Evaluación 2

César Andrés Pérez Robinson

December 1, 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 Intento de código

```
subroutine e(x, y, npts, sum)
  implicit none
  real (kind=8), dimension(100), intent(in) :: x
  real (kind=8), dimension(100), intent(out) :: y
  integer (kind=8), intent(in) :: npts
  integer (kind=8) , intent(out) :: sum
```

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

  term = 1.0
  partial_sum = term

  do j = 1, sum
     term = term * x(j) / float(j)
     partial_sum = partial_sum + term
     y(j) = partial_sum
  end do



  end subroutine e

program sub

  implicit none
  real (kind=8) :: exp_true, fj, fi
  integer (kind=8) :: sum, npts
  integer :: j, i
  real (kind=8), dimension(100) :: x
  real (kind=8), dimension(100) :: y
  integer, dimension(15) :: n
  real, dimension(100,15) :: sum

  open(unit=2, file="exponencial.dat", status="unknown")

  npts = 100

  do j= 1, 15, 2
     n(j) = j
  end do


  do i = 0, npts
     fi = float(i)
     x(i) = fi /10.0
  end do

     call e(x, y, npts, sum)
     exp_true = exp(x(i))
     print *, "x = ", x
```

```
    print *, "exp_true = ",exp_true
    print *, "exptaylor = ",y
    print *, "error  = ",y - exp_true

    do i = 1, npts

       write(2,*) x(i), y(i)

  end do

  write(2,*) ' '




  close(unit=2)

end program sub
```

El código anterior fue el avance realizado para la actividad dos. Esta no fue completada.