(F(z) = q, z, -4, 4)
if p < 0.5
  -s
else
  s
  
```

S


Los Fragmentos de Codigo pueden introducirse en un archivo de SMath Studio pulsando en un sitio vacio del cuaderno, y despues usando la opcion: "Servicio > Gestor de Fragmentos de Codigo" en la barra del menu de SMath Studio.

De esta manera se abre una lista de todos los Fragmentos de Codigo disponible en la computadora. En mi computadora, por ejemplo, obtengo la siguiente lista:



Una vez seleccionado el Fragmento de Código que se quiera introducir, presione el botón [ Insertar ].

Por ejemplo, a continuación se introduce el Fragmento de Código "StandardNormalCDF&ICDF.sm":

 StandardNormalCDF&ICDF.sm

El resultado es un área del cuaderno colapsada a la que se le asigna el nombre del Fragmento de Código introducido. En este caso el nombre del área colapsada es "StandardNormalCDF&ICDF.sm". El símbolo [+] localizado a la izquierda del nombre del Fragmento de Código indica que el área que contiene el Fragmento de Código está cerrada. Si Ud. pulsa sobre el símbolo [+], el área colapsada se abre mostrando el contenido de los Fragmentos de Código.

Después de haber introducido un Fragmento de Código en un cuaderno se pueden utilizar las funciones definidas en el Fragmento de Código para cálculos en cualquier localidad del cuaderno debajo del punto de inserción del Fragmento. Por ejemplo, los siguientes son cálculos realizados con las funciones  $\Phi(z)$  y  $\Phi_{inv}(p)$ :

$$\Phi(1.2) = 0.8849$$

$$\Phi(-1.2) = 0.1151$$

$$\Phi(1.2) + \Phi(-1.2) = 1$$

$$\Phi_{inv}(0.6) = 0.2533$$

$$\Phi_{inv}(0.2) = -0.8416$$

$$\Phi_{inv}(0.86) = 1.0803$$



**EJEMPLO DE PROGRAMACION No.2 - Metodo de Newton-Raphson generalizado a un sistema de ecuaciones**

Dado el sistema de ecuaciones:  $f_1(x) = 0$   
 $f_2(x) = 0$  con  $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$   
 $\dots$   
 $f_n(x) = 0$

Establecemos la funcion vectorial:  $f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{pmatrix}$

Sea  $x_0 = \begin{pmatrix} x_{01} \\ x_{02} \\ \vdots \\ x_{0n} \end{pmatrix}$  un estimado inicial de la solucion buscada.

El metodo de Newton-Raphson generalizado sugiere que se puede calcular una mejor aproximacion de la solucion usando:

$$x_{k+1} = x_k - J^{-1} \cdot f(x_k), \quad k = 0, 1, 2, \dots$$

en la cual J es el Jacobiano de la funcion vectorial:  $J_{ij} = \frac{d}{dx_j} f_i$

El Jacobiano puede calcularse usando la funcion "Jacob" en SMath Studio.

Para verificar la convergencia usamos el criterio:

$$\text{norme}(f(x_{k+1})) \leq \varepsilon$$

en la que la funcion "norme" calcula la norma Euclidiana de la funcion.

Asi mismo verificamos para divergencia al mantener el numero de iteraciones a un valor maximo "Nmax".

La siguiente funcion, llamada "fNewtonm", calcula el metodo de Newton-Raphson multivariable:

```
fNewtonm(f(x), xs, x0, ε, Nmax) :=
  fJ(x) := Jacob(f(x), xs)
  k := 0
  xG := x0
  while ((k ≤ Nmax) ∧ (norme(f(xG)) > ε))
    JJ := eval(fJ(xG))
    JJI := eval(invert(JJ))
    fxG := eval(f(xG))
    DxG := eval(JJI · fxG)
    xGp1 := eval(xG - DxG)
    k := k + 1
    xG := xGp1
  (xG)
  (k)
```

En este código,  $f(x)$  es una función vectorial cuyos elementos corresponden a las diferentes ecuaciones no lineales que se quieren resolver, y  $x_s$  es un vector columna que lista las variables involucradas en las ecuaciones. La función "fNewtonm" utiliza la función "Jacob", la cual calcula la matriz Jacobiana de la función  $f(x)$ .

