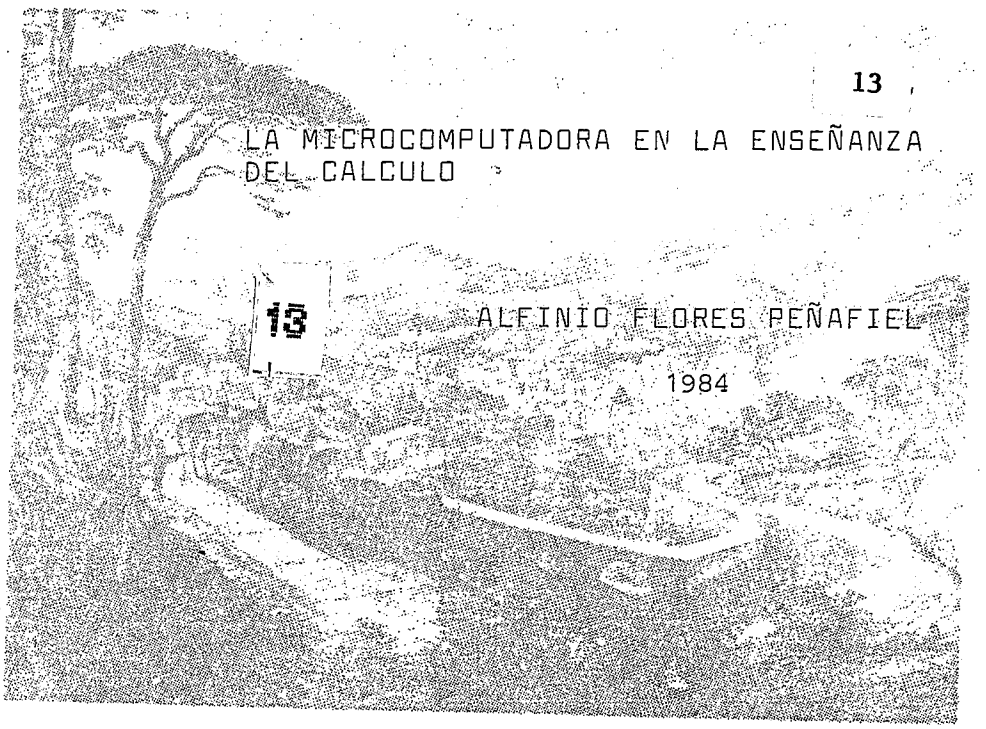
COMUNICACIONES DEL CIMAT



Presentado en: Congreso Anual de la Sociedad Matemática Mexicana, noviembre de 1984, Mérida, Yuc.

CENTRO DE INVESTIGACION EN MATEMATICAS

Apartado Postal 402
Guanajuato, Gto.
México
Tels. (473) 2-25-50
2-02-58

LA MICROCOMPUTADORA EN LA ENSEÑANZA DEL CALCULO

Alfinio Flores Peñafiel
Centro de Investigación en Matemáticas

RESUMEN Usando programas cortos la microcomputadora puede ayudar en la enseñanza del cálculo. Se dan ejemplos para graficar, para ilustrar el método de incrementos, la aproximación lineal a una función, la obtención de ceros de funciones, y la integración numérica.

INTRODUCCION

La microcomputadora tiene un potencial muy grande para modificar la enseñanza de las matemáticas. Puede ser un medio para lograr un curriculum de matemáticas más balanceado. Con la computadora se puede establecer la conexión entre la teoría y la práctica, asímismo lograr la unificación de las matemáticas puras y aplicadas. La computadora puede también facilitar el uso de las matemáticas en otras áreas. Además los alumnos tienen así la oportunidad de mejorar su sentido de los números, desarrollar el pensamiento algorítmico, y practicar el planteamiento y la resolución de problemas. Estas habilidades serán cada vez más importantes en el curriculum de matemáticas (Conference Board of the Mathematical Sciences, 1982).

Usar una microcomputadora en la clase de cálculo tiene varios beneficios. Hickernell y Proskurowski (1984) mencionan los siguientes:

- ayuda a familiarizar a los alumnos con las computadoras al principio de su formación universitaria
- estimula la intuición geométrica a través de la graficación
- mejora la comprensión conceptual de las matemáticas
- permite al alumno considerar ejemplos sofisticados y más realistas
- motiva al alumno a trabajar más y pensar en términos matemáticos
- estimula el interés en el cálculo

No es necesario contar con equipo de cómputo muy sofisticado y costoso para obtener estos beneficios. Aún con computadoras muy baratas, sin necesidad de equipos periféricos caros se pueden lograr los mismos resultados, utilizando programas muy cortos (12 - 15 renglones).

