Tarea 2

Carmelo García
Física Computacional

30 de noviembre de 2024

1. Problemas

1. Utilice el método de *bisección* para aproximar la raíz quinta de 2 con tres cifras decimales exactas.

2. Hallar la menor raíz positiva con su error de la ecuación $e^{-x^2} - \cos(x) = 0$ con una tolerancia de 10^{-5}, utilizando el método de *Regula Falsi* y muestre en pantalla el error cometido en cada iteración.

3. Compara en términos de rapidez de convergencia, los métodos numéricos *Bisección* y *Regula Falsi*, para resolver la ecuación $p(x) = x^{10} - 1 = 0$, con una tolerancia de 10^{-6}.

4. La corriente eléctrica en un circuito RLC, para determinados valores de la resistencia R, la inductancia L y la capacitancia C, viene dada por la expresión:

   $$i(t) = 10e^{-t/2}\cos(3t).$$

   Una serie de preguntas sobre el circuito nos obligan a resolver ecuaciones no lineales. Veamos alguna de ellas.

   a) ¿Para qué instantes de tiempo la intensidad de la corriente es nula?

   b) ¿En qué instantes de tiempo la corriente es de 2 mA?

   c) ¿En qué instantes de tiempo la intensidad de la corriente es máxima?

2. Soluciones

2.1. Problema 1

El fragmento de código presentado a continuación, es el corazón de la solución del problema 1 el cual implementa un algoritmo de bisección para hallar las raíces de una función en un intervalo dado $[a, b]$. Inicialmente, el programa solicita al usuario que introduzca los límites del intervalo. Dentro de un bucle que itera hasta 100 veces, se calcula el punto medio

x_r del intervalo $[a, b]$. Si el valor absoluto de la función evaluada en x_r es menor o igual a una tolerancia predeterminada, se considera que se ha encontrado una raíz y el programa imprime el valor de x_r junto con el número de iteraciones realizadas. Si la función evaluada en el punto medio x_r y en el extremo inferior a tienen signos opuestos, la raíz se encuentra en el intervalo $[a, x_r]$, y se actualiza el extremo superior b a x_r. De lo contrario, la raíz está en el intervalo $[x_r, b]$ y se actualiza el extremo inferior a a x_r. Si el número máximo de iteraciones se alcanza sin encontrar una raíz que satisfaga la tolerancia especificada, el programa informa al usuario. Esta técnica garantiza la convergencia hacia una raíz en casos donde la función es continua en el intervalo especificado.

```
void main() {
    double a, b , x_r;
    int count = 0;
    printf("Ingrese el rango donde buscar las raices [a,b]: ");
    scanf("%lf %lf", &a, &b);
    while (count < 100) {
        x_r = (a + b) / 2; // Calcular el punto medio del metodo de biseccion
        if (fabs(f(x_r)) <= tolerancia(3)) {
            printf("La raiz es: %lf\n", x_r);
            printf("Numero de pasos: %d\n", count);
            break;
        } else if (f(a) * f(x_r) < 0) { // Si f(a) * f(x_r) < 0, entonces la raiz esta en el intervalo [a, x_r]
            b = x_r;
        } else { // Si no, la raiz esta en el intervalo [x_r, b]
            a = x_r;
        }
        count++;
    }
    if (count == 100) {// Si se alcanza el numero maximo de iteraciones sin convergencia
        printf("Se alcanzo el numero maximo de iteraciones sin convergencia.\n");
    }
}

```

Listing 1: Problema 1.

2.2. Problema 2

El fragmento de código presentado implementa un algoritmo de bisección modificado para hallar las raíces de una función en un intervalo dado $[a, b]$. El programa solicita al usuario que introduzca los límites del intervalo. En cada iteración del bucle, que puede ejecutar hasta 100 veces, se calcula el punto x_r utilizando la fórmula del método de la falsa posición. Se calcula también el error relativo entre el valor actual de x_r y el valor anterior x_{rant}. Si el valor absoluto de la función evaluada en x_r es menor o igual a una tolerancia especificada, se considera que se ha encontrado una raíz, y el programa imprime el valor de x_r junto con el número de iteraciones realizadas. Si el producto de la función evaluada en el extremo inferior