$x_0$  es un estimado inicial de la solución,  $\epsilon$  es la tolerancia de convergencia, y "Nmax" es el máximo número de iteraciones permitido antes de cancelar una solución debido a divergencia.

La solución es un vector: 
$$\begin{pmatrix} xG \\ k \end{pmatrix} = \begin{pmatrix} \text{solucion} \\ \text{Numero\_de\_iteraciones} \end{pmatrix}$$

EJEMPLO de aplicación de la función "fNewtonm" a un sistema de ecuaciones no lineales

Resolvemos el siguiente sistema de ecuaciones:

$$\begin{aligned} 280 - HJ &= 0.0603 \cdot Q1 \cdot |Q1| \\ 290 - HJ &= 0.0203 \cdot Q2 \cdot |Q2| \\ 150 - HJ &= 0.0543 \cdot Q3 \cdot |Q3| \\ Q1 + Q2 + Q3 - 80 &= 0 \end{aligned}$$

En el cual cambiamos variables a:  $x_1 = Q1$   $x_2 = Q2$   $x_3 = Q3$   $x_4 = HJ$

Re-escribimos las ecuaciones para producir la siguiente función vectorial en forma de columna,  $fK$ , y el vector que lista las variables,  $xK$ :

$$fK(x) := \begin{pmatrix} 280 - x_4 - 0.0603 \cdot x_1 \cdot |x_1| \\ 290 - x_4 - 0.0203 \cdot x_2 \cdot |x_2| \\ 150 - x_4 - 0.0543 \cdot x_3 \cdot |x_3| \\ x_1 + x_2 + x_3 - 80 \end{pmatrix} \quad xK := \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Usando los siguientes valores de  $\epsilon$  y Nmax:  $\epsilon := 1 \cdot 10^{-10}$  Nmax := 20

y el estimado inicial:  $x_0 := \begin{pmatrix} 0.5 \\ 0.4 \\ 0.3 \\ 200 \end{pmatrix}$  se encuentra la siguiente solución:

$XSol := fNewtonm(fK(x), xK, x_0, \epsilon, Nmax)$

Extrayendo las soluciones para  $x$ :

$$XSol = \begin{pmatrix} 0.5 \\ 0.4 \\ 0.3 \\ 200 \\ 0 \end{pmatrix}$$

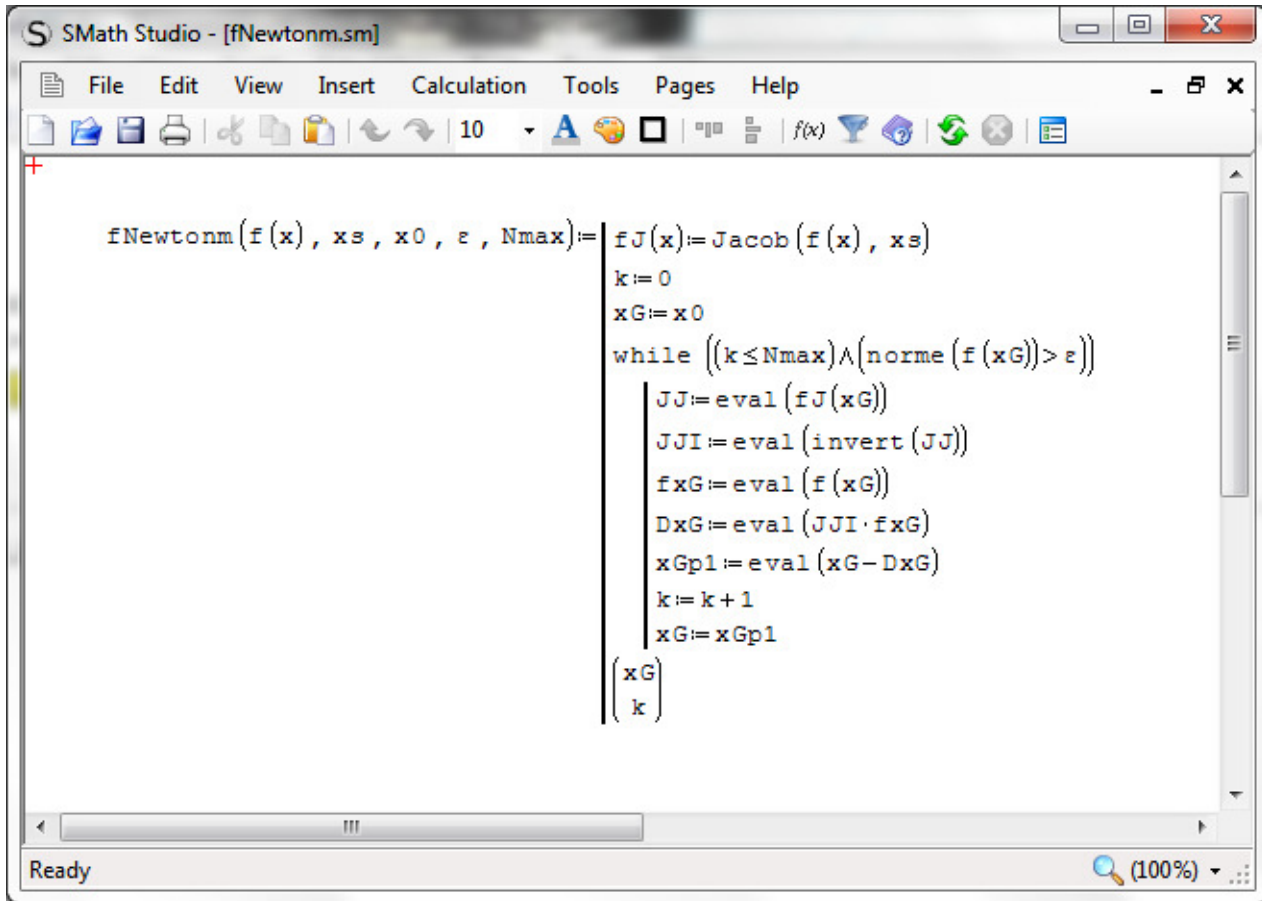
$xsol := XSol_1$

$$xsol = \begin{pmatrix} 0.5 \\ 0.4 \\ 0.3 \\ 200 \end{pmatrix}$$

$$fK(xsol) = \begin{pmatrix} 79.98 \\ 90 \\ -50 \\ -78.8 \end{pmatrix}$$

### Fragmento de código para la función "fNewtonm"

El fragmento de código llamado "fNewton.sm", que se muestra a continuación, contiene la función "fNewton", que se definió anteriormente:



```

SMATH Studio - [fNewtonm.sm]
File Edit View Insert Calculation Tools Pages Help
fNewtonm(f(x), xS, x0, ε, Nmax) :=
  fJ(x) := Jacob(f(x), xS)
  k := 0
  xG := x0
  while ((k ≤ Nmax) ∧ (norme(f(xG)) > ε))
    JJ := eval(fJ(xG))
    JJI := eval(invert(JJ))
    fxG := eval(f(xG))
    DxG := eval(JJI · fxG)
    xGp1 := eval(xG - DxG)
    k := k + 1
    xG := xGp1
  (xG)
  (k)
  
```

Este Fragmento de Código está disponible en mi carpeta "snippets".

A continuación, se introduce el segmento de código "fNewtonm.sm" y repetimos la solución del sistema de ecuaciones no lineales que se propuso anteriormente:

fNewtonm.sm

```
XSol := fNewtonm(fK(x), xK, x0, ε, Nmax)
```

Extrayendo las soluciones para x:

$$XSol = \begin{pmatrix} 0.5 \\ 0.4 \\ 0.3 \\ 200 \\ 0 \end{pmatrix}$$

xsol := XSol<sub>1</sub>

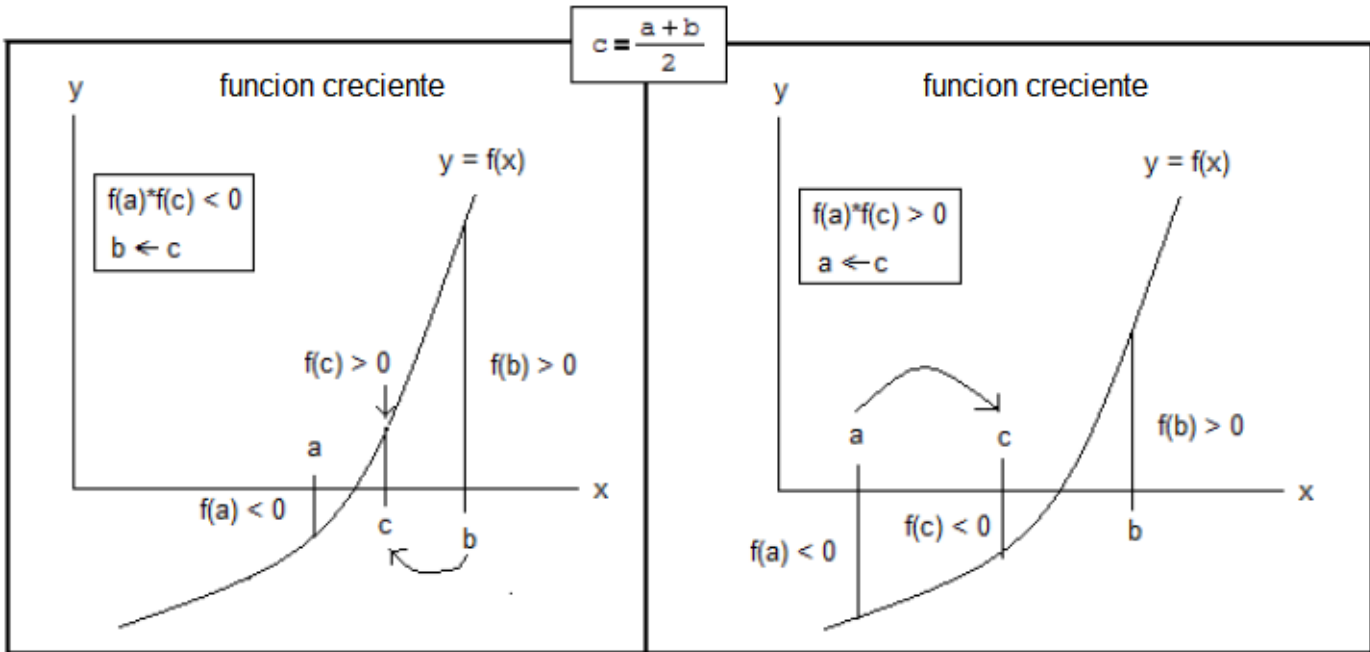
