Evaluación 2

César Andrés Pérez Robinson

December 8, 2017

1 Serie de MacLaurin

La serie de Taylor es una aproximación de funciones mediante una serie de potencias o suma de potencias enteras de polinomios llamados términos de la serie, dicha suma se calcula a partir de las derivadas de la función para un determinado valor. Si esta serie está centrada sobre el punto cero, se le denomina serie de McLaurin.

2 Código

El siguiente código usa una funciona para la Serie de McLaurin para aproximar la función exponencial en el punto x=1, utilizando n=20 términos.

```
! ----------- Begin ------------

!taylor.f90

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
        enddo
     exptaylor = partial_sum  ! this is the value returned
end function exptaylor

! --------  End -------------
```

2.1 Resultados

```
x =    1.0000000000000000
 exp_true  =    2.7182818284590451
 exptaylor =    2.7182818284590455
 error     =    4.4408920985006262E-016
```

2.2 Exponencial

El siguiente código fue utilizado para realizar la gráfica 1, la cuál aproxíma la funcion exponencial usando los términos 1,3,5,...,15.

```
subroutine sub1(fi, y, n, res)
  implicit none
```

```
  real (kind=8), dimension(100), intent(in) :: fi
  real (kind=8), dimension(100), intent(out) :: y, res
  integer, intent(in) :: n

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


  term = 1.0
  partial_sum = 0.0


  do j = 1, 100
     term = term * fi(j) / float(j)
     partial_sum = partial_sum + term
  end do



end subroutine sub1


program examen
  implicit none

  real (kind=8) :: exp_true, term, exptaylor
  integer :: n
  integer :: i, npts
  real (kind=8), dimension(100) :: fi, y
  real (kind=8), dimension(100) :: res



  open(unit=3, file="repo.dat", status="unknown")

  npts = 100
  term = 1.0
    do i = 1, 100
     fi(i) = float(i) / 10.0
     write(3,*) fi(i), term
  end do

  write(3,*) ' '
```

```
  do n = 1, 15, 2

     do i = 1, 100
        !call exptaylor(fi(i),n)
        y(i) = exptaylor(fi(i), n)
     end do


     do i= 1, npts
        write(3,*) fi(i), y(i)
     end do
     write(3,*) ' '

end do


  close(unit=3)

  end program examen


  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
        enddo
     exptaylor = partial_sum  ! this is the value returned
end function exptaylor
```

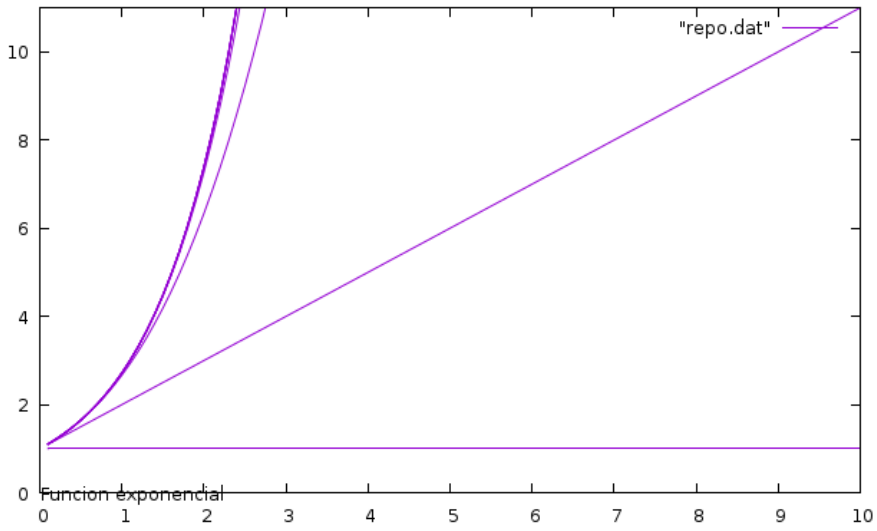

Figure 1: Exponencial

```
! --------  End --------------
```

3 Gráfica