

INFORMATICA

Antes:

- Dudas sobre el examen:
 - Ecuación de segundo grado
 - Sumatorios
 - Vectores
- Clase de hoy:
 - Funciones intrínsecas
 - Ejercicio polinomio de Taylor
 - Matemáticas
 - Programación

Operaciones con vectores y matrices

```
program producto_matriz
```

```
integer, allocatable :: A(:, :)  
integer :: i, k
```

```
allocate (A(3,5))
```

```
A = 0
```

```
k = 7
```

```
do i = 1, 3  
    A(i,i) = 1
```

```
enddo
```

```
A = k*A
```

```
do i = 1, 3  
    write(*,*) A(i, :)
```

```
enddo
```

```
end program
```

iRecordatorio!



7	0	0	0	0
0	7	0	0	0
0	0	7	0	0

Con esto escribimos una matriz por filas (una fila por cada valor de i)

Funciones intrínsecas.

- Son funciones predefinidas que se pueden utilizar sin declarar previamente.
- Las funciones pueden tener argumentos de tipo:
 - Numérico:
 - Conversión de tipo
 - Truncamiento
 - Redondeo
 - Matemáticas
 - Array
 - Matemáticas
 - Búsqueda
 - Dimensiones
 - Character

Funciones intrínsecas.

- Funciones numéricas. Conversión de tipo
 - **Real**(x[,k]) → convierte x a `real(kind=k)`
 - Tipo del argumento (x): `integer`, `real` o `complex`
 - Tipo de la función: `real`
 - **dbble**(x) → convierte x a `real*8`
 - Tipo del argumento (x): `integer`, `real` o `complex`
 - Tipo de la función: `real*8`

(Los corchetes significan opcional)

Funciones intrínsecas.

- Funciones numéricas. Conversión de tipo

```
program prueba_intrinsecas
```

```
integer :: a  
real(8) :: xd  
real :: x
```

```
a = 2
```

```
x = real(a)
```

```
xd = real(a,KIND=8)
```

```
print*, "a", a  
print*, "x", x  
print*, "xd", xd
```

```
end program
```

a	2
x	2.00000000
xd	2.000000000000000000

Funciones intrínsecas.

- Funciones numéricas. Conversión de tipo

• **aimag** (z)

Parte imaginaria del número complejo z

- Tipo del argumento: **complex**
- Tipo de la función: **real**

• **cmplx** (x [, y] [, k])

Número complejo de parte real x e imaginaria y y kind=k

- Tipo del argumento: **real** o **complex**
- Tipo de la función: **complex**

Funciones intrínsecas.

- Funciones numéricas. Conversión de tipo

```
real      :: x      →  cmplx(x)  = x + i0.0
```

```
real      :: x, y    →  cmplx(x,y) = x + iy
```

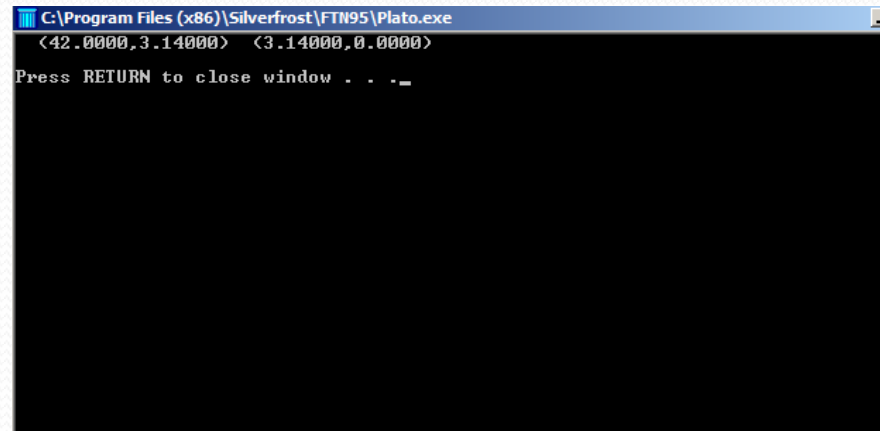
```
complex   :: z      →  cmplx(z)  =  $Re(z) + iIm(z)$ 
```

```
program test_cmplx

  integer :: i = 42
  real    :: x = 3.14
  complex :: z

  z = cmplx(i, x)
  write(*,*) z, cmplx(x)

end program test_cmplx
```



Funciones intrínsecas.

- Funciones numéricas. Truncamiento y redondeo

x	<code>aint(x)</code>	<code>anint(x)</code>	<code>int(x)</code>	<code>nint(x)</code>
± 5.0	± 5.0	± 5.0	± 5	± 5
± 5.1	± 5.0	± 5.0	± 5	± 5
± 5.2	± 5.0	± 5.0	± 5	± 5
± 5.3	± 5.0	± 5.0	± 5	± 5
± 5.4	± 5.0	± 5.0	± 5	± 5
± 5.5	± 5.0	± 6.0	± 5	± 6
± 5.6	± 5.0	± 6.0	± 5	± 6
± 5.7	± 5.0	± 6.0	± 5	± 6
± 5.8	± 5.0	± 6.0	± 5	± 6
± 5.9	± 5.0	± 6.0	± 5	± 6

Funciones intrínsecas.