a y en el punto medio x_r es negativo, la raíz se encuentra en el intervalo $[a, x_r]$ y se actualiza el extremo superior b a x_r. De lo contrario, la raíz está en el intervalo $[x_r, b]$ y se actualiza el extremo inferior a a x_r. En cada iteración, se imprime el error relativo calculado. Si el número máximo de iteraciones se alcanza sin encontrar una raíz que satisfaga la tolerancia especificada, el programa informa al usuario. Esta técnica asegura la convergencia hacia una raíz en casos donde la función es continua en el intervalo especificado.

```
void main() {
    double a, b, x_r, x_r_ant, error;
    int count = 0;
    printf("Ingrese el rango donde buscar las raices [a,b]: ");
    scanf("%lf %lf", &a, &b);
    while (count < 100) {
        x_r = b - (f(b) * (b - a)) / (f(b) - f(a));
        error = fabs((x_r - x_r_ant) / x_r);
        if (fabs(f(x_r)) <= tolerancia(7)) {
            printf("La raiz es: %lf\n", x_r);
            printf("Numero de pasos: %d\n", count);
            break;
        } else if (f(a) * f(x_r) < 0) {
            b = x_r;
        } else {
            a = x_r;
        }
        printf("Error: %lf\n", error);
        x_r_ant = x_r;
        count++;
    }
    }

```

Listing 2: Problema 2.

2.3. Problema 3

El código presentado define y utiliza dos funciones, `regula_fasi` y `biseccion`, para encontrar las raíces de una función en un intervalo dado $[a, b]$. La función `regula_fasi` implementa el método de la falsa posición (regula falsi), calculando en cada iteración el punto x_r utilizando la fórmula de la falsa posición, y verifica si el valor absoluto de la función evaluada en x_r es menor o igual a una tolerancia específica. Si se cumple esta condición, se imprime la raíz y el número de pasos necesarios. De lo contrario, se actualiza el intervalo $[a, b]$ según corresponda y se repite el proceso hasta un máximo de 100 iteraciones.

La función `biseccion`, por otro lado, implementa el método de bisección. En cada iteración, se calcula el punto medio x_r del intervalo $[a, b]$. Si el valor absoluto de la función evaluada en x_r es menor o igual a una tolerancia específica, se considera que se ha encontrado una raíz y se imprime el valor de x_r junto con el número de pasos realizados. Si la función evaluada en x_r y en a tienen signos opuestos, la raíz se encuentra en el intervalo $[a, x_r]$, y se actualiza el extremo superior b a x_r. De lo contrario, la raíz está en el intervalo $[x_r, b]$ y se actualiza el extremo inferior a a x_r. Si el número máximo de iteraciones se alcanza sin encontrar una raíz que satisfaga la tolerancia especificada, se informa al usuario.

Finalmente, la función `main` solicita al usuario que introduzca los límites del intervalo y llama a las dos funciones mencionadas para encontrar las raíces utilizando ambos métodos. Esto con el fin de comparar ambos métodos y determinar cual converge mas rápido, o mejor dicho cual obtiene el resultado en menos pasos.

```
void regula_fasi(double a, double b) {
    double x_r, x_r_ant;
    int count = 0;
    while (count < 100) {
        x_r = b - (f(b) * (b - a)) / (f(b) - f(a));
        if (fabs(f(x_r)) <= tolerancia(7)) {
            printf("Metodo regula fasi\n");
            printf("La raiz es: %lf\n", x_r);
            printf("Numero de pasos: %d\n", count);
            break;
        } else if (f(a) * f(x_r) < 0) {
            b = x_r;
        } else {
            a = x_r;
        }
        x_r_ant = x_r;
        count++;
    }
}
void biseccion(double a, double b) {
    double x_r;
    int count = 0;
    while (count < 100) {
        x_r = (a + b) / 2;
        if (fabs(f(x_r)) <= tolerancia(7)) {
            printf("Metodo biseccion\n");
            printf("La raiz es: %lf\n", x_r);
            printf("Numero de pasos: %d\n", count);
            break;
        } else if (f(a) * f(x_r) < 0) {
            b = x_r;
        } else {
            a = x_r;
        }
        count++;
    }
    if (count == 100) {
        printf("El metodo de biseccion alcanzo el numero maximo de
   iteraciones sin convergencia.\n");
    }
}
void main() {
    double a, b;
    printf("Ingrese el rango donde buscar las raices [a,b]: ");
    scanf("%lf %lf", &a, &b);
    regula_fasi(a, b);
    biseccion(a, b);
}
```

```
48
```

Listing 3: Problema 3.

