Práctica 1 MÉTODOS NUMÉRICOS

G12: SANDRA GÓMEZ Y SERGIO CASADO

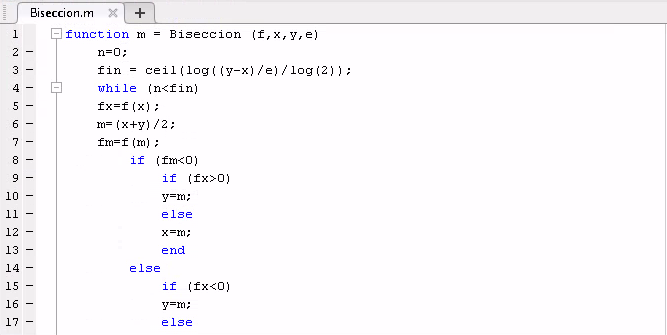
# PRÁCTICA 1

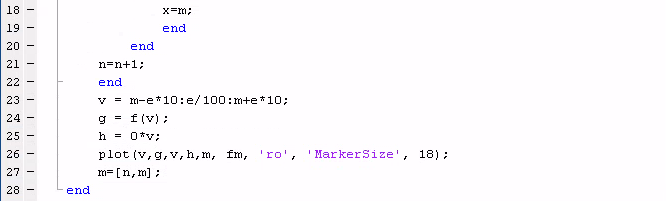
## PARTE 1: RAÍCES NUMÉRICAS DE FUNCIONES NO LINEALES

1. **Programar el método de bisección en Matlab**

*Biseccion* es una función a la que se introduce *f,x,y,e.* Siendo *f* la función f(x) con la que trabajaremos, *x* e *y* los puntos y respectivamente, y *e* la tolerancia.

Esta función nos devolverá *m=[n,m].* Siendo *n* el número de iteraciones y *m* la raíz.

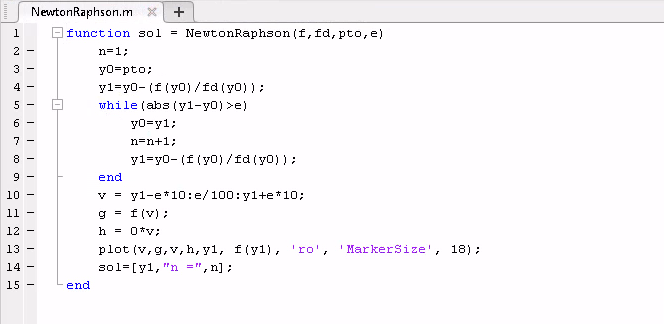




1. **Programar el método de Newton-Raphson en Matlab**

*NewtonRaphson* es una función a la que introduce *f, fd, pto, e.* Siendo *f* la función f(x) con la que trabajaremos, *fd* la derivada de f(x), *pto* es el punto , y *e* es la tolerancia.

Esta función nos devolverá *sol= [y1, “n =”, n]*, siendo *y1* la raíz y *n* el número de iteraciones.

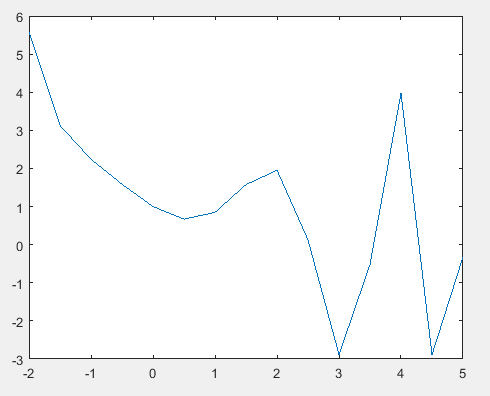


**Aplicar los programas desarrollados a la evaluación de la raíz positiva de la función de una variable real definida por .**

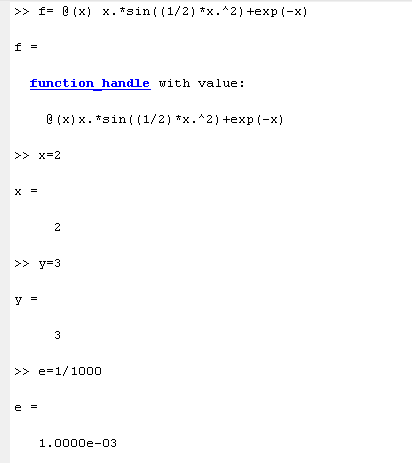
Bisección:

Dibujamos la gráfica por consola para evaluar la raíz positiva:

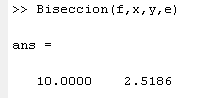


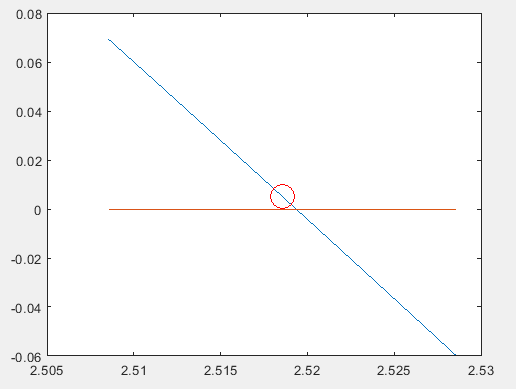


Elegimos los puntos = *2* y =*3*, que serán respectivamente nuestras nuevas *x e y* que introduciremos por pantalla, y elegiremos (por ejemplo) una tolerancia de 1/1000.

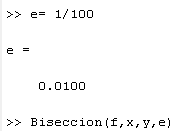


Que resulta:

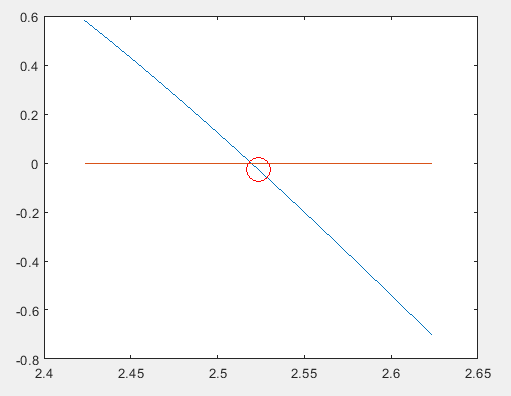


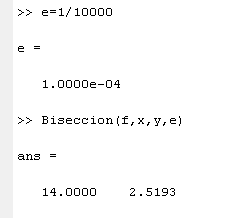


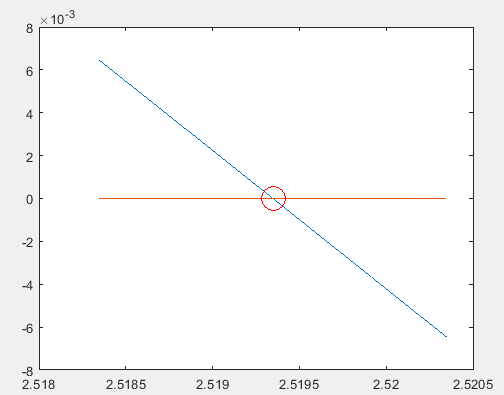
Cambiando *e* resulta:







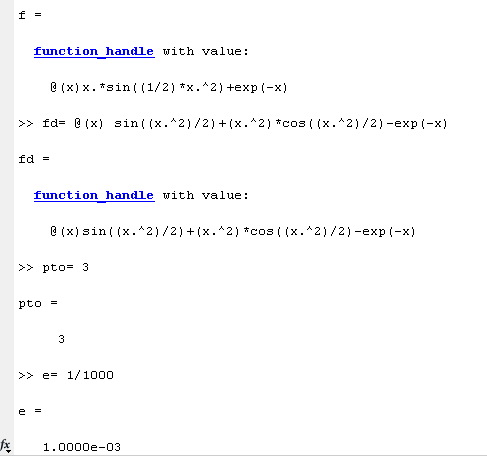




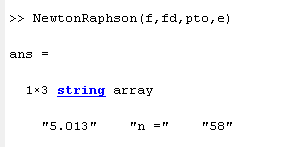
Esto ocurre porque al cambiar la cota de error, la aproximación cambia ajustándose más o menos a la raíz exacta.

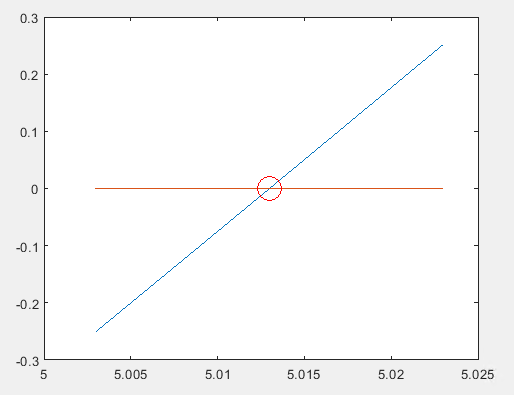
Newton - Raphson:

Elegimos el punto = 3. Y elegiremos (por ejemplo) una tolerancia de 1/1000.