- Funciones numéricas. Matemáticas

<code>sqrt(x)</code>	<code>log(x)</code>
<code>conjg(z)</code>	<code>log10(x)</code>
<code>max(x1,x2,...)</code>	<code>cos(x)</code>
<code>min(x1,x2,...)</code>	<code>sin(x)</code>
<code>exp(x)</code>	<code>tan(x)</code>

<code>acos(x)</code>	$0 \leq \arccos(x) \leq \pi$
<code>asin(x)</code>	$-\pi/2 \leq \arcsen(x) \leq \pi/2$
<code>atan(x)</code>	$-\pi/2 \leq \arctg(x) \leq \pi/2$
<code>atan2(x,y)</code>	$-\pi < \arctg(y/x) \leq \pi$

Funciones intrínsecas.

- Funciones numéricas. Matemáticas

- $\text{abs}(x)$
 - Si x es real $\text{abs}(x) = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x < 0 \end{cases}$

- Si x es complejo $\text{abs}(x) = \sqrt{X_r^2 + X_i^2}$

- $\text{mod}(x, y)$

- Ambos x e y pueden ser reales o enteros

$$\text{mod}(x, y) = x - \text{int}(x/y) \cdot y$$

Funciones intrínsecas.

- Funciones numéricas. Matemáticas.

- `epsilon(x)`

- Devuelve un número del mismo tipo y kind que `x`, que es despreciable frente a la unidad.

$$1.d0 \sim 1.d0 \pm \text{epsilon}(x)$$

Funciones intrínsecas.

- Funciones para arrays. Búsqueda
 - `size(A,[dim=i])` (Los corchetes significan opcional)
 - Devuelve el tamaño de la dimensión i del array A
 - Ej.- `size(V)` devuelve el numero de elementos del vector V
 - Ej.- `size(Matriz,dim=1)` devuelve el número de filas del array Matriz
 - Ej.- `size(Matriz,dim=2)` devuelve el número de columnas del array Matriz

Funciones intrínsecas.

- Funciones para arrays. Búsqueda
 - `maxval(Array)/minval(Array)`
 - Devuelve el mayor/menor elemento del array.
 - `maxval(Matriz,dim = i)/minval(Matriz,dim = i)`
 - `i=1` Devuelve un vector cuyos elementos son el máximo de cada una de las columnas.
 - `i=2` Devuelve un vector cuyos elementos son el máximo de cada una de las filas.
 - Ej.-
 - $A = \begin{pmatrix} 3 & 2 & 1 \\ 4 & -1 & 6 \end{pmatrix}$ `maxval(A,dim = 2) ! (3,6)`

Funciones intrínsecas.

- Funciones para arrays. Búsqueda
 - `maxloc(Array)/minloc(Array)`
 - Devuelve la posición (no el índice) del mayor/menor elemento del array.
 - Ej. $V = \begin{pmatrix} 3 & -1 & 2 \end{pmatrix}$ `minloc(V)` ! 2
 - Ej. $A = \begin{pmatrix} 3 & 2 & 1 \\ 4 & -1 & 6 \end{pmatrix}$ `maxloc(A)` ! (2,3)
 - Ej. $A = \begin{pmatrix} 3 & 2 & -1 \\ 4 & -1 & 6 \end{pmatrix}$ `minloc(A)` ! (2,2)

Funciones intrínsecas.

- Funciones para arrays. Matemáticas
 - `transpose(A)`
 - Devuelve la matriz transpuesta.
 - Ej.- $A = \begin{pmatrix} 3 & 2 & 1 \\ 4 & -1 & 6 \end{pmatrix}$
 - $B = \text{transpose}(A)$ $B = \begin{pmatrix} 3 & 4 \\ 2 & -1 \\ -1 & 6 \end{pmatrix}$

Funciones intrínsecas.

- Funciones para arrays. Matemáticas
 - `dot_product(vector_1,vector_2)`
 - Devuelve el producto escalar de los dos vectores.
 - `matmul(A,B)`
 - Devuelve el producto de dos matrices(arrays) con formas compatibles.

Funciones intrínsecas.

```
program producto
```

```
integer      :: V(3), U(3)
```

```
integer      :: A(3,2), B(2,3), C(3,3)
```

```
integer      :: i,j
```

```
A = 0
```

```
B = 0
```

```
do i = 1,2
```

```
    A(i,i) = 1
```

```
        do j = 1,3
```

```
            B(i,j) = j
```

```
        enddo
```

```
enddo
```

```
C = matmul(A,B)
```

```
do j = 1,3
```

```
    write(*,*) C(j,:)
```

```
enddo
```

```
end program producto
```

Funciones intrínsecas.

```
program producto
```

```
integer      :: V(3), U(3)
```

```
integer      :: A(3,2), B(2,3), C(3,3)
```

```
integer      :: i,j
```

```
A = 0
```

```
B = 0
```

```
do i = 1,2
```

```
    A(i,i) = 1
```

```
        do j = 1,3
```

```
            B(i,j) = j
```

```
        enddo
```

```
enddo
```

```
C = matmul(A,B)
```

```
do j = 1,3
```

```
    write(*,*) C(j,:)
```

```
enddo
```

```
end program producto
```