$$xsol = \begin{pmatrix} 0.5 \\ 0.4 \\ 0.3 \\ 200 \end{pmatrix}$$

$$fK(xsol) = \begin{pmatrix} 79.98 \\ 90 \\ -50 \\ -78.8 \end{pmatrix}$$

La solución se encuentra en el vector "xsol" que se muestra anteriormente. Las definiciones de fK(x), xK, x0, ε, y Nmax se dieron anteriormente.

**EJEMPLO DE PROGRAMACION No. 3 - El metodo de la biseccion para resolver ecuaciones**

La solucion de la ecuacion  $f(x) = 0$  es el punto donde la grafica de la funcion  $y = f(x)$  cruza el eje de las  $x$ . La figura que se muestra a continuacion ilustra tal situacion para el caso de una funcion que es creciente cerca de la solucion. Para encerrar la solucion en un cierto intervalo, primero identificamos un par de valores  $a$  y  $b$ , tales que  $a < b$  y, en este caso,  $f(a) < 0$  y  $f(b) > 0$ . En tal situacion sabemos que la solucion,  $x$ , se encuentra entre  $a$  y  $b$ , es decir,  $a < x < b$ .



Como una primera aproximacion a la solucion calculamos el punto medio del intervalo  $\{a, b\}$ , es decir,  $c = (a+b)/2$ . La figura presentada anteriormente sugiere dos posibilidades:

- (i)  $f(c) > 0$ , en cuyo caso podemos encerrar la solucion en un intervalo mas pequeno al tomar  $b = c$ , y calcular un nuevo valor de  $c = (a+b)/2$ .
- (ii)  $f(c) < 0$ , en cuyo caso podemos encerrar la solucion en un intervalo mas pequeno al tomar  $a = c$ , y calcular un nuevo valor de  $c = (a+b)/2$ .

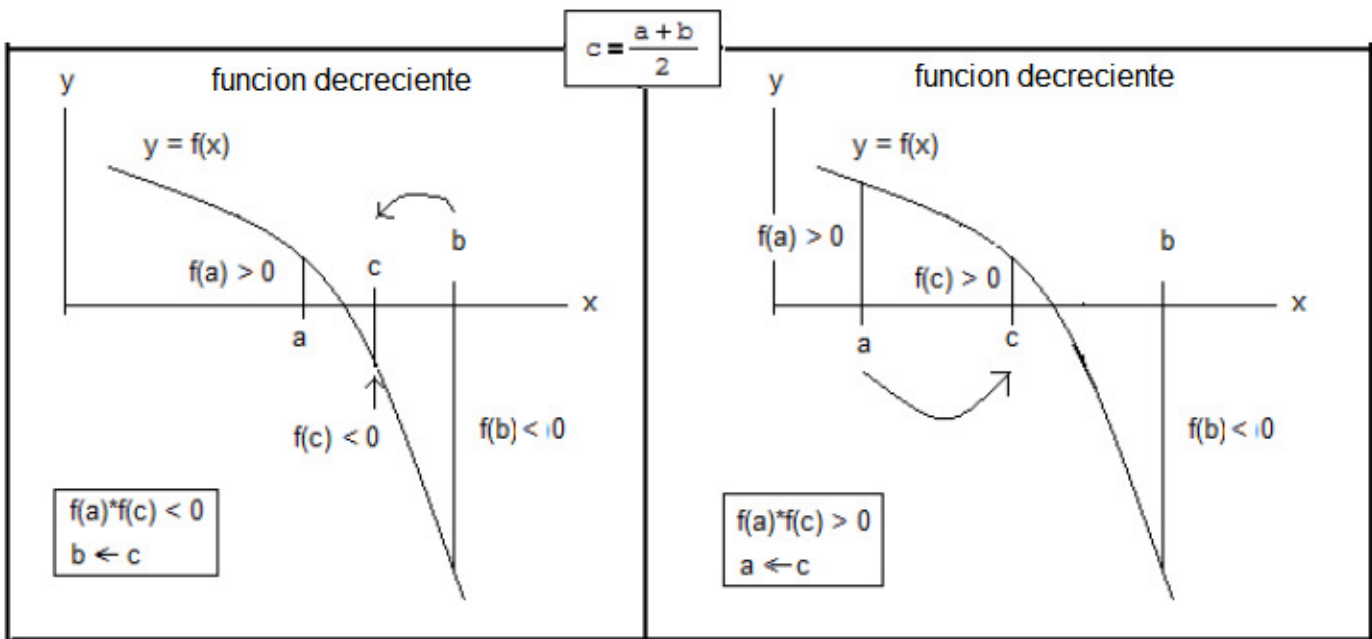
Cuando se repite el proceso como se indico' anteriormente se puede llegar a encontrar un valor de  $c$  para el cual  $|f(c)| < \epsilon$ , en donde  $\epsilon$  es un numero muy pequeno (es decir,  $f(c)$  llega a estar tan cerca de cero como se requiera). Alternativamente, podemos parar el proceso cuando el numero de iteraciones, digamos,  $k$ , excede un valor dado, digamos,  $N_{max}$ .

El proceso descrito anteriormente es un primer boceto de un algoritmo para el llamado metodo de la Biseccion para resolver una sola ecuacion de la forma  $f(x) = 0$ , para el caso en que  $y = f(x)$  es una funcion creciente.

Para el caso de una funcion decreciente, la figura a continuacion indica que una solucion estara' encerrada por los valores  $a$  y  $b$ , con  $a < b$ , si  $f(a) > 0$  y  $f(b) < 0$ . El punto medio del intervalo  $[a, b]$  se calcula como se hizo anteriormente, es decir, con  $c = (a+b)/2$ . Para este caso, la figura sugiere tambien dos posibilidades.

- (i)  $f(c) < 0$ , en este caso podemos encerrar la solucion en un intervalo mas pequeno al tomar  $b = c$ , y calcular un nuevo valor de  $c = (a+b)/2$ .
- (ii)  $f(c) > 0$ , en este caso podemos encerrar la solucion en un intervalo mas pequeno al tomar  $a = c$ , y calcular un nuevo valor de  $c = (a+b)/2$ .

Notese que, para ambos casos, la solucion se encuentra en el intervalo  $[a, b]$  si  $f(a) \cdot f(b) < 0$ , es decir, siempre y cuando  $f(a)$  y  $f(b)$  tengan signos opuestos. El valor de  $c = (a+b)/2$ , pasa a reemplazar el valor de  $a$  o el valor de  $b$  dependiendo de si  $f(a) \cdot f(c) > 0$  or si  $f(a) \cdot f(c) < 0$ . Se calcula entonces un nuevo valor de  $c = (a+b)/2$ , y el proceso continua hasta que  $|f(c)| < \epsilon$ , o hasta que  $k > N_{max}$ .



La figura siguiente resume el algoritmo del metodo de la biseccion para resolver ecuaciones de la forma  $f(x) = 0$  basado en la informacion obtenida anteriormente para funciones crecientes y decrecientes.

