PRÁCTICA 4: Raíces de funciones

Como en los módulos anteriores, construye tu informe de prácticas incorporando tus programas, resultados, figuras o/y comentarios o conclusiones.

Introducción al cálculo de raíces con MATLAB

MATLAB proporciona la función x=***fzero***(*nombre_funcion*,*x0*,*tol*,*it*) para obtener la raíz de una función. Si la función está definida en un fichero '.m' (en cuyo caso debe tener la estructura y=nombre_funcion(x)) el primer argumento de ***fzero*** es una cadena con el nombre de la función (por ejemplo, el nombre de la función entre comillas simples). Alternativamente, la función puede ser definida usando el comando ***inline***. Los otros argumentos de entrada a ***fzero*** son la aproximación inicial *x0*, el número de iteraciones del proceso iterativo *it* (si *it* es igual a 1, el proceso se repite hasta que la solución esté dentro de una tolerancia *tol*). Los dos últimos argumentos se pueden omitir. La función ***fzero*** emplea el método de Brent, que combina la interpolación cuadrática inversa con la bisección.

Para evaluar el valor de una función almacenada en una variable se usa la función interna ***feval***. Su sintaxis es y = ***feval***('fun', *x*). Teclea help feval para más información.

Existen también funciones internas de Matlab para el cálculo de raíces de polinomios. La función r=***roots***(*c*) calcula las raíces *r* de un polinomio cuyos coeficientes se encuentran almacenados en el vector *c* de la forma: $c(1)x^n+...+c(n)x+c(n+1)$. De forma inversa, la función c=***poly***(*r*) genera un polinomio que tiene como raíces los valores almacenados en el vector *r*. Finalmente, y=***polyval***(*c*,*x*) evalúa el valor del polinomio cuyos coeficientes vienen dados en *c* en el punto *x*.

.....

1. Sea el siguiente polinomio $p(x)=x^2-5x+6$. Determina sus raíces mediante funciones internas de Matlab. Determina los valores que toma el polinomio p(x) en x=[1.5 3.5].

2. Con las funciones internas **conv** y **deconv** se puede multiplicar y dividir polinomios. Sean p(x) = 2x + 1 y q(x) = 3x + 4, determina el polinomio resultante de su multiplicación y asigna el resultado a un polinomio r(x). Divide este polinomio r(x) por el polinomio p(x). Con los operadores + y – se calcula la suma y resta de polinomios. Resta y suma los polinomios r y p.

3. El siguiente ejemplo nos permite analizar el **condicionamiento de una función**, en este caso de un polinomio de grado 10. Una función está mal condicionada cuando sus raíces son muy sensibles a pequeños cambios en los coeficientes. Es importante conocer si una función está mal o bien condicionada ya que en caso de que esté mal condicionada, como en este ejemplo, los resultados obtenidos serían poco fiables. En este ejemplo estudiaremos cómo cambian las raíces de un polinomio cuando se altera levemente uno de sus coeficientes.

   ```
   >>X=[1:10]; % X son las raíces del polinomio P

   >>P=poly(X); P(2)=P(2)+0.001; nuevas_raices=roots(P)
   ```

   Compara las raíces originales y las nuevas raíces tras la modificación del polinomio. ¿Este polinomio está bien o mal condicionado?

4. Programa una función en el fichero mifuncion.m que evalúe la siguiente expresión matemática $y=e^{\sin(x)} - 2*\cos(x)$. Dibuja dicha función en el intervalo [0,10]. ¿Cuántas raíces tiene esta función? Calcular todas las raíces de la función en el intervalo [0, 10] llamando a la función ***fzero*** cuantas veces sean necesarias, usando distintos valores de x0 cada vez. Detalla todas las instrucciones y las raíces que has obtenido.

5. Programa una función df=***resta***(*fun1*, *fun2*, *x*) que evalúe la diferencia entre dos funciones arbitrarias *fun1* y *fun2* en el punto *x*. Usa tu función para dibujar la diferencia entre las funciones $y_1=x^2$, $y_2=\sin(x)$ en el intervalo [0, 2].

6. Escribe una función que calcule una raíz de una función cualquiera usando el método de bisección. Para ello escribe una función, [x,it]=***bisecc***(*funcion*,*a*,*b*), que admita como parámetros de entrada cualquier función tipo *fun.m* y el intervalo (*a*,*b*) donde se ha de buscar la raíz, y devuelva la raíz x y el número de iteraciones it del método de bisección. La tolerancia, *tol*, para el cálculo de la raíz se introducirá por teclado usando ***input***. Utiliza el comando ***fprintf***('\n %i %f %f %f %f',it, x, y,a,b) para sacar por pantalla durante la ejecución los resultados parciales en cada iteración. Usa tu función ***bisecc*** para calcular las raíces de la función f(x)= x-sen(x)-1 y comprueba el resultado obtenido comparando tu raíz con la obtenida usando ***fzero***.

7. Aplica tu función bisecc.m para determinar las dos raíces del ejercicio 1.

8. Programa una función [x,it] = ***interpol***(*g*,*tol*,*x0*,*x1*) que obtenga numéricamente una raíz de la función matemática g(x) usando el método iterativo de interpolación lineal con una tolerancia *tol*, siendo *x0* y *x1* los valores iniciales (próximos a la raíz exacta x) con los que comienza el método. La ejecución de la función debe incluir la aparición en pantalla de los pasos o iteraciones sucesivas, mostrando los valores de: nº de iteración, solución parcial aproximada x y el valor de fun(x). En este caso, se definirá la función de entrada *g* usando el comando ***inline*** en vez de mediante un fichero '.m'. Usa tu programa para calcular las raíces de la función f(x) del ejercicio 6.

9. El cálculo de la raíz cuadrada de 3 se puede determinar calculando la raíz positiva de la ecuación $x^2=3$; si la reescribimos tenemos la expresión $x=(3+x)/(1+x)$. Programa una función que determine la raíz cuadrada de 3 por el ***método del punto fijo*** con la expresión que se indica, considerando como valor inicial 1. ¿Cuántas iteraciones se han realizado para llegar a tener trece decimales exactos de precisión?

10. Programa una función [x, it]=***newton***(*fun*, *der*, *x0*, *tol*) para calcular la raíz de una función cualquiera usando el método de Newton, y emplea dicha función para calcular las raíces de la función del ejercicio 6 y la raíz cuadrada del ejercicio 9. En este caso, la función ***newton*** recibirá como argumentos de entrada tanto la función cuya raíz se desea calcular (*fun*) como su función derivada (*der*), que calcularás analíticamente previamente a la ejecución, , *x0* la aproximación inicial y *tol* la tolerancia. Las funciones *fun* y *der* serán definidas en ficheros *.m*. Para cada una de las funciones de los ejercicios 6 y 9 construye una tabla de síntesis indicando, el valor inicial considerado, si converge o no el método, si converge indicar el número de iteraciones realizadas y la raíz. En todos los casos considerar una tolerancia de 10^{-10}.