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 0 & 0 & 0 \end{pmatrix}$$

Funciones intrínsecas.

```
program producto

integer    :: V(2), U(3)
integer    :: A(3,2)
integer    :: i

do i =1,3
    U(i) = i
enddo

A = 1
V = matmul(U,A)

write(*,*) V

end program producto
```

Funciones intrínsecas.

```
program producto
```

```
integer    :: V(2), U(3)
```

```
integer    :: A(3,2)
```

```
integer    :: i
```

```
do i =1,3
```

```
    U(i) = i
```

```
enddo
```

```
A = 1
```

```
V = matmul(U,A)
```

```
write(*,*) V
```

```
end program producto
```

$$U = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \quad A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$V = \begin{pmatrix} 6 & 6 \end{pmatrix}$$

Funciones intrínsecas.

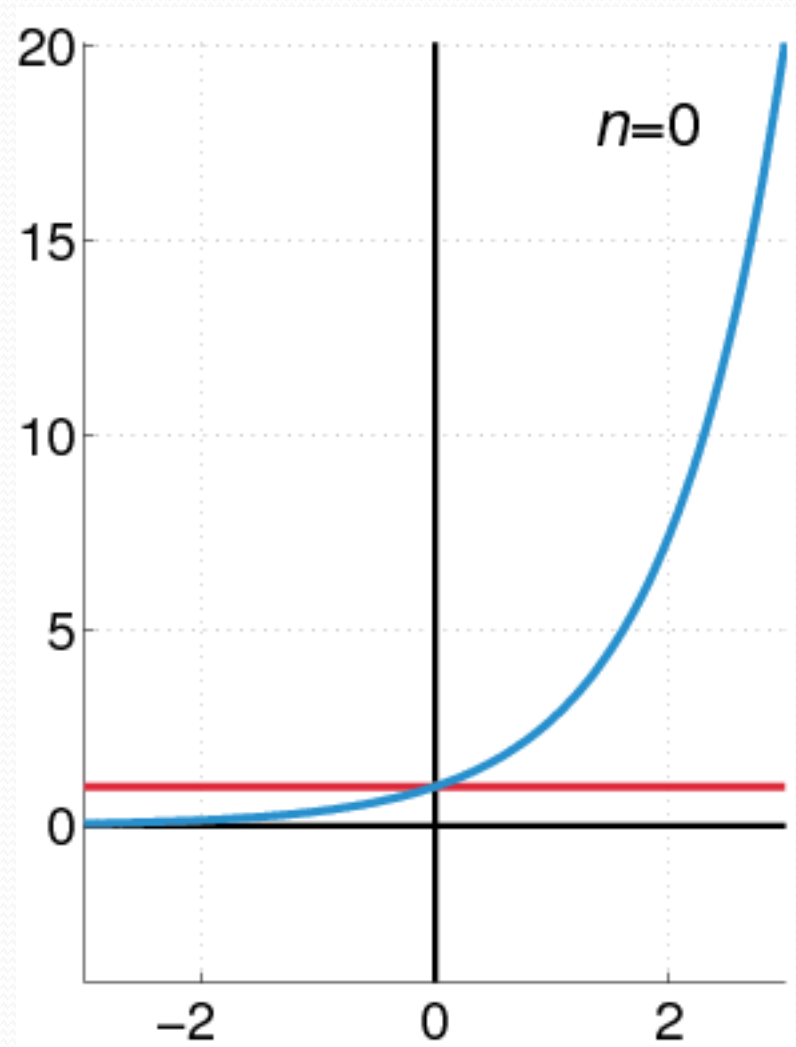
- Funciones para Character.
 - `len(string)`
 - Devuelve la longitud del string (cadena).
 - Ej.- `len(' Fortran & magic ') = 20`
 - `trim(string)`
 - Devuelve el mismo string quitando los espacios en blanco a la derecha del mismo.
 - Ej.- `trim (' Fortran & magic ') =
 ' Fortran & magic'`

Antes:

- Dudas sobre el examen:
 - Ecuación de segundo grado
 - Sumatorios
 - Vectores
- Clase de hoy:
 - Funciones intrínsecas
 - Ejercicio polinomio de Taylor
 - Matemáticas
 - Programación

Caso de aplicación:

- Serie de Taylor de una función.



