

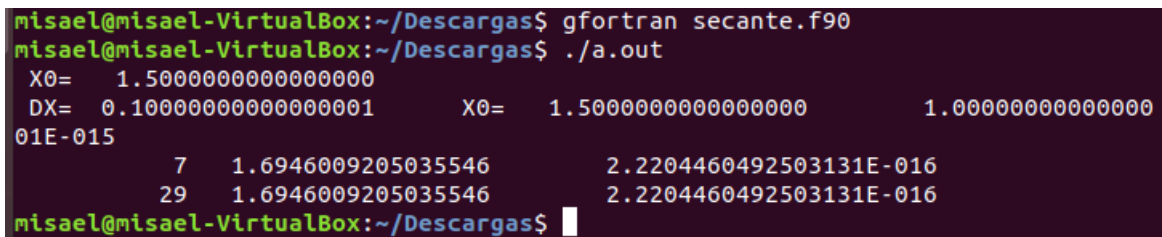
1. Investiga el método de la posición falsa para encontrar raíces de una función y a partir del programa de la secante (como subrutina) implementa el método

```
PROGRAM secante
!Main program to use the Secant Method to find the root of
! f(x)=exp(x)*ln(x)-x*x=0.
!
REAL*8 :: DL,A,B,DX,X0,H,C
INTEGER :: ISTEP
C = 0.0
DL = 1.0D-15
A = 1.0
B = 2.0
DX = (B-A)/10.0
X0 = (A+B)/2.0
PRINT*, 'X0=', X0
CALL SECANT (DL,X0,DX,ISTEP)
PRINT*, ISTEP,X0,DX
DX = (B-A)/10.0
CALL FP (A,B,C,DX,DL,ISTEP)
PRINT*, ISTEP,C,DX
END PROGRAM SECANTE
!
SUBROUTINE SECANT (DL,X0,DX,ISTEP)
!
! Subroutine for the root of f(x)=0 with the secant method.
!
IMPLICIT NONE
INTEGER, INTENT (INOUT) :: ISTEP
REAL*8, INTENT (INOUT) :: X0,DX
REAL*8 :: X1,X2,D,F,FX
REAL*8, INTENT (IN) :: DL
PRINT*, 'DX=', DX, 'X0=', X0, DL
ISTEP = 0
X1 = X0 + DX
DO WHILE (DX.GT.DL)
D = F(X1) - F(X0)
X2 = X1 - F(X1)*(X1-X0)/D
X0 = X1
X1 = X2
DX = ABS(X1 - X0)
ISTEP = ISTEP + 1
ENDDO
END SUBROUTINE SECANT
!
SUBROUTINE FP(A,B,C,DX,DL,ISTEP)
IMPLICIT NONE
```

```

INTEGER, INTENT (INOUT) :: ISTEP
REAL*8, INTENT (IN) :: A,B,DL
REAL*8 :: F,X,Y
REAL*8, INTENT (INOUT) :: C,DX
X=B
Y=A
ISTEP=0
DO WHILE (DX.GT.DL)
  ISTEP=ISTEP+1
  C=(F(X)*Y-F(Y)*X)/(F(X)-F(Y))
  IF (F(A)*F(C).LT.0) THEN
    X=C
    DX=ABS(Y-C)
  ELSE
    Y=C
    DX=ABS(X-C)
  END IF
END DO
END SUBROUTINE FP
!
FUNCTION F(X)
IMPLICIT NONE
REAL*8 :: F
REAL*8, INTENT (IN) :: X
F = EXP(X)*LOG(X) - X*X
END

```



```

misael@misael-VirtualBox:~/Descargas$ gfortran secante.f90
misael@misael-VirtualBox:~/Descargas$ ./a.out
X0= 1.5000000000000000
DX= 0.10000000000000001      X0= 1.5000000000000000      1.0000000000000000
01E-015
          7  1.6946009205035546      2.2204460492503131E-016
          29 1.6946009205035546      2.2204460492503131E-016
misael@misael-VirtualBox:~/Descargas$

```

El método de la posición falsa tarda poco más de 4 veces que el método de la secante lo que es bastante

2. Haz un programa que haga distintas aproximaciones ($h \rightarrow 0$) a la derivada de una función que conozcas utilizando la precisión “extra” que hemos visto para un valor determinado comparando con el resultado exacto. ¿cuál es la mejor h ?

```

PROGRAM derivada
IMPLICIT NONE
INTEGER :: i
INTEGER, PARAMETER :: extra = SELECTED_REAL_KIND(p=24,r=1000)

```

```

REAL(extra) :: x,hh,xx,dif,ff
WRITE(*,*) "x=7.1"
WRITE(*,*) "f(x)=x**(1/2)"
WRITE(*,*) "f'(x)=1/2*sqrt(x)"

x=7.1_extra
xx= 1/(2*SQRT(x))

WRITE(*,*) "          ", "i/h", "          ", "f'(x)", "          ", "f'(x)(approx)",

DO i=1,24
  WRITE(*,*) i
  hh=1._extra/10._extra**i
  ff=(SQRT(x+hh)-SQRT(x))/hh
  dif=ABS((ff-xx)/xx)
  PRINT 10,hh ,xx,ff, dif
10 FORMAT(2X,6F16.13)
END DO

END PROGRAM derivada

