Evaluacion 2

Cabello Lopez Marco Antonio
Departamento de Fisica
Universidad de Sonora

Hermosilo, Sonora a Viernes 1 de Diciembre del 2017

1 Problema 1: Taylor

```
program taylor

    implicit none
real (kind=8) :: x, exp_true, y
    real (kind=8), external :: exptaylor
    integer :: n

    n = 20               ! number of terms to use
    x = 1.0
    exp_true = exp(x)
    y = exptaylor(x,n)   ! uses function below
    print *, "x = ",x
    print *, "exp_true  = ",exp_true
    print *, "exptaylor = ",y
    print *, "error     = ",y - exp_true

end program taylor

!==========================
function exptaylor(x,n)
!==========================
    implicit none

    ! function arguments:
    real (kind=8), intent(in) :: x
    integer, intent(in) :: n
    real (kind=8) :: exptaylor

    ! local variables:
    real (kind=8) :: term, partial_sum
    integer :: j

    term = 1.
    partial_sum = term

    do j=1,n
        ! j'th term is  x**j / j!  which is the previous term times x/j:
        term = term*x/j
        ! add this term to the partial sum:
        partial_sum = partial_sum + term
        end do
     exptaylor = partial_sum  ! this is the value returned
end function exptaylor
```

Resultados: x = 1.0000000000000000
exptrue = 2.7182818284590451
exptaylor = 2.7182818284590455
error = 4.4408920985006262E-016
Es la diferencia entre el valor "verdadero" de la funcion y el valor usando la serie de taylor en determinado punto.

2 Problema 2: Exponencial

```
SUBROUTINE exptaylor (n, j, fi, fj, exptay)
integer, intent (IN)      :: n
double precision, intent (IN) :: fi
integer :: j
double precision, dimension (100), intent(OUT) :: exptay
double precision :: fj, term, partial_sum



term = 1
partial_sum = term
DO j = 1, n
 fj = dble(j)
 term = term * fi / fj
 partial_sum = partial_sum + term
 exptay(j) = partial_sum
END DO


END SUBROUTINE exptaylor


PROGRAM exponencial
double precision, dimension (15) :: f
integer :: i, j, n
double precision, dimension (100)   :: x
double precision, dimension (100) :: exptay
double precision, dimension (100) :: funcion
double precision :: fi, fj, term, partial_sum

     OPEN (1, FILE = 'exp.dat', STATUS = 'unknown')

DO n=1, 15, 2
DO i=0, 100, 1
  fi = dble(i)
  fi = fi / 10.0d0
CALL exptaylor (n, j, fi, fj, exptay)
funcion(n) = exptay(n)
WRITE(1,*) fi, funcion(n)
END DO
WRITE (1,*) ' '
END DO
     CLOSE (1)
END PROGRAM exponencial
```

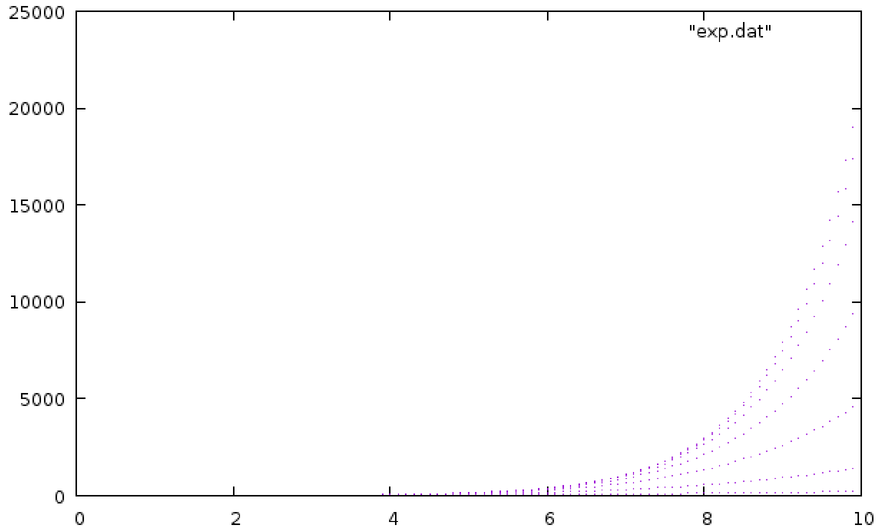

Figure 1: Grafica de exp(x)

3 Problema 3: Seno

```
SUBROUTINE seno (n, j, fi, fj, sen, signo, potencia, factorial)
integer, intent (IN)      :: n
double precision, intent (IN) :: fi
integer :: j
double precision, dimension (10000), intent(OUT) :: sen
double precision :: fj, term, partial_sum, signo, potencia, factorial


signo = 1.0d0
term = fi
partial_sum = term
potencia = fi
factorial = 1
DO j = 1, n
 fj = dble(j)
 potencia = fi**(j + 2)
 factorial = factorial * (j + 1) * (j + 2)
 signo = signo * (-1.0d0)
 term = potencia / factorial
 term = term * signo
 partial_sum = partial_sum + term
 sen(j) = partial_sum

END DO


END SUBROUTINE seno


PROGRAM senos
double precision, dimension (10000) :: f
integer :: i, j, n
double precision, dimension (10000)   :: x
double precision, dimension (10000) :: sen
double precision, dimension (10000) :: funcion
double precision :: fi, fj, term, partial_sum, signo, potencia, factorial


     OPEN (1, FILE = 'funciones.dat', STATUS = 'unknown')
fi = -3.1d0
DO i=1, 60
WRITE (1,*) fi, fi
fi = fi + 0.1d0
```

```
END DO
WRITE (1,*) ' '
DO n=1, 15, 2
  fi = -3.1d0
DO i=1, 60
fi = fi + 0.1d0
CALL seno (n, j, fi, fj, sen, signo, potencia, factorial)
funcion(n) = sen(n)
WRITE (1,*) fi, funcion(n)

END DO
WRITE (1,*) ' '
END DO
     CLOSE (1)

END PROGRAM senos
```

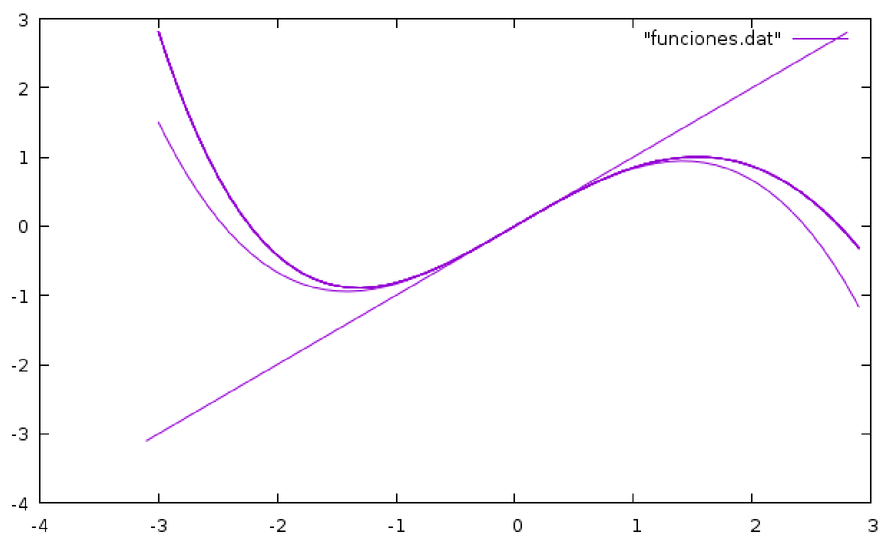

Figure 2: Series de Taylor Sen(X)