Serie de Taylor de una función

Buscamos el polinomio que "más se parece" a una función en el entorno del punto $x = 0$

$$f(x) \approx P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_Nx^N = \sum_{i=0}^N a_i x^i$$

Que "se parezcan" implica que al menos su valor sea el mismo en $x = 0$:

$$f(0) = P(0) = a_0 \Rightarrow P(x) = f(0) \text{ Polinomio aproximador de orden 1.}$$

Serie de Taylor de una función

Buscamos el polinomio que "más se parece" a una función en el entorno del punto $x = 0$

$$f(x) \approx P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_Nx^N = \sum_{i=0}^N a_i x^i$$

Que "se parezcan" implica que al menos su valor sea el mismo en $x = 0$:

$$f(0) = P(0) = a_0 \Rightarrow P(x) = f(0) \text{ Polinomio aproximador de orden 0.}$$

Ejemplo:

$$f(x) = \frac{1}{1-x} \quad \rightarrow \quad \text{¿}P(x)\text{?}$$

Orden 0:

$$\boxed{f(0) = 1 \Rightarrow P(x) = 1}$$

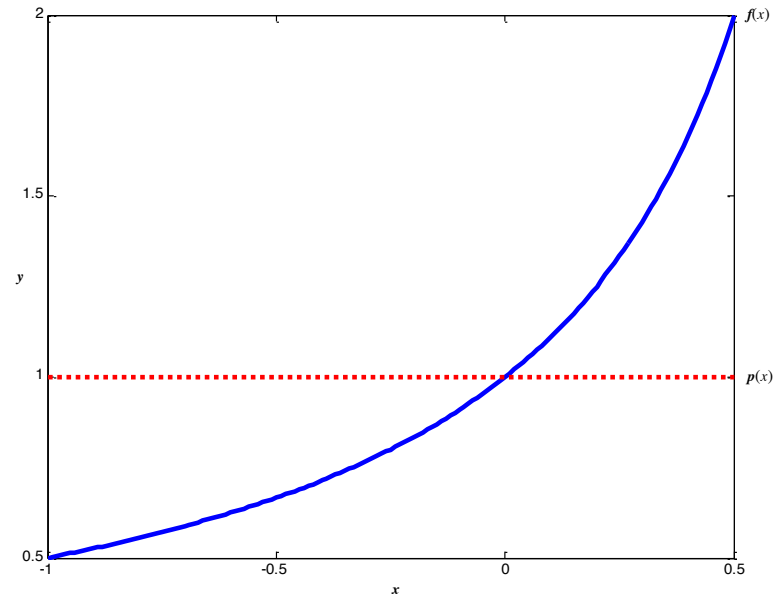
Serie de Taylor de una función

Ejemplo:

$$f(x) = \frac{1}{1-x} \rightarrow \text{¿}P(x)\text{?}$$

Orden 1:

$$\boxed{f(0)=1 \Rightarrow P(x)=1}$$



Que "se parezcan" implica que su pendiente (crecimiento) también coincida en $x = 0$:

$$f'(0) = P'(0) = a_1 \Rightarrow (\text{ya sabíamos que } a_0 = f(0))$$

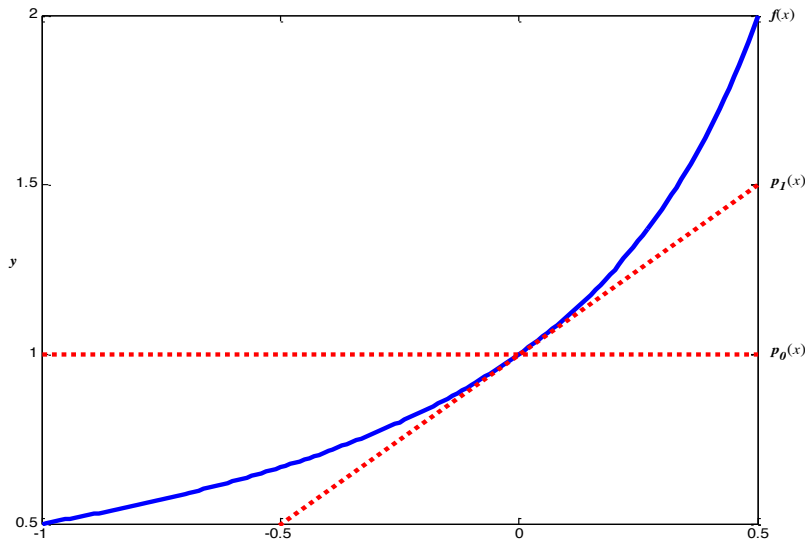
$$\Rightarrow P(x) = f(0) + f'(0)x$$

$$\text{Volviendo al ejemplo: } f'(x) = \frac{1}{(1-x)^2} \rightarrow a_1 = f'(0) = 1$$

Orden 1:

$$\boxed{P(x) = 1 + x}$$

Serie de Taylor de una función



Orden 1:

$$P(x) = 1 + x$$

Si seguimos forzando la igualdad en derivadas sucesivas en $x = 0$:

$$f''(0) = P''(0) = 2 \cdot a_2 \rightarrow a_2 = \frac{f''(0)}{2}$$

$$f'''(0) = P'''(0) = 3 \cdot 2 \cdot a_3 \rightarrow a_3 = \frac{f'''(0)}{3!}$$

$$f^{(4)}(0) = P^{(4)}(0) = 4 \cdot 3 \cdot 2 \cdot a_4 \rightarrow a_4 = \frac{f^{(4)}(0)}{4!}$$

.....

$$f^n(0) = P^n(0) = n \cdot (n-1) \cdot \dots \cdot 3 \cdot 2 \cdot a_n \rightarrow a_n = \frac{f^n(0)}{n!}$$