2.4. Problema 4

El código proporcionado implementa el método de Regula Falsi para resolver ecuaciones no lineales en un circuito RLC. El programa define tres funciones: `f`, `f1` y `df`, que representan la función de corriente eléctrica original, la función ajustada para encontrar cuando la corriente es de 2 mA, y la derivada de la función de corriente, respectivamente. Utiliza un bucle principal para iterar sobre intervalos específicos y aplicar el método de Regula Falsi para encontrar los puntos donde la corriente es nula, máxima y de 2 mA. Las soluciones se determinan al verificar cambios de signo en las evaluaciones de la función dentro de cada intervalo, y el resultado se imprime en pantalla, detallando los instantes de tiempo correspondientes a las condiciones especificadas. La implementación garantiza la precisión de los resultados mediante el uso de una tolerancia predefinida y un número máximo de iteraciones, optimizando así la convergencia hacia las raíces de las ecuaciones no lineales del sistema.

```
1  double f(double x) {
2      return 10 * exp(-x / 2) * cos(3 * x); // Funci n que deseamos
       estudiar
3  }
4  double f1(double x) {
5      return 10 * exp(-x / 2) * cos(3 * x) - 2e-3; // Funcion para 2 mA
6  }
7  double df(double x) {
8      return -5 * exp(-x / 2) * cos(3 * x) - 15 * exp(-x / 2) * sin(3 * x);
       // Derivada de la funcion
9  }
10 // Definir el tipo de la funcion
11 typedef double (*func_ptr)(double);
12 // Definir la tolerancia
13 double tolerancia(int n) {
14     return 0.5 * pow(10, 2 - n);
15 }
16 void regula_falsi(double a, double b, func_ptr f) {
17     double x_r, x_r_ant;
18     int count = 0;
19     while (count < 100) {
20         x_r = b - (f(b) * (b - a)) / (f(b) - f(a));
21         if (fabs(f(x_r)) <= tolerancia(7)) {
22             printf("t= %lf s\n", x_r);
23             break;
24         } else if (f(a) * f(x_r) < 0) {
25             b = x_r;
26         } else {
27             a = x_r;
28         }
29         x_r_ant = x_r;
30         count++;
31     }
```

```
}
// Funcion para encontrar los puntos criticos usando el metodo de Regula Falsi
void encontrar_maximos(double a, double b, double paso) {
    double x, x_prev, df_val;
    int maximos_encontrados = 0;
    for (x = a; x <= b; x += paso) {
        df_val = df(x); // Verificar si la derivada cambia de signo
        if (x != a) {
            if (df(x_prev) > 0 && df_val < 0) {
                // Usar regula_falsi para encontrar el maximo local
                regula_falsi(x_prev, x, df);
                maximos_encontrados++;
            }
        }
        x_prev = x;
    }
    if (maximos_encontrados == 0) {
        printf("No se encontraron maximos locales en el intervalo [%lf, %lf].\n", a, b);
    }
}
int main() {
    int a = 0, b = 0, c = 0;
    double paso = 0.1; // Tamano del paso para encontrar maximos
    printf("La intensidad de corriente se anula en los siguientes instantes:\n");
    while (a < 14) {
        regula_falsi(a, a + 1, f);
        a++;
    }
    printf("La intensidad de corriente es maxima en los siguientes instantes:\n");
    encontrar_maximos(0, 14, paso);
    printf("La intensidad de corriente es de 2 mA en los siguientes instantes:\n");
    while (c < 15) {
        regula_falsi(c, c + 1, f1);
        c++;
    }
    return 0;
}

```

Listing 4: Problema 4.

3. Resultados

3.1. Problema 1

Para hacer este ejercicio, primero se definió la función para la cual se le van a buscar las raíces por el método de bisección. Esta función es la siguiente:

$$F(x) = x^5 - 2 \tag{1}$$

El código 1 encuentra la raíz de la función 1. La cual es:

$$x = 1{,}148438$$

3.2. Problema 2

Este problema nos exige encontrar la menor raíz positiva de la función 2, es decir, la raíz mas cercana al cero.

$$f(x) = e^{-x^2} - \cos(x) \tag{2}$$

Si se gráfica esta función, podremos observar que la primera raíz es mayor que 1.