Algunos investigadores en educación matemática han encontrado paralelismos entre el pensamiento necesario para escribir, probar, corregir y refinar un programa y varios aspectos del pensamiento matemático. Hatfield (1982) menciona los siguientes procesos del pensamiento matemático como algunos de los que pueden ser mejorados a través de programar una computadora:

analizar
simplificar
particularizar
generalizar
justificar
conjeturar
estructurar

En las siguientes secciones se da un bosquejo de cómo la computadora puede ser utilizada en algunos temas de cálculo. Se ha tratado que cada aspecto quede ilustrado por un programa corto de computación. Estos programas pueden después estructurarse en programas más grandes que estudien varios aspectos a la vez.

Es importante hacer notar que los alumnos interactúan constantemente con la computadora. Ellos escriben, ejecutan y modifican sus programas, cambian parámetros y datos, y observan los resultados numéricos y gráficos en la pantalla.

No se pretende que éste sea un curso de programación. La computación es aquí un medio para aprender mejor conceptos matemáticos, y no es un objeto de estudio en sí mismo.

Con el objeto de dar una idea mejor de lo que hacen los programas se dan ejemplos. En algunos casos se menciona el tiempo aproximado que tardó el programa en ser ejecutado. Estos tiempos son para una computadora Timex-Sinclair 2068 que es una de las microcomputadoras más baratas.

La utilización del lenguaje BASIC para escribir los programas es debida sólo al hecho a que es el lenguaje más difundido entre las microcomputadoras y no requiere hacer ningún gasto adicional. Los programas de este trabajo están escritos en la versión de BASIC utilizada por la computadora Timex-Sinclair, pero pueden ser facilmente modificados para cualquier otra computadora.

1 PROGRAMAS PARA GRAFICAR

No es necesario enfatizar el hecho de que la computadora facilita los cálculos para graficar. Además la computadora tiene elementos dinámicos que la imagen en un libro no puede proporcionar. El hecho mismo de que la computadora grafique utilizando un conjunto discreto de puntos y tome tiempo en graficar puede tener ventajas didácticas. Por ejemplo, al graficar en coordenadas cartesianas y = x * x, la separación entre los puntos no será uniforme, dando idea de que la velocidad con la que crece la función no es constante. En el caso de ecuaciones paramétricas, aunque las ecuaciones y = t*t*t, x = t*t*t dan una recta al igual que las ecuaciones y = t, x = t, al correr el programa se ve claramente que la velocidad con la que se recorre la curva es diferente en cada caso, cosa que sería difícil apreciar en un dibujo terminado.

1.1 PROGRAMA PARA COORDENADAS CARTESIANAS

Este programa grafica una función y = f(x) en coordenadas cartesianas.

```
5  INPUT " dame la escala " ; d
10 FOR x = -128/d TO 127/d STEP 1/d
20 LET y = x * x
30 GOSUB 100
40 NEXT x
99 STOP
100 REM ****************************************
                  subrutina para graficar
        ****************************************
110 LET h = d * x + 128
120 LET v = d * y + 88
130 IF v < 0 OR v > 175 THEN RETURN
140 PLOT h , v
150 RETURN
```

La computadora tarda en graficar las siguientes funciones aproximadamente los tiempos indicados:

y = x * x	10 segundos
y = x * x * x	12 segundos
y = EXP(x)	20 segundos
y = SIN (EXP x)	35 segundos

1.2 PROGRAMA PARA ECUACIONES PARAMETRICAS

Este programa grafica curvas dadas por ecuaciones paramétricas, y = f(t), x = g(t).

```
5  INPUT " dame la escala " ; d
10 FOR t = -6.3 TO 6.3 STEP .05
15 LET x = t * t * t
20 LET y = t * t
30 GOSUB 100
40 NEXT t
99 STOP
100 REM ***************************
          subrutina para graficar
        ***************************
110 LET h = d * x + 128
120 LET v = d * y + 88
125 IF h < 0 OR h > 255 THEN RETURN
130 IF v < 0 OR v > 175 THEN RETURN
140 PLOT h , v
150 RETURN
```

El tiempo necesario para dibujar las siguientes curvas es:

x=t*t*t,	y=t*t	12 segundos
x=t*t,	y=EXP(-t)	22 segundos
x=SIN(t),	y=EXP(t)	35 segundos
x=SIN(t),	y=COS(3*t)	35 segundos

1.3 PROGRAMA PARA COORDENADAS POLARES

Este programa grafica curvas dadas en coordenadas polares. Para hacer esto, primero transforma las coordenas de cada punto a coordenadas cartesianas.

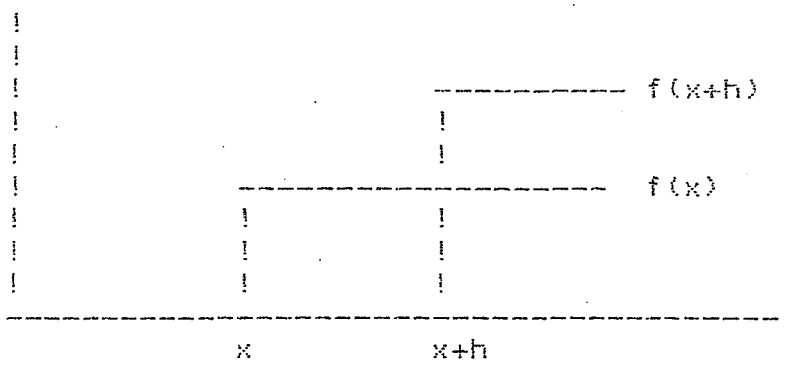
```
5  INPUT " dame la escala " ; d
10 FOR t = -6.3 TO 6.3 STEP .05
20 LET r = EXP (t)
30 LET x = r * COS(t)
40 LET y = r * SIN(t)
50 GOSUB 100
60 NEXT t
99 STOP
100 REM ***************************
          subrutina para graficar
        ***************************
110 LET h = d * x + 128
120 LET v = d * y + 88
125 IF h < 0 OR h > 255 THEN RETURN
130 IF v < 0 OR v > 175 THEN RETURN
140 PLOT h , v
150 RETURN
```

Para dibujar las siguientes funciones la máquina tardó:

r=t/3	40 segundos
r=EXP(t/3)	50 segundos
r=t*t/5	55 segundos

2 EL METODO DE INCREMENTOS

Por el método de incrementos se calculan las pendientes de rectas secantes a la curva. Al hacer el incremento cada vez más pequeño, las secantes se van pareciendo cada vez más a la tangente a la curva. El límite de las pendientes de las secantes nos da la pendiente de la tangente.



2.1 PROGRAMA PARA CALCULAR LA PENDIENTE DE LA SECANTE A UNA CURVA

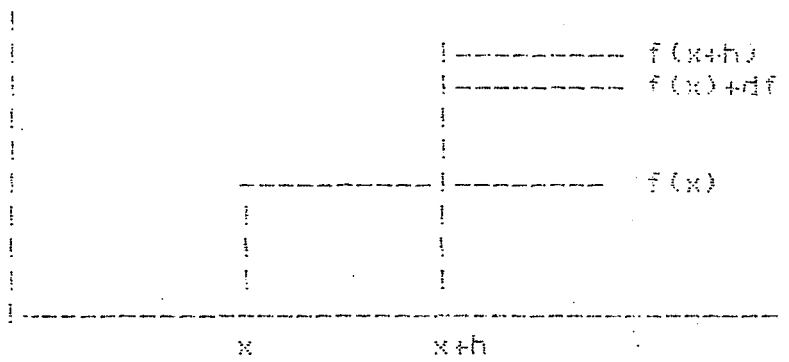
```
10 DEF FN y(x) = x * x
20 INPUT " punto inicial "; x
25 INPUT " incremento "; h
30 FOR N = 1 TO 10
40 LET s = ( FN y(x+h) - FN y(x) ) / h
50 PRINT h , s
60 LET h = h / 2
70 NEXT N
```

Ejemplo: para x = 1, h = 1 se obtiene (2 segs)

h	s
1	3
0.5	2.5
0.25	2.25
0.125	2.125
.0625	2.0625
.03125	2.03125
.015625	2.015625
.0078125	2.0078125
.00390625	2.0039062
.001953125	2.0019531
.0009765625	2.0009766
.00048828125	2.0004883
.00024414063	2.0002441
.00012207031	2.0001221
.000061035156	2.0000610

3 APROXIMACION LINEAL A UNA FUNCION

La derivada nos da información local de qué tan rápido está creciendo una función. Conociendo el valor de la función y de su derivada en un punto podemos calcular aproximaciones a los valores de la función para puntos cercanos.



3.1 PROGRAMA QUE COMPARA EL ERROR DE LA APROXIMACION LINEAL CON EL INCREMENTO

```
10 DEF FN y(a) = a * a
20 INPUT " valor de x " ; x
30 INPUT " incremento " ; h
40 LET dy = 2 * x * h
50 LET e = FN y(x+h) - FN y(x) - dy
60 PRINT e , e/h
```

El programa puede ser modificado para tomar incrementos cada vez más pequeños añadiendo estas líneas :

```
35 FOR N = 1 TO 20
70 LET h = h / 2
80 NEXT N
```

Corriendo el programa extendido se obtiene:

e	e/h
1	1
0.25	0.5
.0625	0.25
.015625	0.125
.00390625	0.0625
.0009765625	0.03125
.00024414063	0.015625
.000061035156	0.0078125
.000015258789	0.0039062
3.8146973E-6	0.0019531
9.5367432E-7	0.0009766
2.3841858E-7	0.0004883
5.9604645E-8	0.0002441
1.4901161E-8	0.0001221
3.7252903E-9	0.0000610

Los alumnos pueden así notar que para incrementos chicos, el error es muy pequeño comparado con el incremento.

USO DE LA APROXIMACION LINEAL PARA OBTENER RAICES

El hecho que para valores pequeños de h la aproximación lineal a (x + h) * (x + h) dada por x * x + 2 * x * h, es muy buena nos permite entender por qué funciona tan bien el antiguo algoritmo para calcular raíces cuadradas conocido por los babilonios. Con este método, si n es el número del cual queremos obtener la raíz, y r es nuestra primera aproximación, entonces (r + n / r) / 2, será [illegible] Repitiendo el proceso, podemos obtener la precisión que queramos.

3.2 PROGRAMA PARA APROXIMAR LA RAIZ CUADRADA DE UN NUMERO

```
10 INPUT " número " ; n
20 INPUT " aproximación " ; a
30 PRINT a , a * a
40 LET a = ( a + n / a ) / 2
50 IF ABS (a * a - n) > .0000001 THEN GOTO 30
60 PRINT a , a * a
```

Ejemplo: para n = 5, a = 2

a	a*a
2	4
2.25	5.0625
2.236111	5.0001929
2.236068	5

Aún cuando la primera aproximación no sea muy buena, sólo se necesitan unos cuantos pasos más. Por ejemplo, para n = 30, si hacemos a = 1 se tiene:

a	a * a
1	1
15.5	240.25
8.7177419	75.999024
6.0795	36.96032
5.5070582	30.32769
5.4773064	30.000885
5.4772256	30

3.3 PROGRAMA PARA APROXIMAR LA RAIZ CUBICA DE UN NUMERO

La aproximación lineal a (x + h) * (x + h) * (x + h), dada por x * x * x + 3 * x * x * h, nos permite obtener un algoritmo semejante al anterior para obtener raíces cúbicas.

```
10 INPUT " número " ; n
20 INPUT " aproximación " ; a
30 PRINT a , a * a * a
40 LET a = ( 2*a + n/(a*a) ) / 3
50 IF ABS (a * a * a - n) > .0000001 THEN GOTO 30
60 PRINT a , a * a * a
```

Ejemplo: para n =10, a = 1

1	1
4	64
2.875	23.763672
2.3199433	12.486252
2.1659616	10.161369
2.1544959	10.000853
2.1544347	10

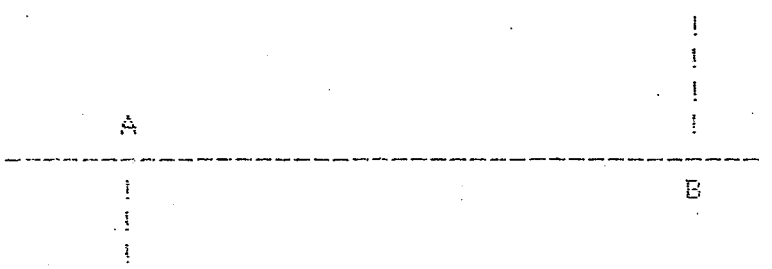
4 CEROS DE FUNCIONES

En esta sección se verán dos métodos para obtener ceros de funciones. En ambos métodos se utiliza una estrategia muy poderosa y de amplia aplicación para resolver problemas:

ESTIMA
VERIFICA SI TU ESTIMACION ES SUFICIENTEMENTE BUENA
OBTEN UNA MEJOR APROXIMACION A PARTIR DE LA ANTERIOR
REPITE EL PROCESO

4.1 EL METODO DE BISECCION

Este método usa el hecho de que si una función continua cambia de signo, debe intersectar al eje horizontal.



Para el siguiente programa se debe encontrar un punto A donde f(A) < 0 y un punto B donde f(B) > 0

4.1 PROGRAMA METODO DE BISECCION

```
10 DEF FN y(a) = 14.5 + a * (-2.7 + a * (4 + a))
20 INPUT " intervalo " ; a , b
30 LET r = ( a + b ) / 2
40 PRINT r , FN y(r)
50 IF ABS FN y(r) < .0000001 THEN STOP
60 IF FN y(r) > 0 THEN LET b = r
70 IF FN y(r) < 0 THEN LET a = r
80 GOTO 30
```

Ejemplo: tomando a = -5, b = -6, r toma los valores

-5.5
-5.25
-5.125
-5.1875
-5.15625
-5.140625
-5.1484375
-5.1445313
-5.1464844
-5.1474609
-5.1469727
-5.1472168
-5.1473389
-5.1472778
-5.1473083
-5.1473236
-5.1473160
-5.1473122
-5.1473103
-5.1473093
-5.1473098
-5.1473095
-5.1473097

Como puede observarse, para ganar una precisión de tres cifras más, es necesario repetir el proceso aproximadamente diez veces.

4.2 METODO DE NEWTON PARA OBTENER CEROS DE UNA FUNCION

A diferencia del método anterior, el método de Newton utiliza la información que nos da la derivada de la función y el hecho de que para incrementos pequeños la aproximación lineal a una función es muy buena. La esencia de este método se puede ilustrar geometricamente. Si x1 es una buena aproximación a un cero de la función, entonces la tangente a la curva en (x1 , f(x1)) intersectará al eje horizontal en una mejor aproximación.

La ecuación de la tangente es

$$y - f(x1) = f'(x1)(x-x1)$$

La nueva aproximación está dada por

$$x2 = x1 - f(x1) / f'(x1)$$

Repitiendo este proceso podemos ir obteniendo aproximaciones cada vez mejores.

4.2 PROGRAMA METODO DE NEWTON

```
10 DEF FN y(a) = 16.5 + a*(-2.7 + a*(4 + a))
20 DEF FN z(a) = -2.7 + a*(8 + 3*a)
30 INPUT "aproximación " ; x
40 PRINT x , FN y(x)
50 IF ABS FN y(x) < .0000001 THEN STOP
60 LET x = x - FN y(x) / FN z(x)
70 GOTO 40
```

Ejemplo

-5	5
-5.1547988	-0.26729839
-5.1473276	-.00063952804
-5.1473097	7.4505806E-9

Otro ejemplo

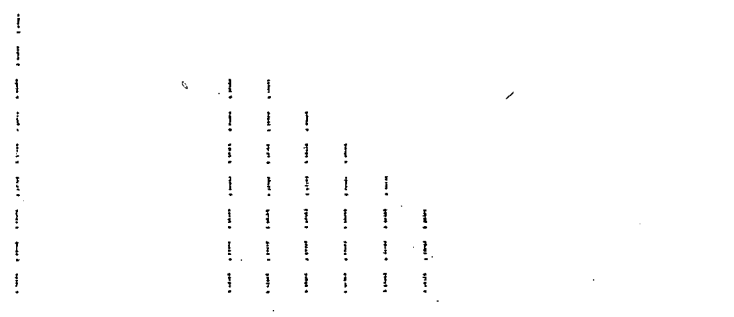
-6	-39.3
-5.3141361	-6.2630928
-5.1556051	-0.29615355
-5.1473317	-.00078430772
-5.1473097	7.4505806E-9

En ambos casos se ve que con este método la aproximación al cero de la función es mucho más rápida que con el método de bisección. Esto ilustra un punto muy importante en la enseñanza de las matemáticas. No es sólo que la computadora nos facilita los cálculos y operaciones y así ayuda al estudio de las matemáticas, sino que también el estudio de matemáticas más avanzadas y sofisticadas nos puede guiar para obtener métodos numéricos más efectivos. En el caso del método de Newton, la teoría matemática nos permite saber que nuestra siguiente aproximación no sólo será mejor que la anterior, sino mucho mejor. Por otra parte nos permite ver en qué casos el método de Newton no es el más indicado.