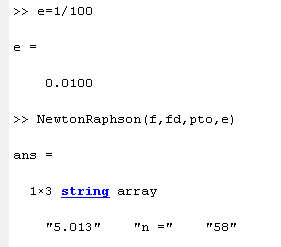


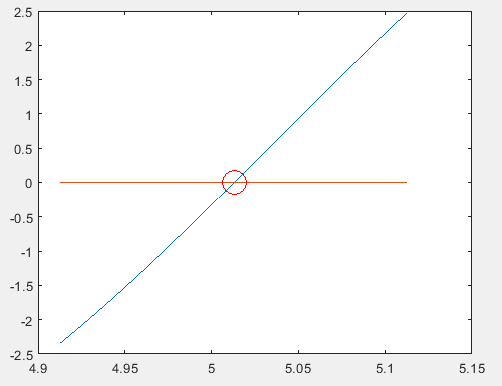
Que resulta:

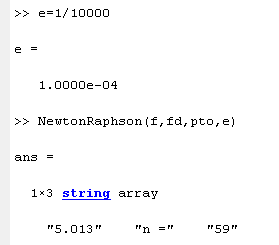


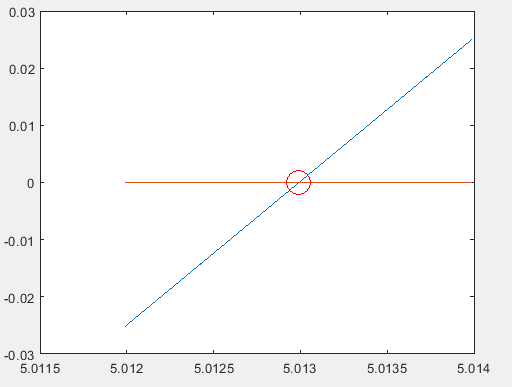


Cambiando *e* resulta:

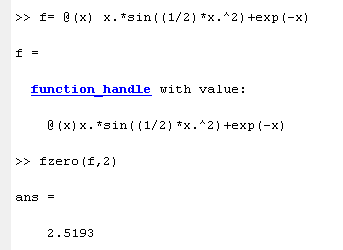






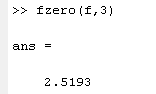


**Comparar los resultados con el resultado de referencia Rr obtenido empleando el comando fzero (help fzero).**



Mediante fzero con x=2 podemos observar como el método *Bisección* nos da el mismo valor de raíz.

Con x=3:



Por tanto, el método *Bisección* nos da una mejor aproximación a la raíz que el método *Newton-Raphson.*

## PARTE 2: RAÍCES NUMÉRICAS DE FUNCIONES NO LINEALES

**Aplicar los programas desarrollados anteriormente a los problemas siguientes.**

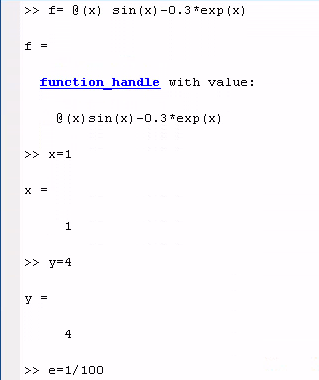
Utilizaremos los programas desarrollados en el apartado 1) y 2) anterior.

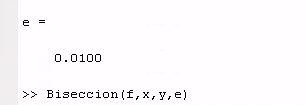
**2.1)**

**Raíz positiva >0,6 a partir de los puntos iniciales 1 y 4 (Bisección) y 1 (Newton-Raphson) con tolerancia ε=1/100.**

*Bisección*:

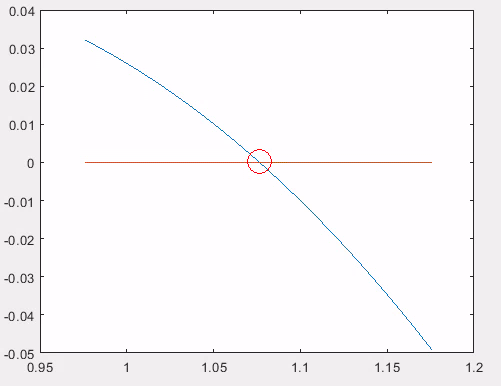
Por consola introducimos:





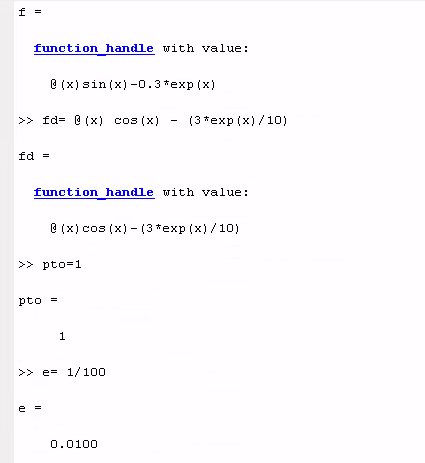
Que resulta:



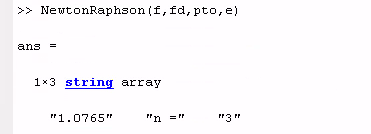


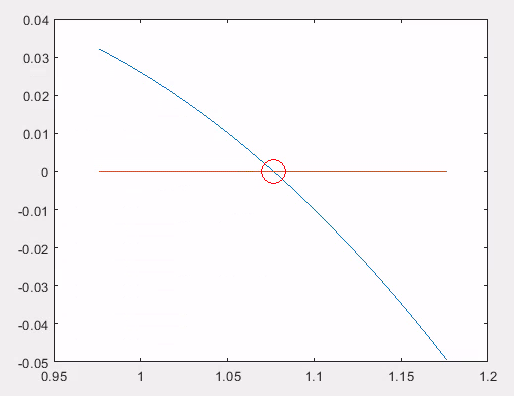
*Newton-Raphson:*

Por consola introducimos:



Que resulta:





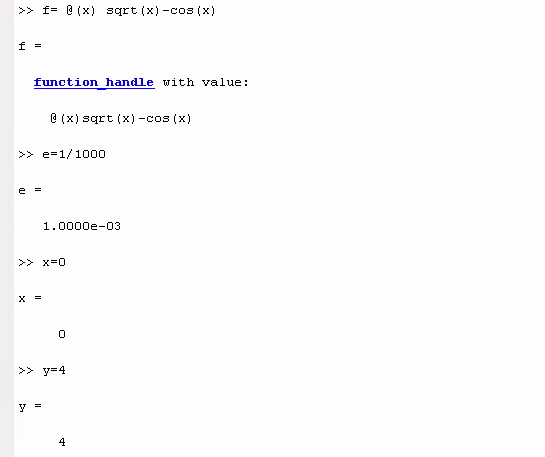
**2.2)**

**Raíz positiva con tolerancia ε=1/1000 a partir de los puntos iniciales: a) 0 y 4 (Bisección) y 1 (Newton-Raphson); b) 0,5 y 1 (Bisección) y 0,5 (Newton-Raphson).**

***a)***

*Bisección*:

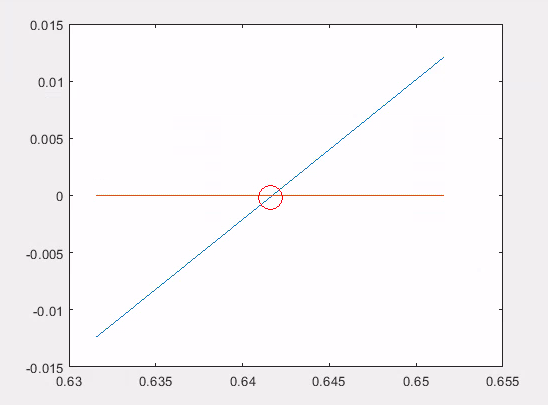
Por consola introducimos:





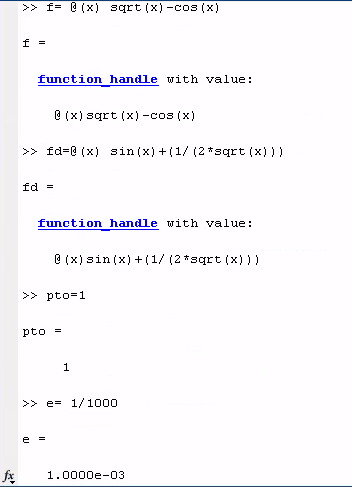
Que resulta:



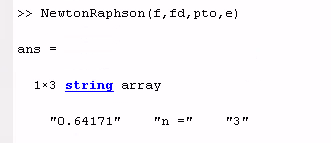


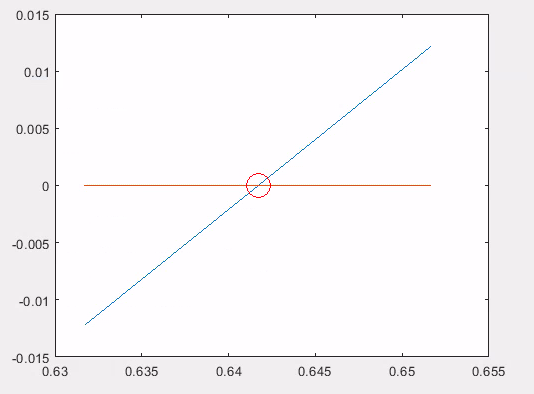
*Newton-Raphson:*

Por consola introducimos:



Que resulta:

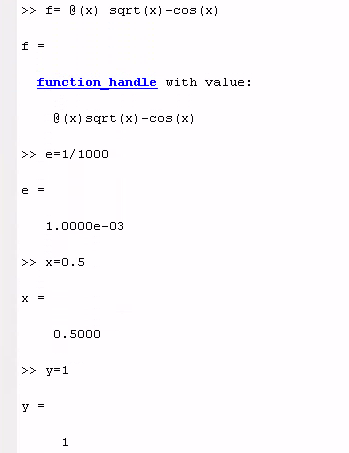




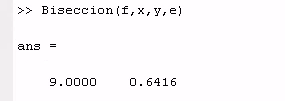
**b)**

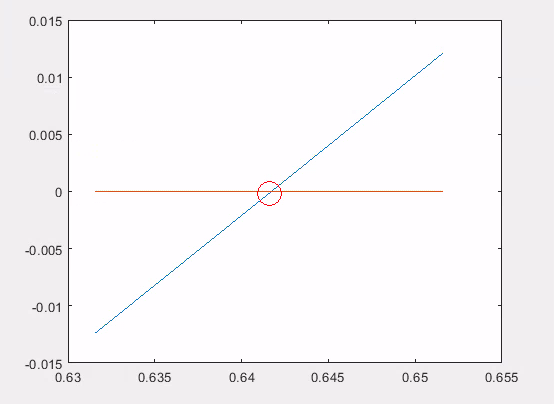
*Bisección*:

Por consola introducimos:



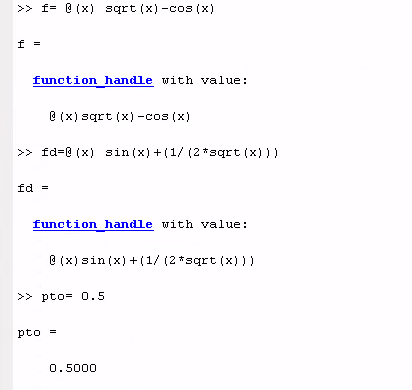
Que resulta:

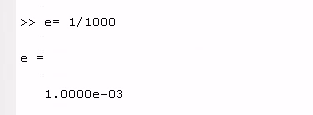




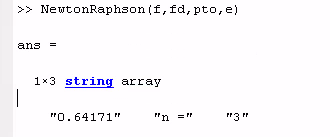
*Newton-Raphson:*

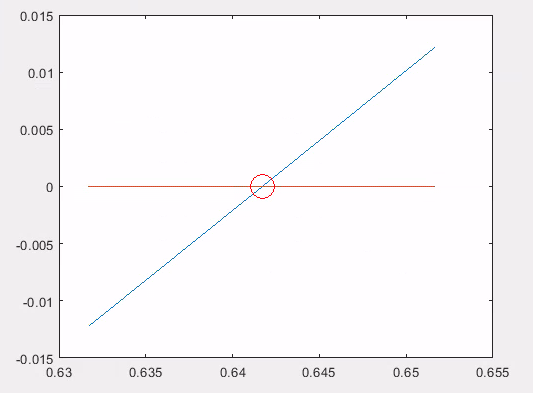
Por consola introducimos:





Que resulta:





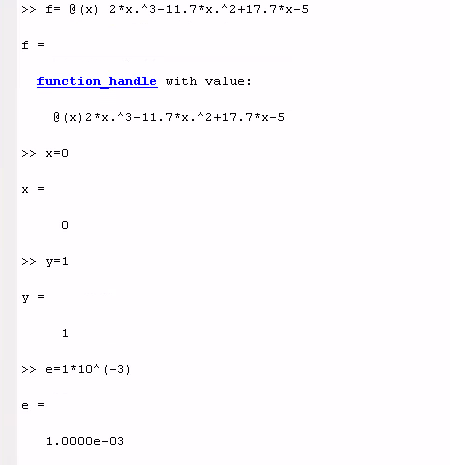
**2.3)**

**Raíces a partir de los puntos iniciales: a) 0 y 1, 1 y 3, 3 y 5 (Bisección con tolerancia ε=1×10^-3 ); b) 0,5 , 1,5 , 4 (Newton-Raphson con tolerancia ε=1×10^-10).**

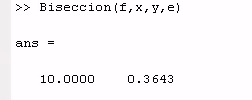
**a)**

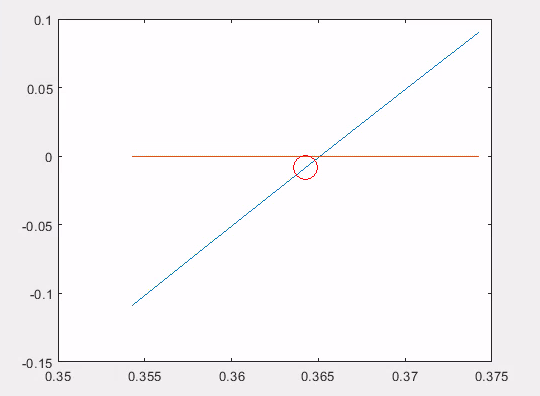
*Bisección 0 y 1*:

Por consola introducimos:



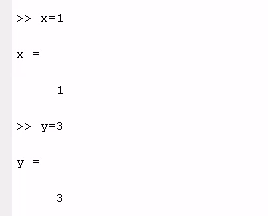
Que resulta:



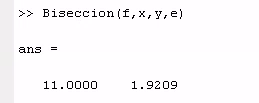


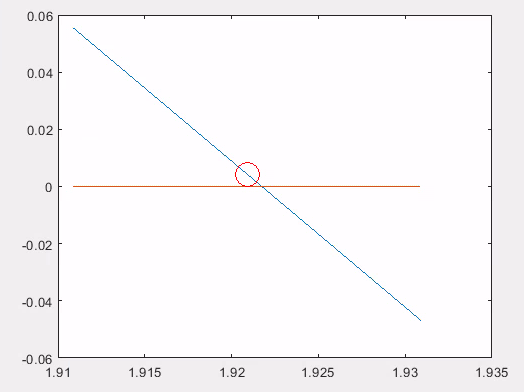
*Bisección 1 y 3*:

Por consola introducimos:



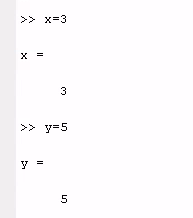
Que resulta:



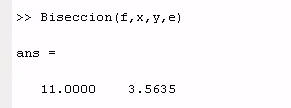


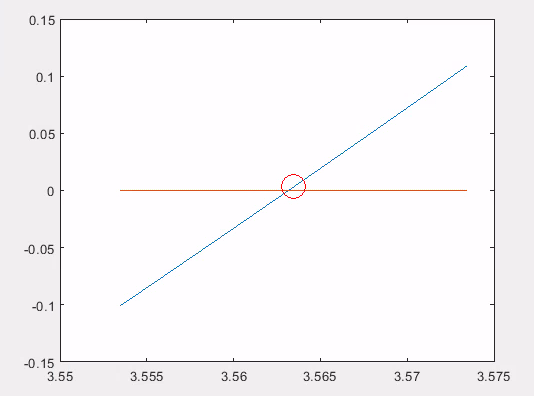
*Bisección 3 y 5*:

Por consola introducimos:



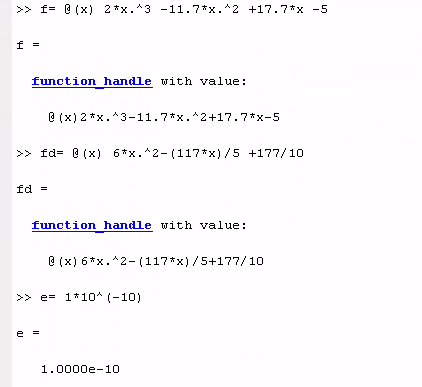
Que resulta:





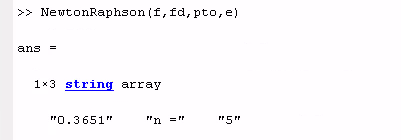
*Newton-Raphson 0.5:*

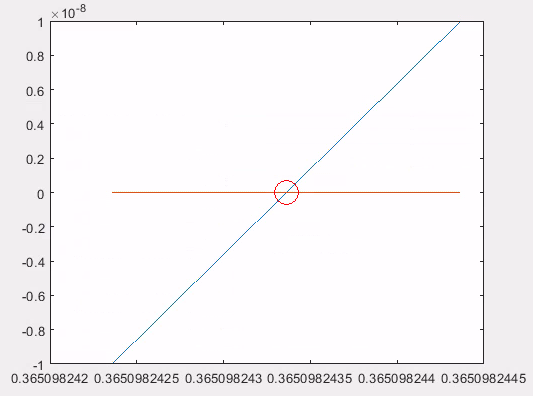
Por consola introducimos:





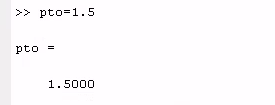
Que resulta:



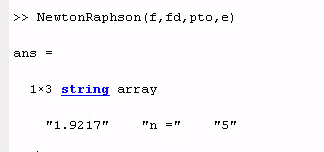


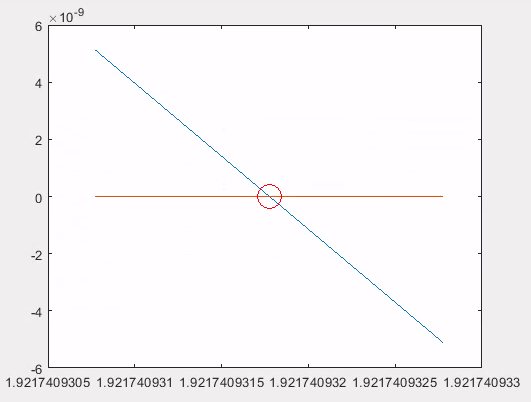
*Newton-Raphson 1.5:*

Por consola introducimos:



Que resulta:



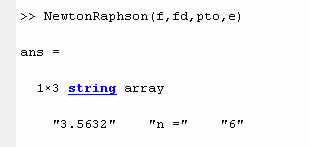


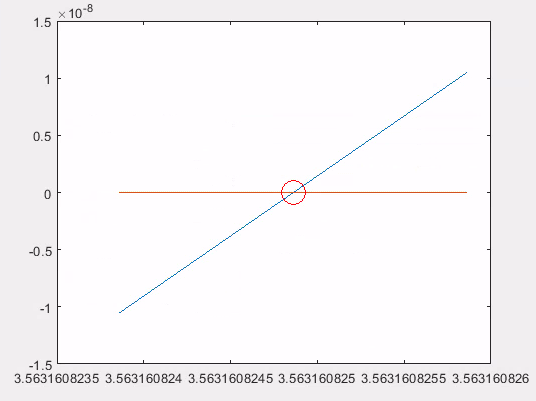
*Newton-Raphson 4:*

Por consola introducimos:



Que resulta:



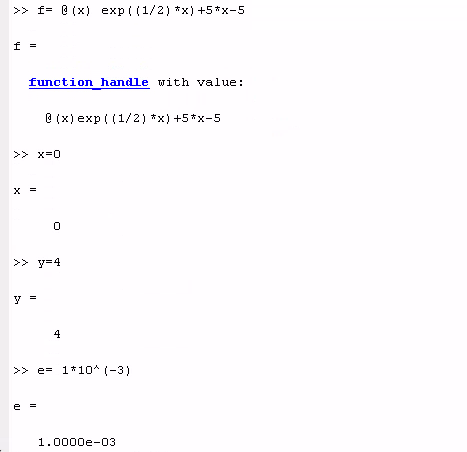


**2.4)**

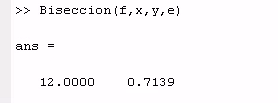
**Raíz con tolerancia ε=1×10^-3 a partir de los puntos iniciales: a) 0 y 4 (Bisección) y 1 (Newton-Raphson).**

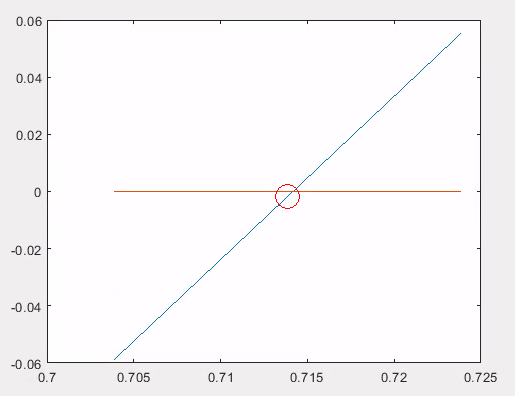
*Bisección*:

Por consola introducimos:



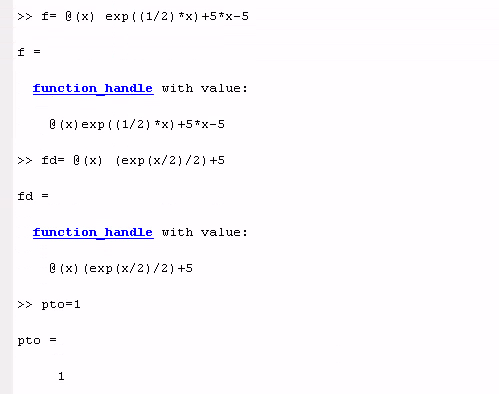
Que resulta:





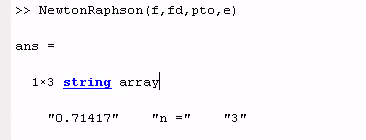
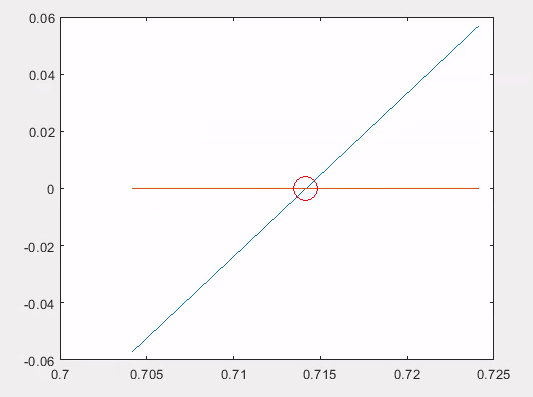
*Newton-Raphson:*

Por consola introducimos:

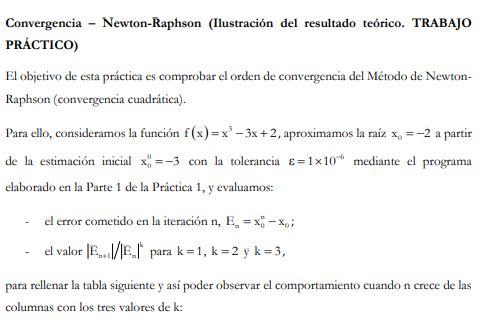


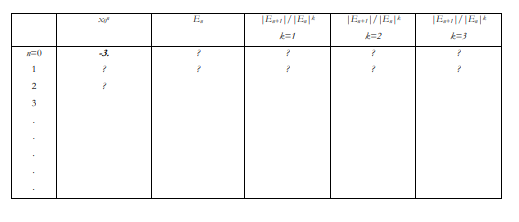


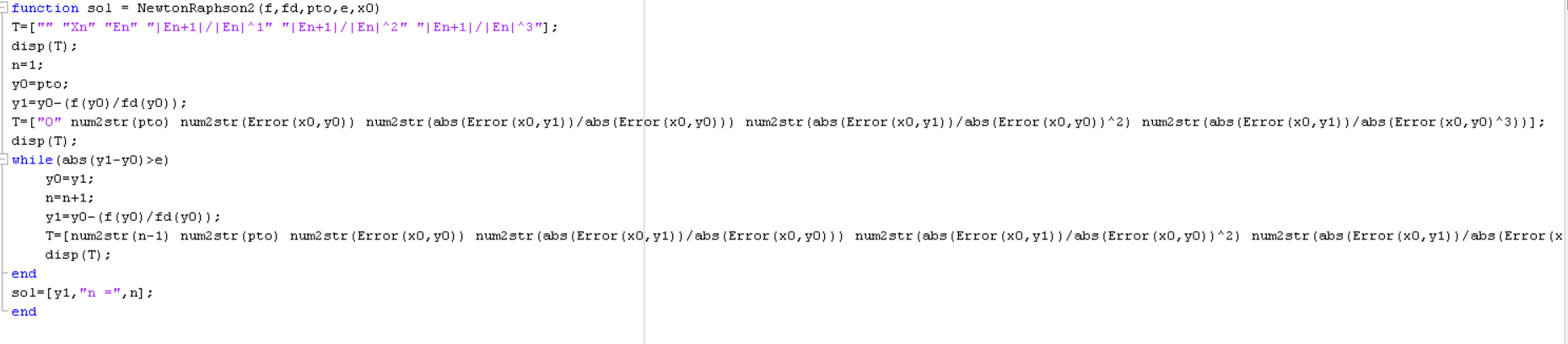
Que resulta:

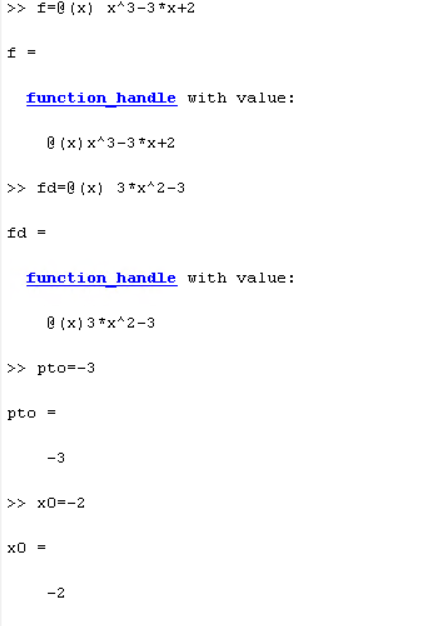
## PARTE 3: RAÍCES NUMÉRICAS DE FUNCIONES NO LINEALES

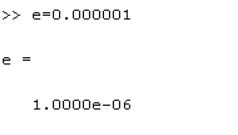




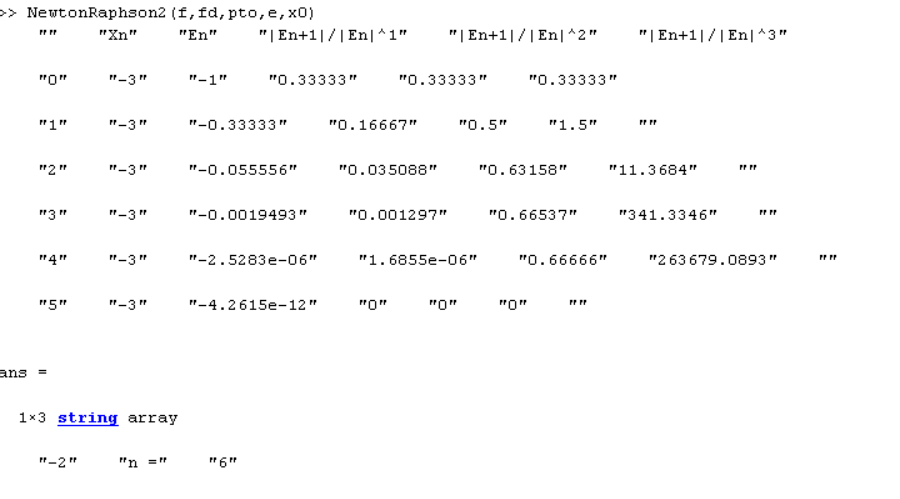
Utilizamos el siguiente programa basado en el programa ya explicado anteriormente:

Primero introducimos los datos:





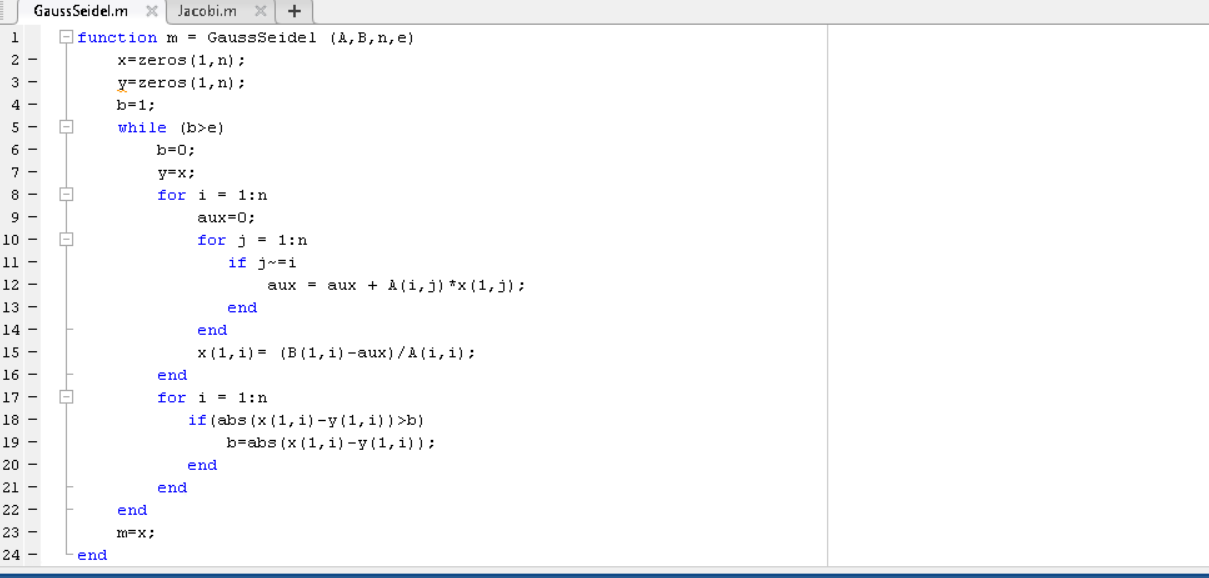
Al utilizar el programa con los datos anteriores, nos sale el siguiente resultado en tabla, y la última línea donde pone ans, nos da el resultado x6=-2 y n=6:



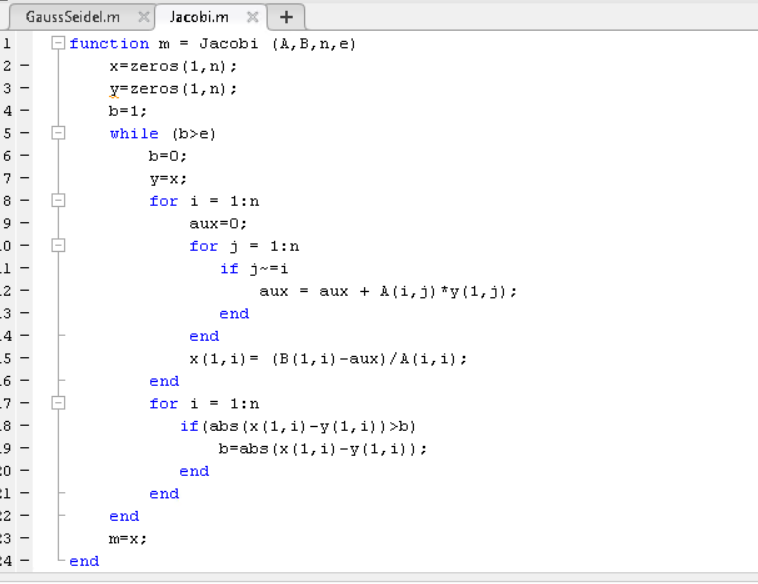
## PARTE 4: RESOLUCIÓN NUMÉRICA DE SISTEMAS LINEALES

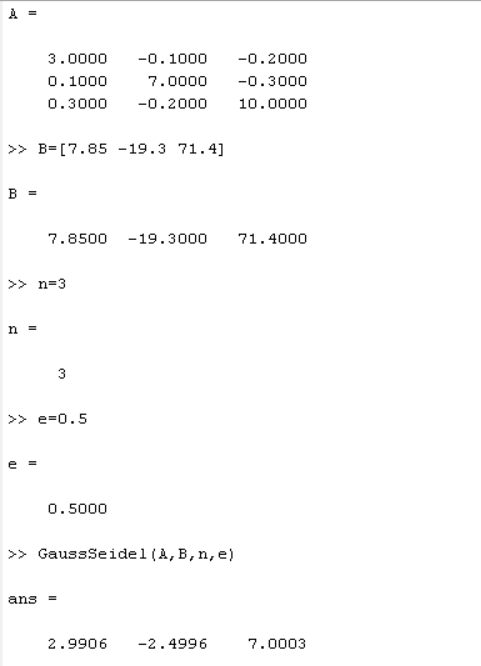
**Programar los métodos de Gauss-Seidel y de Jacobi.**

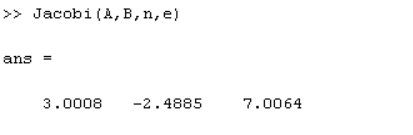
El siguiente código es el correspondiente a el método de Gauss-Seidel para resolución numérica de sistemas lineales, se le debe introducir una matriz “A” y una “B”, el número de incógnitas en la variable “n” y la cota de error en la variable “e”. El sistema a resolver es de la forma AX=B.



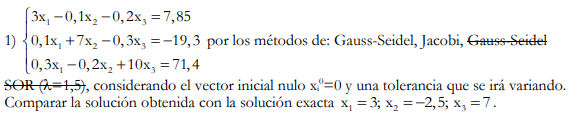
El siguiente código es el correspondiente a el método de Jacobi para resolución numérica de sistemas lineales, se le debe introducir una matriz “A” y una “B”, el número de incógnitas en la variable “n” y la cota de error en la variable “e”. El sistema a resolver es de la forma AX=B.

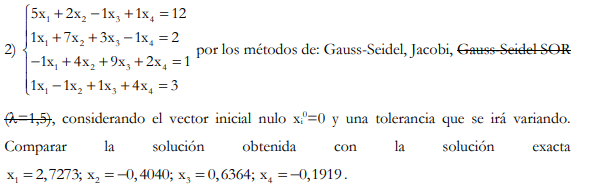




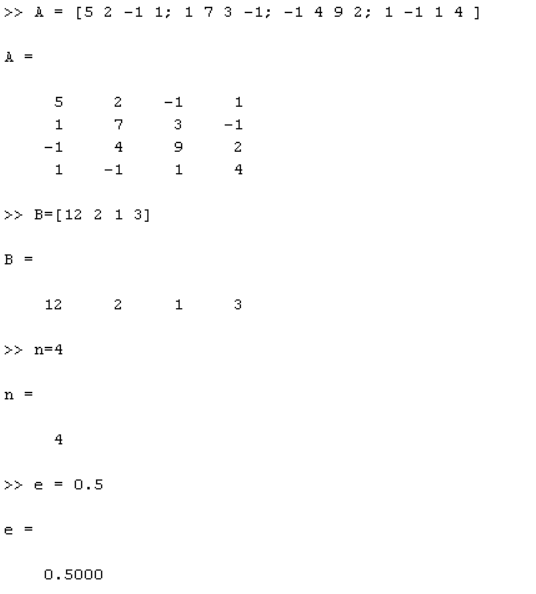


Así es como se debe introducir los datos para el funcionamiento del programa, este ejercicio es el número uno:

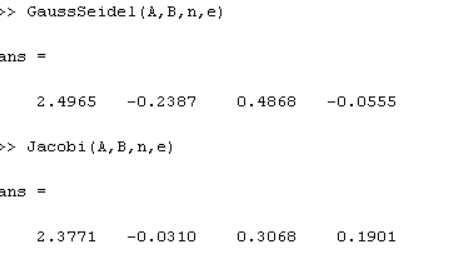




Introducimos incialmente los datos en las varibles de la siguiente manera:



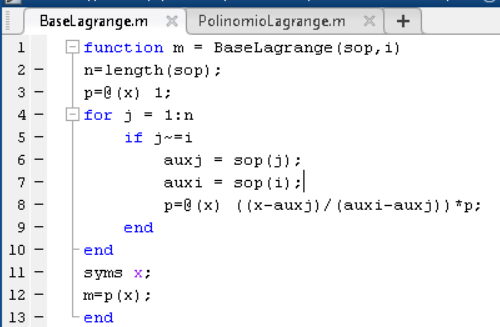
Luego usamos los programas para calcular la solución numérica:



## PARTE 5: INTERPOLACIÓN NUMÉRICA (LAGRANGE)

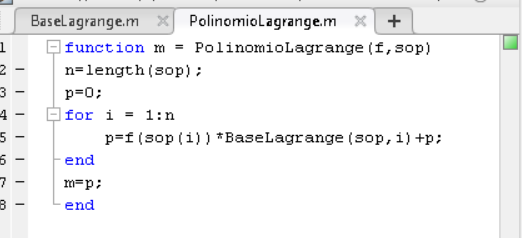
1. **Programar la expresión de los polinomios de base de Lagrange**

El código que vamos a utilizar es el siguiente:



En este primer código, le introducimos todos los valores del soporte en un array en la variable “sop”, y en la variable “i” le introducimos el orden de la base de Lagrange menos uno. Y nos devuelve una base de Lagrange.

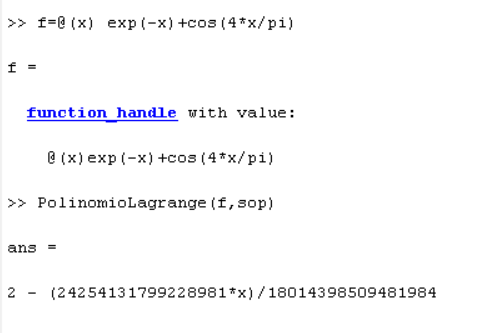
1. **Programar la expresión del polinomio interpolador de Lagrange**



En este segundo código, utilizamos el anterior para ir calculando cada una de la bases para ir incluyendo en el polinomio interpolador. Le tenemos que meter una función de la forma f = @(x) …

También le tenemos que introducir todos los valores del soporte en un array en la variable “sop”.

Con un soporte de dos valores, [0,2], y la función, el polinomio interpolador resultante es:



Si cambiamos el soporte a [0,0.5,1,1.5,2] nos sale la siguiente aproximación:

(372146326806111\*x\*(2\*x - 1)\*(2\*x - 3)\*(x - 2))/562949953421312 + (3944195464726781\*x\*(2\*x - 2)\*(2\*x - 4)\*(x - 1/2))/54043195528445952 - (6239733289746997\*x\*(2\*x - 3)\*((2\*x)/3 - 1/3)\*(x - 1))/18014398509481984 - (6352959975846737\*x\*(2\*x - 2)\*((2\*x)/3 - 4/3)\*(x - 3/2))/2251799813685248 + 2\*(2\*x - 1)\*(x/2 - 1)\*((2\*x)/3 - 1)\*(x - 1)

Si cambiamos el soporte a uno que vaya desde 0 hasta 2 con un paso de 0.1, nos sale la aproximación:

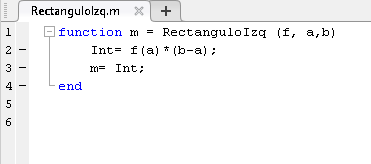
2\*(2\*x - 1)\*(x/2 - 1)\*((2\*x)/3 - 1)\*(5\*x - 1)\*((5\*x)/2 - 1)\*((5\*x)/3 - 1)\*((5\*x)/4 - 1)\*(10\*x - 1)\*((5\*x)/6 - 1)\*((5\*x)/7 - 1)\*((5\*x)/8 - 1)\*((10\*x)/3 - 1)\*((5\*x)/9 - 1)\*((10\*x)/7 - 1)\*((10\*x)/9 - 1)\*((10\*x)/11 - 1)\*((10\*x)/13 - 1)\*((10\*x)/17 - 1)\*((10\*x)/19 - 1)\*(x - 1) + (30992864632301945\*x\*((5\*x)/2 - 2)\*((5\*x)/3 - 1)\*(5\*x - 5)\*((5\*x)/2 - 4)\*((5\*x)/3 - 3)\*(5\*x - 7)\*((5\*x)/4 - 1/2)\*(2\*x - 7/5)\*((5\*x)/4 - 5/2)\*((10\*x)/3 - 3)\*((10\*x)/3 - 5)\*(10\*x - 11)\*(10\*x - 13)\*((10\*x)/9 - 1/3)\*(2\*x - 17/5)\*((10\*x)/7 - 5/7)\*((10\*x)/11 - 1/11)\*((10\*x)/7 - 19/7)\*(x - 1/5))/108086391056891904 + (372146326806111\*x\*(2\*x - 1)\*(2\*x - 3)\*(5\*x - 4)\*(5\*x - 6)\*((5\*x)/2 - 3/2)\*((5\*x)/3 - 2/3)\*((5\*x)/4 - 1/4)\*((5\*x)/2 - 7/2)\*((5\*x)/3 - 8/3)\*(10\*x - 9)\*(10\*x - 11)\*((5\*x)/4 - 9/4)\*((10\*x)/3 - 7/3)\*((10\*x)/7 - 3/7)\*((10\*x)/3 - 13/3)\*((10\*x)/9 - 1/9)\*((10\*x)/7 - 17/7)\*((10\*x)/9 - 19/9)\*(x - 2))/562949953421312 + (1192445375670355\*x\*(5\*x - 2)\*(2\*x - 1/5)\*(5\*x - 4)\*((5\*x)/2 - 1/2)\*((5\*x)/3 - 2)\*((5\*x)/2 - 5/2)\*((10\*x)/3 - 1)\*(10\*x - 5)\*((5\*x)/6 - 3/2)\*((10\*x)/3 - 3)\*(10\*x - 7)\*(2\*x - 11/5)\*((5\*x)/4 - 7/4)\*((10\*x)/9 - 5/3)\*((5\*x)/7 - 10/7)\*((10\*x)/7 - 13/7)\*((10\*x)/11 - 17/11)\*((10\*x)/13 - 19/13)\*(x - 8/5))/562949953421312 + (10965076979671585\*x\*(5\*x - 3)\*((5\*x)/2 - 1)\*(2\*x - 3/5)\*(5\*x - 5)\*((5\*x)/2 - 3)\*((5\*x)/4 - 2)\*((5\*x)/3 - 1/3)\*(10\*x - 7)\*((5\*x)/3 - 7/3)\*(10\*x - 9)\*((5\*x)/6 - 5/3)\*(2\*x - 13/5)\*((10\*x)/3 - 5/3)\*((10\*x)/7 - 1/7)\*((10\*x)/3 - 11/3)\*((10\*x)/7 - 15/7)\*((10\*x)/9 - 17/9)\*((10\*x)/11 - 19/11)\*(x - 9/5))/9007199254740992 + (3944195464726781\*x\*(2\*x - 2)\*(2\*x - 4)\*((5\*x)/3 - 3/2)\*((5\*x)/6 - 1/4)\*((10\*x)/3 - 4)\*((10\*x)/3 - 6)\*(5\*x - 13/2)\*((5\*x)/2 - 11/4)\*(5\*x - 17/2)\*((5\*x)/4 - 7/8)\*(10\*x - 14)\*((10\*x)/9 - 2/3)\*(10\*x - 16)\*((5\*x)/7 - 1/14)\*((5\*x)/2 - 19/4)\*((10\*x)/7 - 8/7)\*((10\*x)/11 - 4/11)\*((10\*x)/13 - 2/13)\*(x - 1/2))/54043195528445952 + (820478150644545\*x\*(5\*x - 6)\*(5\*x - 8)\*((5\*x)/2 - 5/2)\*((5\*x)/3 - 4/3)\*(2\*x - 9/5)\*((5\*x)/4 - 3/4)\*((5\*x)/2 - 9/2)\*((5\*x)/6 - 1/6)\*((10\*x)/7 - 1)\*((5\*x)/3 - 10/3)\*(10\*x - 13)\*(10\*x - 15)\*(2\*x - 19/5)\*((10\*x)/3 - 11/3)\*((10\*x)/3 - 17/3)\*((10\*x)/9 - 5/9)\*((10\*x)/11 - 3/11)\*((10\*x)/13 - 1/13)\*(x - 2/5))/31525197391593472 - (44633470468070265\*x\*((5\*x)/2 - 3)\*((5\*x)/4 - 1)\*(5\*x - 7)\*((5\*x)/2 - 5)\*(5\*x - 9)\*((5\*x)/6 - 1/3)\*((5\*x)/3 - 5/3)\*(2\*x - 11/5)\*((5\*x)/7 - 1/7)\*((2\*x)/3 - 1/15)\*(10\*x - 15)\*(10\*x - 17)\*((10\*x)/3 - 13/3)\*((10\*x)/7 - 9/7)\*((10\*x)/3 - 19/3)\*((10\*x)/9 - 7/9)\*((10\*x)/11 - 5/11)\*((10\*x)/13 - 3/13)\*(x - 3/5))/288230376151711744 - (18437326707188215\*x\*(2\*x - 4/5)\*((5\*x)/3 - 1/2)\*(5\*x - 7/2)\*((5\*x)/3 - 5/2)\*((10\*x)/3 - 2)\*((5\*x)/2 - 5/4)\*((10\*x)/3 - 4)\*(5\*x - 11/2)\*((5\*x)/4 - 1/8)\*(10\*x - 8)\*(10\*x - 10)\*(2\*x - 14/5)\*((10\*x)/9 - 2)\*((5\*x)/2 - 13/4)\*((10\*x)/7 - 2/7)\*((5\*x)/4 - 17/8)\*((10\*x)/7 - 16/7)\*((10\*x)/11 - 20/11)\*(x - 19/10))/20266198323167232 - (5571667004306525\*x\*((5\*x)/3 - 2)\*((2\*x)/3 - 1/5)\*(5\*x - 8)\*((5\*x)/6 - 1/2)\*(5\*x - 10)\*((5\*x)/2 - 7/2)\*((5\*x)/4 - 5/4)\*((10\*x)/3 - 5)\*(2\*x - 13/5)\*((10\*x)/9 - 1)\*((5\*x)/7 - 2/7)\*((5\*x)/8 - 1/8)\*(10\*x - 17)\*(10\*x - 19)\*((10\*x)/7 - 11/7)\*((10\*x)/11 - 7/11)\*((10\*x)/13 - 5/13)\*((10\*x)/17 - 1/17)\*(x - 4/5))/20266198323167232 + (8688615196656625\*x\*(5\*x - 1)\*(5\*x - 3)\*((5\*x)/2 - 2)\*(10\*x - 3)\*((5\*x)/4 - 3/2)\*(10\*x - 5)\*(2\*x - 9/5)\*((5\*x)/3 - 5/3)\*((10\*x)/3 - 1/3)\*((5\*x)/6 - 4/3)\*((5\*x)/8 - 5/4)\*((10\*x)/3 - 7/3)\*((5\*x)/7 - 9/7)\*((10\*x)/7 - 11/7)\*((2\*x)/3 - 19/15)\*((10\*x)/9 - 13/9)\*((10\*x)/11 - 15/11)\*((10\*x)/13 - 17/13)\*(x - 7/5))/2251799813685248 - (6352959975846737\*x\*(2\*x - 2)\*(5\*x - 3/2)\*((2\*x)/3 - 4/3)\*((5\*x)/2 - 1/4)\*(5\*x - 7/2)\*(10\*x - 4)\*(10\*x - 6)\*((10\*x)/3 - 2/3)\*((5\*x)/2 - 9/4)\*((10\*x)/3 - 8/3)\*((5\*x)/3 - 11/6)\*((5\*x)/4 - 13/8)\*((10\*x)/7 - 12/7)\*((5\*x)/6 - 17/12)\*((10\*x)/9 - 14/9)\*((5\*x)/7 - 19/14)\*((10\*x)/11 - 16/11)\*((10\*x)/13 - 18/13)\*(x - 3/2))/2251799813685248 - (6239733289746997\*x\*(2\*x - 3)\*((2\*x)/3 - 1/3)\*((5\*x)/2 - 4)\*(5\*x - 9)\*((5\*x)/4 - 3/2)\*((5\*x)/6 - 2/3)\*((5\*x)/3 - 7/3)\*((5\*x)/8 - 1/4)\*((5\*x)/7 - 3/7)\*((5\*x)/9 - 1/9)\*(10\*x - 19)\*((10\*x)/3 - 17/3)\*((10\*x)/7 - 13/7)\*((10\*x)/9 - 11/9)\*((10\*x)/11 - 9/11)\*((10\*x)/13 - 7/13)\*((10\*x)/17 - 3/17)\*((10\*x)/19 - 1/19)\*(x - 1))/18014398509481984 - (176725084298555\*x\*(2\*x - 6/5)\*(5\*x - 9/2)\*((5\*x)/2 - 7/4)\*((5\*x)/3 - 5/6)\*(5\*x - 13/2)\*((5\*x)/4 - 3/8)\*(10\*x - 10)\*(10\*x - 12)\*(2\*x - 16/5)\*((10\*x)/3 - 8/3)\*((5\*x)/2 - 15/4)\*((10\*x)/7 - 4/7)\*((10\*x)/3 - 14/3)\*((10\*x)/9 - 2/9)\*((5\*x)/3 - 17/6)\*((5\*x)/4 - 19/8)\*((10\*x)/7 - 18/7)\*((10\*x)/9 - 20/9)\*(x - 1/10))/387028092977152 - (3166626548798675\*x\*(2\*x - 2/5)\*(5\*x - 5/2)\*((5\*x)/2 - 3/4)\*((5\*x)/3 - 1/6)\*(5\*x - 9/2)\*(10\*x - 6)\*(10\*x - 8)\*(2\*x - 12/5)\*((10\*x)/7 - 2)\*((10\*x)/3 - 4/3)\*((5\*x)/2 - 11/4)\*((10\*x)/3 - 10/3)\*((5\*x)/3 - 13/6)\*((5\*x)/4 - 15/8)\*((5\*x)/6 - 19/12)\*((10\*x)/9 - 16/9)\*((10\*x)/11 - 18/11)\*((10\*x)/13 - 20/13)\*(x - 17/10))/1970324836974592 - (33906148123244585\*x\*(2\*x - 8/5)\*(5\*x - 11/2)\*((5\*x)/2 - 9/4)\*((5\*x)/3 - 7/6)\*(5\*x - 15/2)\*((5\*x)/4 - 5/8)\*(10\*x - 12)\*(10\*x - 14)\*((5\*x)/6 - 1/12)\*(2\*x - 18/5)\*((10\*x)/3 - 10/3)\*((5\*x)/2 - 17/4)\*((10\*x)/7 - 6/7)\*((10\*x)/3 - 16/3)\*((10\*x)/9 - 4/9)\*((5\*x)/3 - 19/6)\*((10\*x)/11 - 2/11)\*((10\*x)/7 - 20/7)\*(x - 3/10))/234187180623265792 + (20114008205015035\*x\*(5\*x - 2)\*(10\*x - 1)\*((5\*x)/2 - 3/2)\*(10\*x - 3)\*(2\*x - 7/5)\*((5\*x)/3 - 4/3)\*((5\*x)/4 - 5/4)\*((10\*x)/3 - 5/3)\*((5\*x)/6 - 7/6)\*((5\*x)/7 - 8/7)\*((5\*x)/8 - 9/8)\*((5\*x)/9 - 10/9)\*((10\*x)/7 - 9/7)\*((2\*x)/3 - 17/15)\*((10\*x)/9 - 11/9)\*((10\*x)/11 - 13/11)\*((10\*x)/13 - 15/13)\*((10\*x)/17 - 19/17)\*(x - 6/5))/2251799813685248 - (37576901437273555\*x\*(5\*x - 1/2)\*(5\*x - 5/2)\*(10\*x - 2)\*((5\*x)/3 - 3/2)\*(10\*x - 4)\*(2\*x - 8/5)\*((10\*x)/3 - 2)\*((2\*x)/3 - 6/5)\*((5\*x)/2 - 7/4)\*((5\*x)/6 - 5/4)\*((10\*x)/9 - 4/3)\*((5\*x)/4 - 11/8)\*((10\*x)/7 - 10/7)\*((5\*x)/7 - 17/14)\*((10\*x)/11 - 14/11)\*((5\*x)/8 - 19/16)\*((10\*x)/13 - 16/13)\*((10\*x)/17 - 20/17)\*(x - 13/10))/6755399441055744 + (2120954471629175\*x\*(2\*x - 12/5)\*((2\*x)/3 - 2/15)\*(5\*x - 15/2)\*((5\*x)/2 - 13/4)\*((5\*x)/3 - 11/6)\*(5\*x - 19/2)\*((5\*x)/4 - 9/8)\*(10\*x - 16)\*(10\*x - 18)\*((5\*x)/6 - 5/12)\*((5\*x)/7 - 3/14)\*((5\*x)/8 - 1/16)\*((10\*x)/3 - 14/3)\*((10\*x)/7 - 10/7)\*((10\*x)/3 - 20/3)\*((10\*x)/9 - 8/9)\*((10\*x)/11 - 6/11)\*((10\*x)/13 - 4/13)\*(x - 7/10))/9570149208162304 + (27049863061981845\*x\*(2\*x - 14/5)\*((2\*x)/3 - 4/15)\*(5\*x - 17/2)\*((5\*x)/2 - 15/4)\*((5\*x)/3 - 13/6)\*((5\*x)/4 - 11/8)\*(10\*x - 18)\*(10\*x - 20)\*((5\*x)/6 - 7/12)\*((5\*x)/7 - 5/14)\*((5\*x)/8 - 3/16)\*((10\*x)/3 - 16/3)\*((5\*x)/9 - 1/18)\*((10\*x)/7 - 12/7)\*((10\*x)/9 - 10/9)\*((10\*x)/11 - 8/11)\*((10\*x)/13 - 6/13)\*((10\*x)/17 - 2/17)\*(x - 9/10))/85568392920039424 - (5338855982479255\*x\*(5\*x - 3/2)\*(10\*x - 2)\*(2\*x - 6/5)\*((5\*x)/2 - 5/4)\*((10\*x)/3 - 4/3)\*((5\*x)/3 - 7/6)\*((5\*x)/4 - 9/8)\*((10\*x)/7 - 8/7)\*((2\*x)/3 - 16/15)\*((5\*x)/6 - 13/12)\*((10\*x)/9 - 10/9)\*((5\*x)/7 - 15/14)\*((10\*x)/11 - 12/11)\*((5\*x)/8 - 17/16)\*((10\*x)/13 - 14/13)\*((5\*x)/9 - 19/18)\*((10\*x)/17 - 18/17)\*((10\*x)/19 - 20/19)\*(x - 11/10))/281474976710656