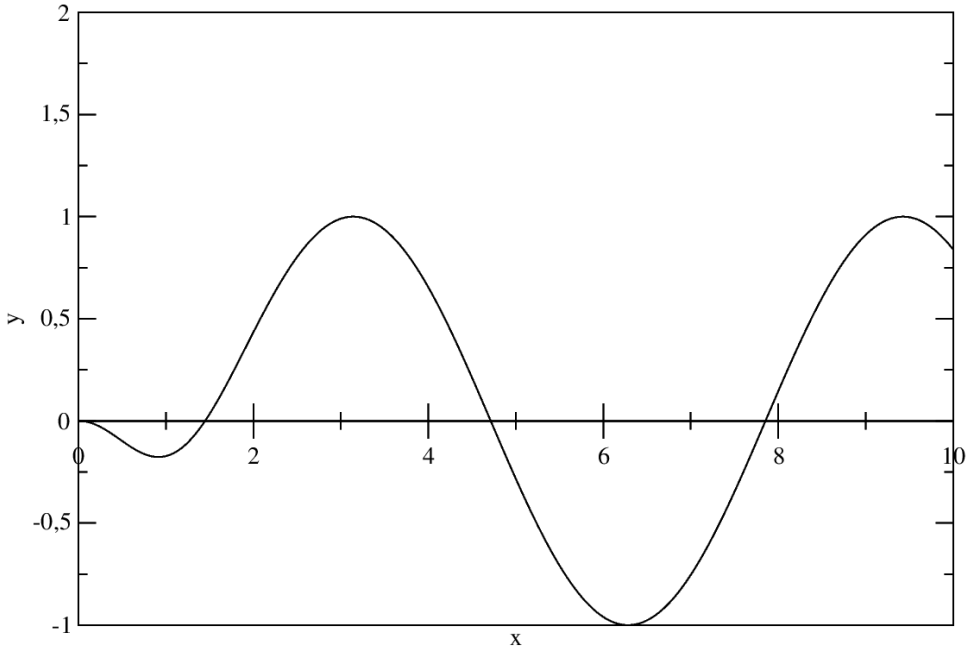

Figura 1: Función 2

El programa que se hizo para analizar esta función, se puede revisar en la sección 2.2, da como resultado que la primera raíz positiva es:

$$x = 1{,}447412 \pm 0{,}000032$$

3.3. Problema 3

El objetivo de este problema es comparar el método de bisección y el de *Regula Falsi*. La función a estudiar será:

$$f(x) = x^{10} - 1 \tag{3}$$

Claramente, la raíz de esta función es $f(x = 1) = 0$. Pero como se mencionó antes, el objetivo es comparar la rapidez de convergencia entre ambos métodos.

Parámetros [a,b]	[0.8, 1.4]
Regula Falsi	97 pasos
Bisección	20 pasos

Cuadro 1: Comparación de métodos.

A partir de de lo anterior se podría decir que el método de bisección es mucho mas rápido. Sin embargo, esto podría no ser cierto. La velocidad del método regula falsi puede converger más rápidamente que el método de bisección, especialmente si la función es bien comportada cerca de la raíz. Sin embargo, en casos donde la función es muy no lineal o las pendientes son desiguales, puede volverse más lento y menos eficiente. Si vemos la gráfica de la función 3, se puede notar que en donde está la raíz la función es casi vertical, lo que le dificulta al método regula falsi converger.

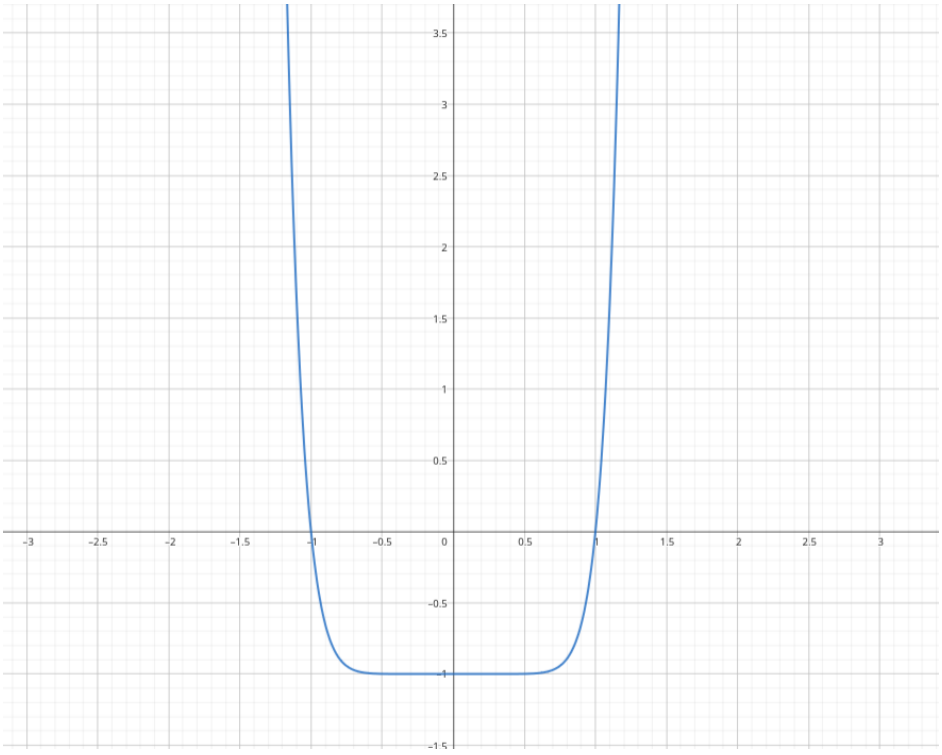

Figura 2: Función 3

Si probamos el programa con la función 2 del problema anterior, podemos ver una historia diferente.