Serie de Taylor de una función

$$f(x) \approx P(x) = f(0) + \frac{f'(0)}{1}x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \dots + \frac{f^N(0)}{N!}x^N = \sum_{i=0}^N \frac{f^i(0)}{i!}x^i$$

En nuestro caso particular de $f(x) = \frac{1}{1-x}$ en un entorno de $x = 0$

$$f''(x) = \frac{2(1-x)}{(1-x)^4} = \frac{2}{(1-x)^3} \Rightarrow f''(0) = 2$$

$$f'''(x) = \frac{3 \cdot 2 \cdot (1-x)^2}{(1-x)^6} = \frac{3 \cdot 2}{(1-x)^4} \Rightarrow f'''(0) = 3 \cdot 2$$

⋮

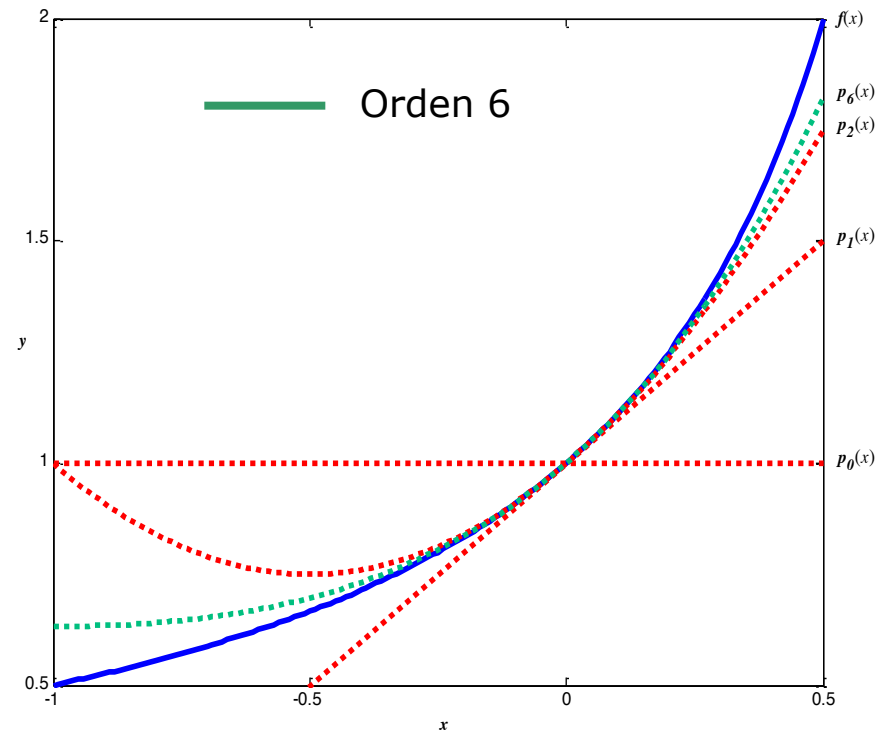
$$f'^n(x) = \frac{4 \cdot 3 \cdot 2}{(1-x)^{n+1}} \Rightarrow f'^n(0) = n!$$

$$f(x) = \frac{1}{1-x} = \sum_{n=0}^{\infty} \frac{n!}{n!} x^n = \sum_{n=0}^{\infty} x^n$$