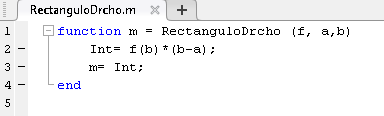
## PARTE 6: INTEGRACIÓN NUMÉRICA

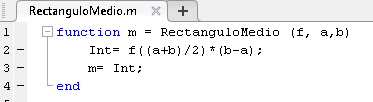
1. **Programar las 3 fórmulas del Rectángulo en Matlab**

Las 3 fórmulas del rectángulo son: Rectángulo con el punto en el extremo izquierdo, rectángulo con el punto en el extremo derecho, rectángulo con el punto en el punto medio.

A cada función *RectanguloIzq, RectanguloDrcho, RectanguloMedio* se la introducen *f, a, b,* siendo *f* la función correspondiente a la integral, y *a, b* los puntos de la integral. Todas las funciones devuelven *m* que es el resultado de la integral.

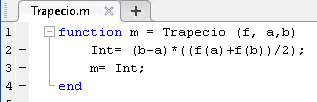






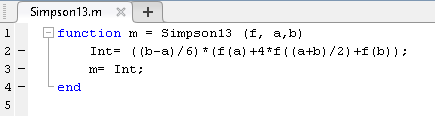
1. **Programar la fórmula del Trapecio en Matlab**

A la función *Trapecio* se la introducen *f, a, b,* siendo *f* la función correspondiente a la integral, y *a, b* los puntos de la integral. Todas las funciones devuelven *m* que es el resultado de la integral.



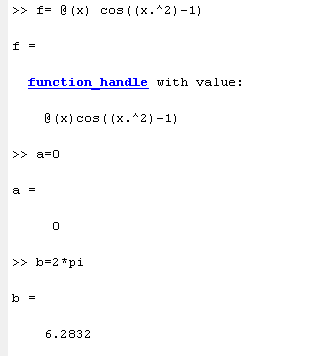
1. **Programar la fórmula de Simpson 1/3 en Matlab**

A la función *Simpson13* se la introducen *f, a, b,* siendo *f* la función correspondiente a la integral, y *a, b* los puntos de la integral. Todas las funciones devuelven *m* que es el resultado de la integral.

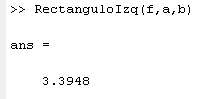


**Aplicar los programas desarrollados al cálculo de la integral **

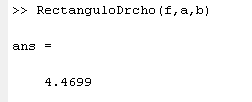
Introducimos por consola:



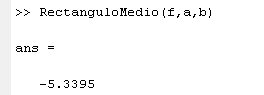
Rectángulo Extremo Izquierdo:

****

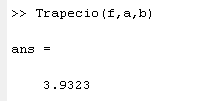
Rectángulo Extremo Derecho:



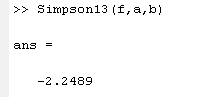
Rectángulo Punto Medio:



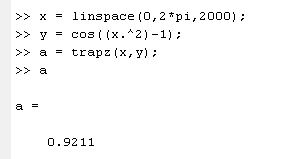
Trapecio:

****

Simpson ⅓:



**Comparar los resultados con el resultado de referencia Ir obtenido empleando el comando trapz(x,f) o el comando @(x) (error relativo |I-Ir|/Ir)**



Como se puede comprobar, las soluciones aproximadas obtenidas por nuestros programas son bastante dispares unas de otras y, además, están bastante lejanas de la solución obtenida por la función “trapz”. Esto se debe a que todas son aproximaciones o cálculos numéricos de las integrales y no cálculos analíticos de la integral, y en este caso se nota bastante ya que la función oscila y según nos vamos separando del x=0, la velocidad a la que oscila va aumentando rápidamente, esto da lugar a que una aproximación en la que cogemos solo una cierta cantidad de puntos, puedan no ser suficientes para que la solución obtenida se acerque mucho a la real.

* **Pregunta optativa:**

**Deducir la fórmula de Milne compuesta y la expresión de su error. Programar la fórmula de Milne compuesta en Matlab.**

Formula de Milne Compuesta:

;

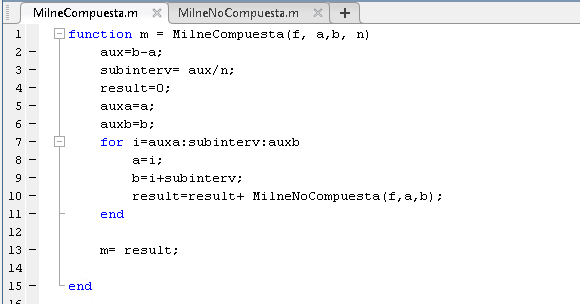
Expresión de su Error:

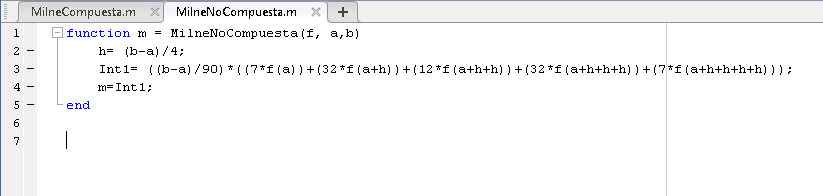
Siendo K= 8/945 O(

Para programar la formula de Milne hemos utilizado dos funciones: *MilneCompuesta* y *MilneNoCompuesta.*

*MilneCompuesta* es una función a la que se le introduce *f, a, b , n.* Siendo *f* la función de la integral, *a* y *b* los puntos extremos de la integral, y *n* los subintervalos de la integral. Esta función nos devuelve *m* que es el resultado de la integral utilizando *MilneCompuesta* en *n* subintervalos.

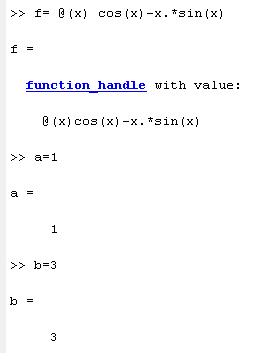
*MilneNoCompuesta* es una función a la que se le introduce *f, a, b.* Siendo *f* la función de la integral,y *a* y *b* los puntos extremos de la integral. Esta función nos devuelve *m* que es el resultado de la formula de Milne en el subintervalo dado.



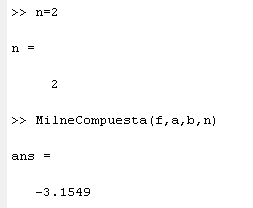


**Aplicar el programa desarrollado al cálculo de la integral  dividiendo el intervalo [1, 3] en N sub-intervalos y dando a N los valores: 2, 4, 8, 16, 32, …**

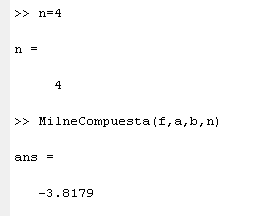
Introducimos por consola:



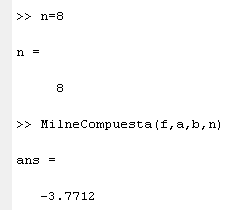
Para N=2:



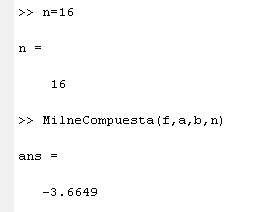
Para N=4:



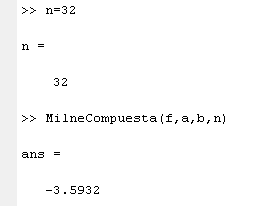
Para N=8:



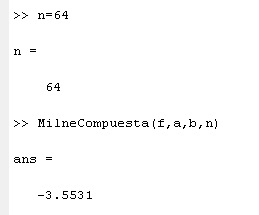
Para N=16:



Para N=32:



Para N=64:



Para N= 128:

