

Método visual de la Secante

```

01: « 1 2 3 4 0
02: « → FA FB
03: « A B A - FB FA - / FA * - »
04: »
05: « 'X' STO EQ EVAL »
06: → FA FB FC C APR SEC F
07: «
08: 105 NEG FC? IF THEN
09: 105 NEG SF
10: 1 'APR' STO
11: END
12:
13: WHILE EQ TYPE 9 ≠ REPEAT
14: "Introduce una expresión
15: algebraica en X:" MSGBOX
16: "EQ(X) ej: 'SQ(X)-X'"
17: "' ' " INPUT OBJ→
18: DUP TYPE 9 == IF THEN
19: 'EQ' STO
20: END
21: END
22:
23: WHILE A TYPE 0 ≠ REPEAT
24: "Define el valor inicial de
25: A:" MSGBOX
26: "A:" " " INPUT OBJ→
27: DUP TYPE 0 == IF THEN
28: 'A' STO
29: END
30: END
31:
32: WHILE B TYPE 0 ≠ REPEAT
33: "Define el valor inicial de
34: B:" MSGBOX
35: "B:" " " INPUT OBJ→
36: DUP TYPE 0 == IF THEN
37: 'B' STO
38: END
39: END
40:
41: WHILE A B == REPEAT
42: "Los valores iniciales de A y
43: B no pueden coincidir. Dale
44: otro valor inicial a B:"
45: MSGBOX
    
```

Declaración de variables locales del programa.

¿Estamos en modo aproximado? Flag: -105 ✓

Comprobación de tipo de variable. EQ tiene que ser una expresión algebraica. TYPE: 9

Comprobación de tipo de variable. A tiene que ser un número real. TYPE: 0

Comprobación de tipo de variable. B tiene que ser un número real. TYPE: 0

¡Los valores de A y B no pueden coincidir! Sino el método de la secante no funciona.

```

46: "B:" " " INPUT OBJ→
47: DUP TYPE 0 == IF THEN
48: 'B' STO
49: END
50: END
51:
52: 1 4 START
53: DEPTH 0 > IF THEN
54: DUP TYPE 12 == IF THEN
55: DROP
56: END
57: END
58: NEXT
59:
60: A F EVAL 'FA' STO
61: B F EVAL 'FB' STO
62:
63: FA ABS 0 > FB ABS 0 > AND
64: IF THEN
65: FA FB SEC EVAL 'C' STO
66: C F EVAL 'FC' STO
67: CASE
68: FA FB * 0 < FA FC * 0 < AND
69: FA FB * 0 > A C - ABS
70: B C - ABS < AND OR
71: THEN
72: C 'B' STO
73: FC 'FB' STO
74: END
75: C 'A' STO
76: FC 'FA' STO
77: END
78: END
79:
80: A "A" →TAG
81: FA "F(A)" →TAG
82: B "B" →TAG
83: FB "F(B)" →TAG
84:
85: 'X' PURGE
86:
87: APR IF THEN
88: 105 NEG CF
89: END
90: »
91: »
    
```

Últimas líneas del bucle WHILE de la página anterior.

Quitamos las 4 líneas del resultado anterior para reemplazarlas y que no se amontonen.

Evaluamos EQ(X) en A y B y guardamos los resultados.

Si no hemos dado en el clavo, buscamos C, corte de la recta de unión entre (A, F(A)) y (B, F(B)) con el eje de abscisas, evaluamos EQ(C) y vemos en que variable guardamos estos nuevos valores.

Mostramos en pantalla los resultados etiquetados.

Borramos la variable global X.

Restauramos la flag -105 en caso necesario.

Explicación del código:

Se trata de un método de la secante visual, que va paso a paso en vez de converger directamente a la solución. De esta forma, el alumno que tenga que escribir en la solución de un problema el desarrollo de este método lo va a tener mucho más fácil.

El método de la secante converge cuando la función que vamos a evaluar es localmente monótona creciente o decreciente. Por tanto es fundamental dar una primera aproximación a los valores de A y B que estén en este rango de validez.

A parte de estimar órdenes de magnitud, una manera rápida de obtener esta aproximación inicial es pintar la función con la calculadora. Así podemos obtener puntos cercanos al corte que nos interese con el eje de abscisas utilizando la función TRACE y moviéndonos por nuestra curva.

01-06: El primer bloque de código declara las variables locales del programa. Éstas son: FA, FB, FC, C, APR, SEC y F. Las 5 primeras son números reales y las dos últimas son funciones.

Los valores iniciales: 1, 2, 3, 4, correspondientes a las 4 primeras variables son arbitrarios. En cambio, el 0 correspondiente a APR es obligatorio porque va a jugar un papel de variable condicional que inicialmente está en modo FALSE.