Serie de Taylor de una función

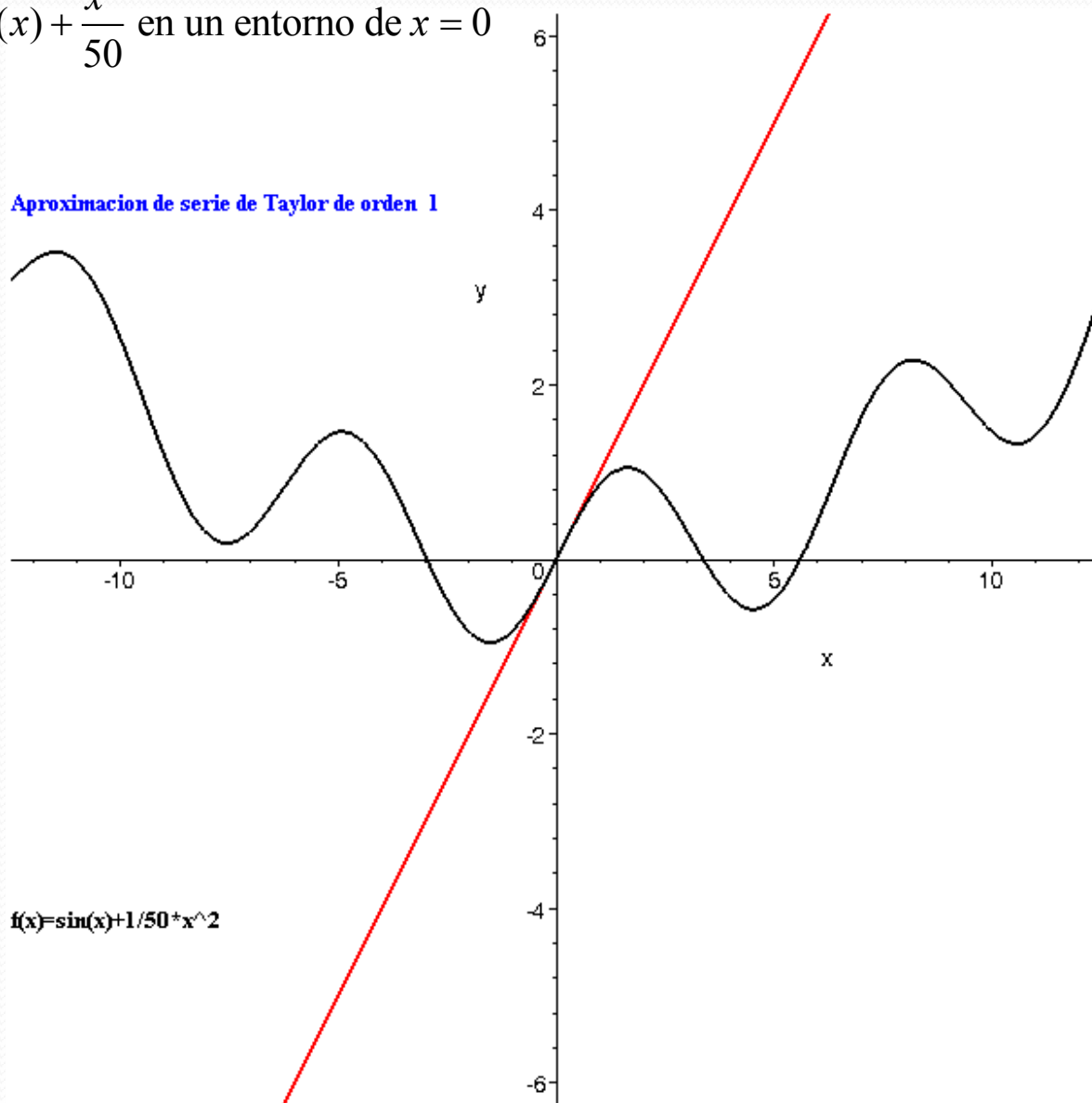
$$f(x) \approx P(x) = f(0) + \frac{f'(0)}{1}x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \dots + \frac{f^N(0)}{N!}x^N = \sum_{i=0}^N \frac{f^i(0)}{i!}x^i$$

En nuestro caso particular: $f(x) = \frac{1}{1-x} = \sum_{n=0}^{\infty} \frac{n!}{n!}x^n = \sum_{n=0}^{\infty} x^n$



Serie de Taylor de una función

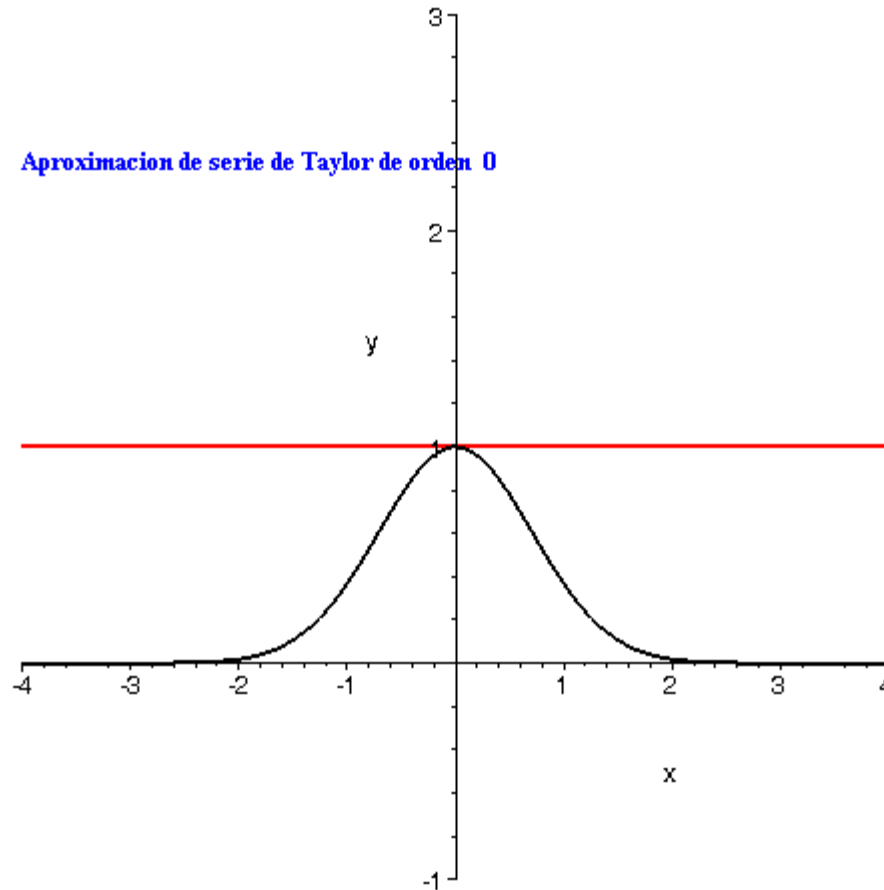
Algunos ejemplos $f(x) = \sin(x) + \frac{x^2}{50}$ en un entorno de $x = 0$



Serie de Taylor de una función

$f(x) = e^{-x^2}$ en un entorno de $x = 0$

Aproximacion de serie de Taylor de orden 0



Serie de Taylor: PROGRAMACIÓN

Buscamos el polinomio que "más se parece" a una función en el entorno del punto $x = 0$

$$f(x) = \ln(1+x) \approx S_n(x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} \dots + (-1)^{n+1} \frac{x^n}{n} + \dots = \sum_{i=0}^N (-1)^{i+1} \frac{x^i}{i}$$

```
Sn = 0.d0  
do i = 1,n
```

```
    Sn = Sn + ((-1)**(i+1)) * ((x**i)/(1.d0*i))
```

```
enddo
```

```
!Sn almacena el valor del polinomio de grado n en x
```

```
error = abs(log(1.d0+x)-Sn)
```

Serie de Taylor: PROGRAMACION

```
...
do n = 1, 15

    Sn = 0.d0
    do i = 1,n

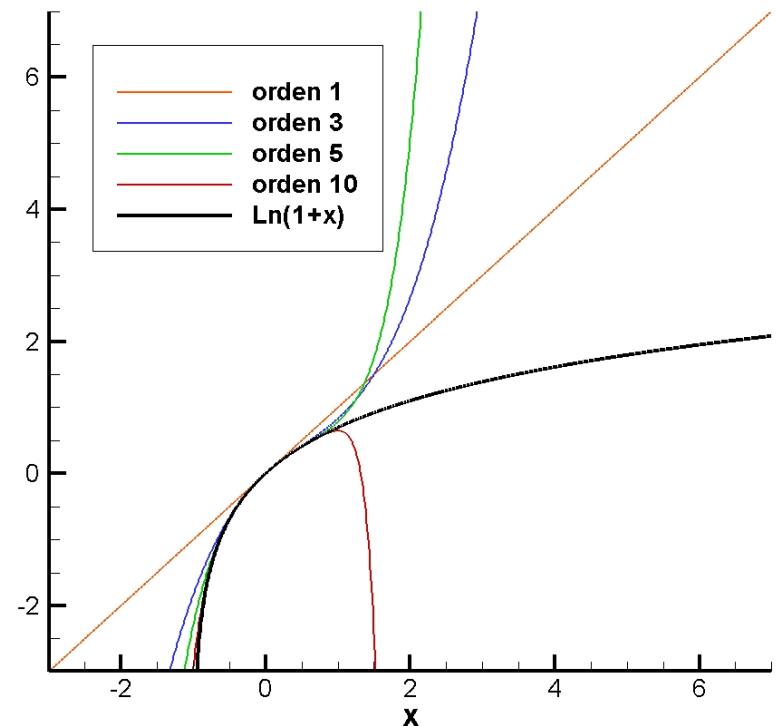
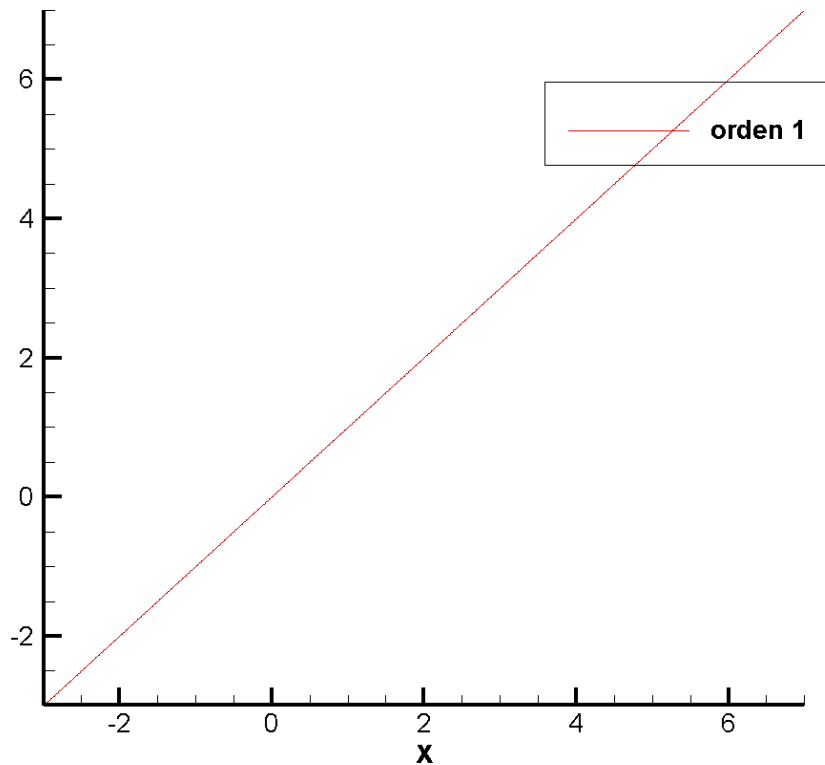
        Sn = Sn + ((-1)**(i+1)) * ((x**i)/(1.d0*i))

    enddo
    !Sn almacena el valor del polinomio de grado n en x

    error = abs(log(1.d0+x)-Sn)
    write(*,*) 'Orden: ',n, 'error: ',error

enddo
...
```

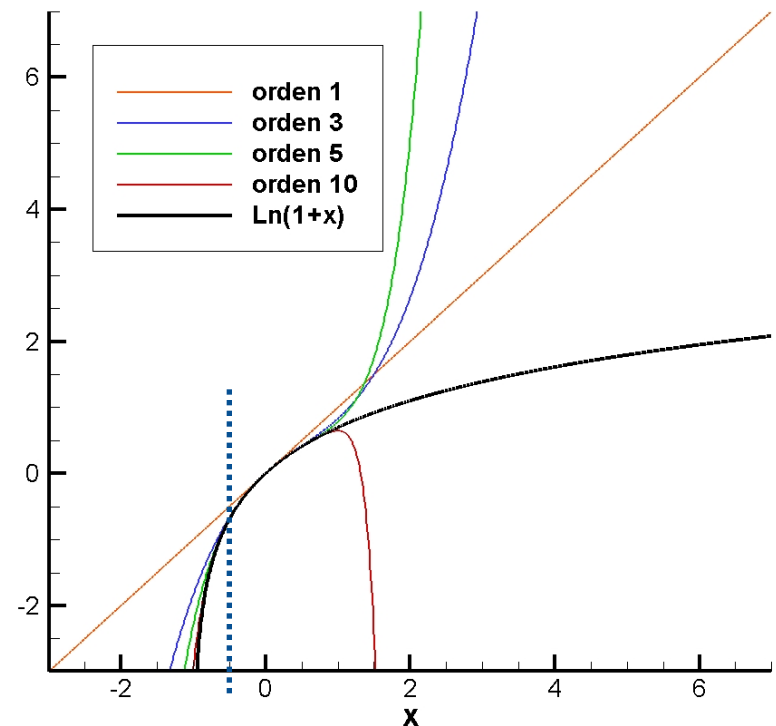
Serie de Taylor: PROGRAMACION



Serie de Taylor: PROGRAMACION

ERROR ENTRE $\ln(-0.5)$ y $P(-0.5)$

Orden 1	0.193147180560E-00
Orden 2	6.814718055995E-02
Orden 3	2.648051389328E-02
Orden 4	1.085551389328E-02
Orden 5	4.605513893279E-03
Orden 6	2.001347226612E-03
Orden 7	8.852757980407E-04
Orden 8	3.969945480407E-04
Orden 9	1.799806591518E-04
Orden 10	8.232440915181E-05
Orden 11	3.793520460640E-05
Orden 12	1.759015252303E-05
Orden 13	8.200128484593E-06
Orden 14	3.840474466689E-06
Orden 15	1.805969258385E-06
Orden 16	8.522949419786E-07
Orden 17	4.035070283691E-07
Orden 18	1.915794024764E-07
Orden 19	9.119263235757E-08
Orden 20	4.350891649285E-08



Serie de Taylor: PROGRAMACION

Volviendo a la notación compacta:

$$f(x) = \sum_{n=0}^N a_n x^n \quad a_n = \frac{f^{(n)}(0)}{n!}$$

Hay funciones cuyos desarrollos pueden escribirse de forma compacta

$$\begin{aligned} \sin x &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} &= x - \frac{x^3}{6} + \frac{x^5}{120} - \dots &\text{for all } x \\ \cos x &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} &= 1 - \frac{x^2}{2} + \frac{x^4}{24} - \dots &\text{for all } x \end{aligned}$$

Serie de Taylor: PROGRAMACION

Escribir un programa (versión básica) que:

- Pida por pantalla un entero N y un real x
- Calcule N términos a_n del desarrollo de Taylor de la función $\sin(x)$
- Evalúe el desarrollo de Taylor en el punto x
- Escriba por pantalla:
 - El el valor del desarrollo de Taylor en x.
 - El valor de la función en x.
 - El error cometido $|\text{Taylor}(x) - \sin(x)|$

$$f(x) = \sum_{n=0}^N a_n x^n$$

$$a_n = \frac{f^{(n)}(0)}{n!}$$

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Serie de Taylor: PROGRAMACION

Escribir un programa (versión avanzada) que:

- Pida por pantalla dos reales: x , tol y un entero: N .
- Calcule N términos a_n del desarrollo de Taylor de la función $\sin(x)$
- Evalúe el desarrollo de Taylor en el punto x , usando el número de términos necesarios para que el error sea menor que tol
- Escriba por pantalla:
 - El el valor del desarrollo de Taylor en x .
 - El valor de la función en x .
 - El error cometido $|\text{Taylor}(x) - \sin(x)|$
 - En caso de que el error cometido sea mayor que tol con el máximo número de términos (N), pedirle al usuario que utilice más términos.

$$f(x) = \sum_{k=0}^N a_k x^k$$

$$a_n = \frac{f^{(n)}(0)}{n!}$$

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$