```

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
misael@misael-VirtualBox:~/Descargas$ nano derivada.f90
misael@misael-VirtualBox:~/Descargas$ gfortran derivada.f90
misael@misael-VirtualBox:~/Descargas$ ./a.out
x=7.1
f(x)=x**(1/2)
f'(x)=1/2*sqrt(x)
          i/h          f'(x)          f'(x)(approx)          |f'-sqrt(x)/sqrt(x)|
1
0.100000000000000 0.1876466562602 0.1869905410490 0.0034965462442
2
0.010000000000000 0.1876466562602 0.1875806299831 0.0003518649274
3
0.001000000000000 0.1876466562602 0.1876400494488 0.0000352087882
4
0.000100000000000 0.1876466562602 0.1876459955372 0.0000035211020
5
0.000010000000000 0.1876466562602 0.1876465901875 0.0000003521124
6
0.000001000000000 0.1876466562602 0.1876466496529 0.0000000352113
7
0.000000100000000 0.1876466562602 0.1876466555995 0.0000000035211
8
0.000000010000000 0.1876466562602 0.1876466561941 0.0000000003521
9
0.000000001000000 0.1876466562602 0.1876466562536 0.00000000000352
10
0.000000000100000 0.1876466562602 0.1876466562595 0.00000000000035
11
0.000000000001000 0.1876466562602 0.1876466562601 0.00000000000004

```

Archivo	Editar	Ver	Buscar	Terminal	Ayuda
11					
0.00000000000100	0.1876466562602	0.1876466562601	0.00000000000004		
12					
0.00000000000010	0.1876466562602	0.1876466562602	0.00000000000000		
13					
0.00000000000001	0.1876466562602	0.1876466562602	0.00000000000000		
14					
0.00000000000000	0.1876466562602	0.1876466562602	0.00000000000000		
15					
0.00000000000000	0.1876466562602	0.1876466562602	0.00000000000000		
16					
0.00000000000000	0.1876466562602	0.1876466562602	0.00000000000000		
17					
0.00000000000000	0.1876466562602	0.1876466562602	0.00000000000000		
18					
0.00000000000000	0.1876466562602	0.1876466562602	0.00000000000000		
19					
0.00000000000000	0.1876466562602	0.1876466562602	0.00000000000000		
20					
0.00000000000000	0.1876466562602	0.1876466562602	0.00000000000000		
21					
0.00000000000000	0.1876466562602	0.1876466562601	0.00000000000004		
22					
0.00000000000000	0.1876466562602	0.1876466562574	0.00000000000148		
23					
0.00000000000000	0.1876466562602	0.1876466562189	0.00000000002201		
24					
0.00000000000000	0.1876466562602	0.1876466562959	0.00000000001905		

El mejor valor de h debe estar entre $1/10^{11}$ y $1/10^{21}$, intenté usar mayor precisión para los decimales pero el programa regresaba puros asteriscos :(

3. Haz un programa que rote una imagen en formato .bmp 180 grados. El problema se reduce a cambiar (en tono de grises) el primer píxel por el último, el segundo por penúltimo, el tercero por el antepenúltimo, el cuarto...y así hasta llegar a la mitad, ¿verdad? (ver videos de arreglos)

```
PROGRAM ROTACION
```

```
IMPLICIT NONE
```

```
CHARACTER*30 :: NOM1,NOM2
```

```
INTEGER :: NTAM,NL,NA,NPIX,NEN,i
```

```
INTEGER*1, ALLOCATABLE, DIMENSION(:) :: x,y
```

```
WRITE(*,*) "Indica el nombre del archivo de imagen"
```

```
READ(*,*) NOM1
```

```
!NOM1 = "gato.bmp"
```

```
WRITE(*,*) "Indica el nombre de archivo rotado"
```

```
READ(*,*) NOM2
```

```
WRITE(*,*) "Escribe el tamaño en bytes"
```

```

READ(*,*) NTAM
!NTAM = 172854

WRITE(*,*) "Indica el largo de la imagen en pixeles"
READ(*,*) NL
!NL = 180

WRITE(*,*) "Indica el alto de imagen en pixeles"
READ(*,*) NA
!NA = 320

WRITE(*,*) "Indica el numero de bytes por pixel"
READ(*,*) NPIX
!NPIX = 3

ALLOCATE(x(NTAM),y(NTAM))

OPEN(1,FILE=NOM1,ACCESS="DIRECT",FORM="UNFORMATTED",STATUS="OLD",RECL=NTAM)!abre
READ(1,REC=1)(x(i),i=1,NTAM)
NEN = NTAM - NL*NA*NPIX

DO i = 1,NEN !copia el encabezado
  y(i)=x(i)
END DO

DO i = NEN+1,NTAM !intercambia los pixeles para rotar 180 grados
  y(i) = x(NTAM-i+NEN)
END DO

OPEN(2,FILE=NOM2,ACCESS="DIRECT",FORM="UNFORMATTED",STATUS="NEW",RECL=NTAM) !abre
WRITE(2,REC=1)(y(i),i=1,NTAM)

CLOSE(1)
CLOSE(2)

END PROGRAM ROTACION

```

Las imágenes original y rotada se pueden ver a continuación, por alguna razón que aun no entiendo la imagen rotada se hizo un poco rojiza pero parece estar bien

```
misael@misael-VirtualBox:~/Descargas$ nano rotacion.f90
misael@misael-VirtualBox:~/Descargas$ gfortran rotacion.f90
misael@misael-VirtualBox:~/Descargas$ ./a.out
Indica el nombre del archivo de imagen
gato.bmp
Indica el nombre de archivo rotado
gatorot.bmp
Escribe el tamaño en bytes
172854
Indica el largo de la imagen en pixeles
180
Indica el alto de imagen en pixeles
320
Indica el numero de bytes por pixel
3
misael@misael-VirtualBox:~/Descargas$
```