Parámetros [a,b]	[1, 2]
Regula Falsi	8 pasos
Bisección	17 pasos

Cuadro 2: Comparación de métodos con función 2.

Ahora si se puede notar la diferencia de velocidades de convergencia, si se observa la figura 1, donde esta la raíz el cambio es menos abrupto.

En resumen, aunque el método de Regula Falsi tiene el potencial de converger más rápidamente que el método de bisección debido a su estrategia de interpolación, la elección del método puede depender de la función específica y del comportamiento cercano a la raíz. En situaciones donde la certeza y la simplicidad son prioritarias, la bisección es una opción sólida, mientras que para una convergencia más rápida y eficiente, especialmente en funciones suaves y bien definidas, el método de Regula Falsi puede ser más adecuado.

3.4. Problema 4

En este problema estudiamos la ecuación de intensidad de corriente.

$$i(t) = 10e^{-t/2}\cos(3t) \tag{4}$$

Si graficamos esta función de intensidad, podemos notar que la intensidad se anula en varios instantes.

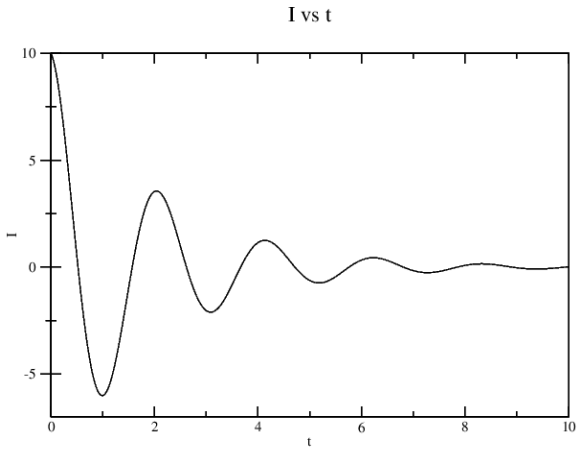

Figura 3: Intensidad vs tiempo

3.4.1. Parte a: ¿Para qué instantes de tiempo la intensidad de la corriente es nula?

Necesitamos saber para cuales instantes de tiempo la intensidad se hace cero, es decir las raíces de la función 4. El codigo 4 nos da los siguientes instantes:

t
0.523599 s
1.570796 s
2.617994 s
3.665191 s
4.712389 s
5.759587 s
6.806785 s
7.853982 s
8.901190 s
9.948381 s

Cuadro 3: Instantes donde la intensidad se anula en los primeros 10s

3.4.2. Parte b: ¿En qué instantes de tiempo la corriente es de 2 mA?

Necesitamos saber para cuales instantes de tiempo la intensidad se hace cero, es decir las raíces de la función 5.

$$i(t) = 10e^{-t/2}\cos(3t) - 2 \times 10^{-3} \tag{5}$$

La función 4 se le agrego un termino para subir el cero una cantidad de $2 \times 10^{-3}A$. Asi las raíces que se encuentren con la función 5 corresponderá para cuando la función 4 sea igual a $2mA$. A continuación estos instantes:

t
0.523512 s
1.570943 s
2.617747 s
3.665608 s
4.711686 s
5.760776 s
6.804783 s
7.857372 s
8.895491 s
9.958072 s

Cuadro 4: Instantes donde la intensidad es igual a $2mA$

3.4.3. Parte c: ¿En qué instantes de tiempo la intensidad de la corriente es máxima?

Necesitamos saber para cuales instantes de tiempo la intensidad es máxima. Para calcular los máximos de una función debemos aplicar el criterio de la primera derivada. Esta primera derivada es:

$$\frac{dF(x)}{dx} = -5 \cdot \exp\left(-\frac{x}{2}\right) \cdot \cos(3x) - 15 \cdot \exp\left(-\frac{x}{2}\right) \cdot \sin(3x) \tag{6}$$

Entonces el código 4, nos da que para los siguientes instantes la intensidad es máxima. Es decir, los instantes donde la función 6 se anula.

t
1.987145 s
4.081540 s
6.175935 s
8.270331 s

Cuadro 5: Instantes donde la intensidad es máxima en los primeros 10s