La función SEC recibe 2 valores que asocia a sus variables locales propias FA y FB. Dentro de una función declarada localmente no se reconocen las variables locales del propio programa, y por eso tenemos que pasarle los valores desde fuera. Podríamos haber llamado a estas dos variables con otros nombres, pero por simplificar las llamamos igual que sus análogas en el programa principal. Esta función calcula el punto de intersección de la recta que une los puntos (A, F(A)) y (B, F(B)), con el eje de abscisas.

La función F evalúa la expresión algebraica EQ(X) almacenada en la variable global EQ. Utilizamos el par EQ, X porque de esta manera podemos interactuar con el editor gráfico de la calculadora para mostrar nuestra función sin tener que definirla de nuevo.

08-11: Lo primero que hacemos en el programa es comprobar si la calculadora está en modo aproximado o exacto. Para que el programa funcione debe estar en modo aproximado, así que comprobamos que la flag -105 esté marcada. Si no lo está, la marcamos y cambiamos el valor de la variable condicional APR a 1 (TRUE), que al final del programa nos ayudará a dejar la configuración del sistema tal como estaba.

13-50: El segundo paso que tenemos que dar es comprobar que las variables globales que nos llegan (EQ, A y B) estén correctamente definidas.

La variable EQ tiene que ser de tipo algebraico (9). Por tanto, si no lo es, o si no está definida, le pedimos al usuario que la introduzca.

En las líneas 16 y 17 hay dos cosas que comentar sobre la función INPUT. Primero, de los 2 argumentos que recibe entre comillas dobles, el segundo tiene dentro unas comillas simples. Esto es así para que el usuario escriba dentro de dichas comillas la expresión algebraica tal como se le dice en el ejemplo. Segundo, la función INPUT devuelve una cadena de caracteres entre comillas dobles, para quitarlas y guardar la expresión algebraica que presumimos va a introducir el usuario, utilizamos la función OBJ→.

Antes de guardar el elemento introducido chequeamos que sea una expresión algebraica, es decir, una variable de tipo 9. Si no lo es, seguimos en el bucle WHILE, y no la guardamos en EQ. Lo único que no comprobamos es que la variable independiente de dicha expresión algebraica sea X, para ello solo nos queda confiar en el usuario.

La siguiente comprobación que hacemos es que los valores de A y B estén definidos y sean reales, es decir, de tipo 0. El proceso es similar al anterior.

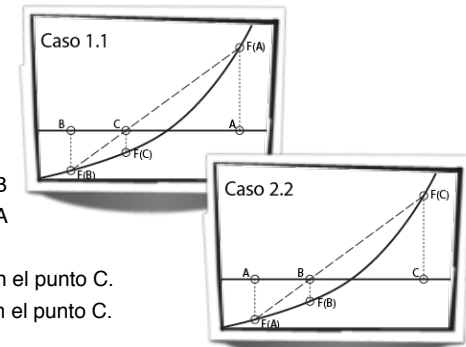
Por último, comprobamos que los valores de A y B sean diferentes, y en caso contrario pedimos que se introduzca otro valor de B.

52-58: El siguiente bloque chequea si es la primera vez que ejecutamos el programa. Cada vez que lo ejecutemos se van a mostrar 4 líneas en la pila, y para no amontonarlas las vamos a ir remplazando. Para ello miramos si las cuatro primeras líneas de la pila son variables con etiqueta como las que vamos a mostrar, de tipo 12, en cuyo caso las borramos.

60-61: Evaluamos EQ en A y en B y guardamos los resultados en FA y en FB. Es importante utilizar el comando EVAL tras llamar a nuestra función F, sino no se ejecuta.

63-78: Comprobamos que los valores de FA y FB son distintos de 0 y si lo son llamamos a la función SEC pasándole los valores de FA y FB, como se dijo anteriormente, para obtener C. Obtenemos FC. Para decidir en qué variable conviene almacenar el valor de C tenemos que diferenciar entre 4 casos:

1. FA y FB son de distinto signo
 - 1.1. FA y FC son de distinto signo
 - 1.2. FA y FC son del mismo signo
2. FA y FB son del mismo signo
 - 2.1. C está más cerca de A que de B
 - 2.2. C está más cerca de B que de A



En los casos 1.1 y 2.1: B pasa a estar en el punto C.

En los casos 1.2 y 2.2: A pasa a estar en el punto C.

Se podría haber utilizado el condicional IF en vez de CASE, que es de mayor utilidad cuando hay más de 2 casos, pero se ha optado por CASE para ver su estructura. También podríamos haber separado en 4 casos el condicional ganando claridad de código, lo malo es que duplicaríamos líneas.

80-89: Por último mostramos en pantalla los valores tras etiquetarlos, borramos la variable X, que es la única global que no queremos guardar, y dependiendo de que hubiéramos cambiado la flag del modo exacto la devolvemos a su estado original o la dejamos tal cual.