5 INTEGRACION

5.1 PROGRAMA PARA APROXIMAR EL AREA BAJO UNA CURVA POR MEDIO DE RECTANGULOS

Este programa aproxima el área bajo la función dada entre los puntos A y B, por medio de rectángulos usando subdivisiones del mismo tamaño.



5.1 INTEGRACION POR RECTANGULOS

```
10 DEF FN y(x) = x * x
20 INPUT " intervalo " ; a , b
30 INPUT " número de subdivisiones " ; n
40 LET s = 0
50 LET dx = (b-a)/n
60 FOR i = 1 TO n
70 LET x = a + i * dx
80 LET r = dx * FN y(x)
90 LET s = s + r
100 NEXT i
110 PRINT s
```

Ejemplo
Para a = 0, b = 1, n = 100 se tiene s = .33835 (3 segundos)

Modificando el renglón

```
60 FOR i = 0 TO n - 1
```

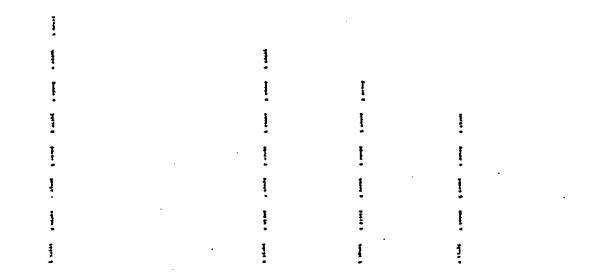
se obtiene una aproximación por rectángulos contenidos dentro del área.
Para a = 0, b = 1, n = 100 se tiene s =.32835

Si se quiere mejorar la aproximación, se puede aumentar el número de subdivisiones. Aproximando por fuera, para a = 0, b = 1, n = 1000 se tiene s = .3338335 (25 segundos) y aproximando por dentro s = .3328335 . Para n = 10000, s =.33328334 por dentro y s = .33338334 por fuera. Para ganar una cifra más de precisión necesitamos aproximadamente 10 veces más tiempo.

Para el mismo intervalo 0,1 pero con la función y=x*TANx, se necesitan para n = 100, 12 segundos, y para n = 1000, 2 minutos.

5.2 APROXIMACION POR MEDIO DE TRAPECIOS

Una forma de obtener una mejor aproximación sin tener que aumentar demasiado el número de subdivisiones, es promediar las aproximaciones obtenidas por dentro y por fuera. Esto equivale a aproximar por trapecios.



5.2 PROGRAMA INTEGRACION POR TRAPECIOS

```
10 DEF FN y(x) = x * x
20 INPUT " intervalo " ; a , b
30 INPUT " número de subdivisiones " ; n
40 LET s = 0
50 LET dx = (b-a)/n
60 FOR i = 1 TO n - 1
70 LET x = a + i * dx
80 LET t = FN y(x)
90 LET s = s + t
100 NEXT i
110 LET s = s + ( FN(a) + FN(b) ) / 2
120 PRINT s * dx
```

Ejemplo
Para a = 0, b = 1, n = 100, se tiene s = .33335 (3 seg)
Para a = 0, b = 1, n = 1000 se tiene s = .3333335 (26 seg)
Como se ve, la aproximación es mucho mejor ahora, ya que el valor real del área es 1/3.
Este programa se puede utilizar para aproximar integrales definidas para funciones para las que no se pueda encontrar una antiderivada en forma explícita, tal como y(x) = x * tan(x). Para esta función se obtuvieron:
para n = 100, s = .42812982 (12 segundos),
para n = 1000, s = .42808872 (2 minutos).

REFERENCIAS

Conference Board of the Mathematical Sciences. The mathematical sciences curriculum: what is still fundamental and what is not. Washington, National Science Foundation, 1982.

Hatfield, L. L. (1982). " Instructional computing in mathematcs teacher education."Journal of Research Development in Education, 15 (4), 30-44.

Hickernell, F., y Proskurowski, Wlodek (1984). "Teaching calculus with computers at U S C." SIAM News, 18 (5),9,16.

AGRADECIMIENTOS

A mis alumnos de cálculo de la Escuela de Matemáticas de la Universidad de Guanajuato, quienes desde el primer día tuvieron una computadora frente a ellos.
A mis colegas del Centro de Investigación en Matemáticas por sus críticas constructivas y sugerencias al presente trabajo.