```

El metodo de la biseccion para resolver  $f(x) = 0$ 

* Comenzar con valores de a y b tal que  $f(a)*f(b) < 0$ 
*  $k = 0$ 
* Calculese  $c = (a+b)/2$ 
*  $k = k + 1$ 
* Si  $k < nmax$ , continuar, sino parar y reportar
* Si  $|f(c)| < \epsilon$ , proceso converge, reportar solucion
* Si  $f(a)*f(x) < 0$  entonces
    tomar  $b = c$ 
  Si no
    tomar  $a = c$ 

```

Una implementacion de este algoritmo se muestra a continuacion. En este programa se usan "cadenas" de caracteres -- es decir, texto -- para reportar resultados que resultan en la falta de solucion de las ecuaciones.

- (i) Si la solucion no esta' dentro del intervalo  $[a,b]$ , los estimados iniciales, a y b, son incorrectos, por lo tanto, el programa indica que el producto " $f(a)*f(b)$  debe ser  $< 0$ "
- (ii) Si no hay convergencia despues de  $Nmax$  iteraciones, el programa reporta "no convergencia"

```

Biseccion(f, a, b, ε, Nmax):= if f(a)·f(b)> 0
    "f(a)*f(b) debe ser < 0"
else
    k:= 0
    c:=  $\frac{a+b}{2}$ 
    while (k<Nmax)^(|f(c)|>ε)
        c:=  $\frac{a+b}{2}$ 
        k:= k+1
        if f(a)·f(c)< 0
            b:= c
        else
            a:= c
    if k>Nmax
        "no convergencia"
    else
        c

```

A continuacion probamos el programa para la funcion:  $f(x) := x^3 - 4 \cdot x^2 - 27 \cdot x + 85$

con parametros:  $\epsilon = 10^{-5}$   $N_{max} = 10$

Varias selecciones de valores de a y b se muestran a continuacion:

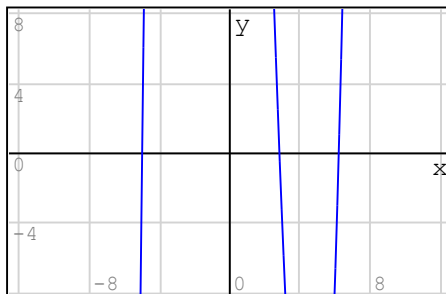
a:=- 8    b:=- 5    Biseccion(f, a, b, ε, Nmax)= "f(a)\*f(b) debe ser < 0"

a:=- 8    b:=- 2    Biseccion(f, a, b, ε, Nmax)=- 4.9473

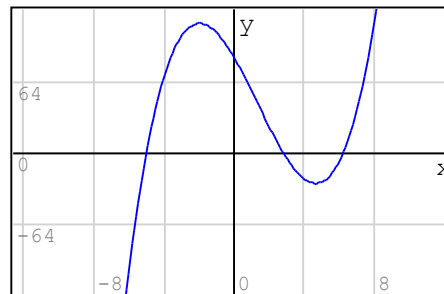
a:= 1    b:= 4    Biseccion(f, a, b, ε, Nmax)= 2.8018

a:= 4    b:= 8    Biseccion(f, a, b, ε, Nmax)= 6.1445

La grafica d>>la funcion f(x) utilizada en este caso se muestra a continuacion. Para producir esta grafica, pulse en algun sitio vacio en el cuadernno, y use la opcion "Insertar>Grafica>Dos-Dimensiones". En el punto de entrada en la esquina inferior izquierda de la grafica que resulta, escriba "f(x)". Para comenzar aparecera' la grafica de la izquierda:



f(x)



f(x)

La figura muestra las tres soluciones, o sean los puntos en que la grafica cruza el eje de las c. Estas soluciones son los valores encontrados anteriormente (-4.9473, 2.8018, y 6.1445). Se puede ajustar la escala vertical al pulsar en el icono "Escala" (segundo icono) en el panel "Grafica", y despues pulsar en la grafica, y apretando la tecla "Cntl", usar la ruedita del mouse. Ruedese la ruedita hacia abajo para incrementar el rango en la escala vertical. La grafica de la derecha muestra el resultado de ajustar la escala vertical. NOTA: Para ajustar la escala horizontal use un procedimiento similar, pero apriete la tecla "Shift" en vez de la tecla "Cntl".