Tarea 4

Carmelo García
Física Computacional

1 de diciembre de 2024

1. Problemas

1. Cierto diodo se comporta como un condensador cuya capacidad C depende de la tensión V entre sus extremos, según la expresión:

$$C(V) = \frac{C_0}{(1 + |V|/\varphi)^\gamma},$$

donde $C_0 = 81 \times 10^{-12}$ F, es la capacidad nominal del diodo, $\varphi = 0{,}6$ V y $\gamma = 0{,}44$, son parámetros característicos del mismo.

a) Halle la carga Q del diodo, cuando la tensión entre sus extremos es de 1 V.

b) Represente gráficamente en xmgrace la carga Q en función de V para valores de la tensión entre 0 y 5 V.

Para analizar el comportamiento del diodo, se necesita conocer la tensión V en función de la carga Q, lo que lleva a la ecuación:

$$V = \frac{Q}{C} = \frac{Q}{C_0(1 + |V|/\varphi)^\gamma},$$

$$C_0 V - Q(1 + |V|/\varphi)^\gamma = 0,$$

donde V es la incógnita.

c) Para un valor $Q = 10^{-10}$ C, determina con precisión de millonésimas el valor de V correspondiente, utilizando el método de Newton.

d) Resuelva el problema por el método de Punto Fijo.

2. Dada la ecuación no lineal:

$$8x - \cos(x) - 2x^2 = 0.$$

a) Haga una representación gráfica con xmgrace, para averiguar el número de soluciones (raíces) y aplique el método incremental para obtener una estimación inicial de las mismas.

b) Resuelva la ecuación utilizando los métodos Newton y secante, con una tolerancia de 10^{-5}.

c) Transforme la ecuación en una de la forma $x = g(x)$, para que la solución converja con un número de iteraciones similar al obtenido con el método de Newton. Aplique para esto, el método de Punto Fijo.

2. Soluciones

2.1. Problema 1

El código en C mostrado a continuación realiza un análisis del comportamiento de un diodo cuya capacidad varía con la tensión aplicada. Incluye la implementación de métodos numéricos iterativos, como el método de Newton-Raphson y el método de Punto Fijo, para encontrar la raíz de una ecuación no lineal que describe esta relación. Además, genera datos para una representación gráfica de la carga del diodo en función de la tensión y utiliza xmgrace para visualizar estos resultados. En resumen, el código proporciona una solución completa para calcular y representar gráficamente las características del diodo bajo diferentes condiciones de tensión, empleando técnicas numéricas avanzadas para obtener resultados precisos.

```
//Definimos la tolerancia
#define TOL 1e-8
// Definir las constantes
double C0 = 81e-12;
double phi = 0.6;
double gamma_val = 0.44;
double Q = 1e-10;
//Definimos la funciones a utilizar
double C(double V){
    return C0/pow((1 + fabs(V)/phi),gamma_val);
}
double f(double V){
    return V*C0 -Q*pow((1 + fabs(V)/phi),gamma_val);
}
double df(double V){
    return C0 -Q*gamma_val*pow((1 + fabs(V)/phi),gamma_val-1)/phi;
}
//Definimos la funcion g utilizada para el metodo de punto fijo
double g(double V){
    return Q/C(V);
}
//Definimos el metodo de newton
void newton(double x0){
    double x1;
    int count = 0;
    while (count < 100)
```

```
    {
        x1 = x0 - f(x0)/df(x0); //calculo del siguiente valor
        if (fabs(x1 - x0) < TOL)
        {
            printf("La raiz es: %lf\n", x1);
            printf("Numero de pasos: %d\n", count);//imprimimos la raiz y el numero de pasos
            break;
        }
        x0 = x1;//actualizamos el valor
        count++;//aumentamos el contador
    }
}
//Definimos el metodo de punto fijo
int punto_fijo(double x_ant) {
    double x_i;
    int count = 0;
    while (count < 100) {
        x_i = g(x_ant);//calculamos el siguiente valor
        if (fabs(x_i - x_ant) <= TOL) {
            printf("La raiz es: %lf\n", x_i);
            printf("Numero de pasos: %d\n", count);
            break;
        } else {
            x_ant = x_i;
            count++;
        }
    }
    if (count == 100) {
        printf("Se alcanzo el numero maximo de iteraciones sin convergencia.\n");
    }
    return 0;
}
void main(){
    //Parte a
    printf("Parte a:\n");
    double Q_0 = C(1);
    printf("La carga del diodo es: %e\n", Q_0);
    // Parte b
    printf("Parte b:\n");
    FILE *data; //creamos un archivo para guardar los datos
    data = fopen("datos.dat","w");//abrimos el archivo en modo escritura
    double x = 0;
    while (x <= 5)//calculamos los valores de Q para V entre 0 y 5
    {
        double q = x*C(x);
        fprintf(data,"%lf %e\n",x,q);
        x += 0.01;
    }
    fclose(data);//cerramos el archivo
    system("xmgrace datos.dat");//imprime el grafico en pantalla
    printf("Grafico generado\n");
    //Parte c
```

```
    printf("Parte C\n");
    newton(2);//llamamos al metodo de newton
    //Parte d
    printf("Parte D\n");
    punto_fijo(2);//llamamos al metodo de punto fijo
}

```

Listing 1: Problema 1.

2.2. Problema 2

El código en C resuelve una ecuación no lineal utilizando varios métodos numéricos, como el método de Newton-Raphson, el método de la secante y el método de Punto Fijo. Además, realiza una búsqueda incremental para encontrar valores iniciales donde la función cambia de signo, indicando posibles raíces. Para cada posible raíz encontrada, el código emplea los métodos mencionados para refinar y confirmar la raíz con la precisión deseada. También genera un archivo de datos que se usa para crear una representación gráfica de la función en el intervalo dado. El código emplea técnicas para gestionar memoria y de cálculo iterativo para asegurar la precisión y eficiencia en la resolución de la ecuación no lineal.

```
#define TOL 1e-5
int n_root = 0;
double* R = NULL; // Inicializar el puntero a NULL
double f(double x){
    return 8*x - cos(x) - 2*pow(x,2);}
\\ Definir las funciones g(x) posibles para el m todo de punto fijo
double g(double x){
    return (cos(x) + 2*pow(x,2))/8;
}
double g2(double x){
    return - cos(x) / (2 * x - 8);
}
double df(double x){
    return 8 + sin(x) - 4*x; }
double tolerancia(int n) { // Definir la tolerancia
    return 0.5 * pow(10, 2 - n); }
int punto_fijo(double x_ant) {
    double x_i;
    int count = 0;
    while (count < 100) {
        x_i = g(x_ant);
        if (fabs(x_i - x_ant) <= TOL) {
            printf("La raiz es: %lf\n", x_i);
            printf("Numero de pasos: %d\n", count);
            break;
        } else {
            x_ant = x_i;
            count++;
        }
    }
    if (count == 100) {
```

```
32          printf("Se alcanzo el n mero maximo de iteraciones sin
    convergencia.\n");
33      }
34      return 0;
35  }
36  // Definir la funcion para el m todo incremental
37  void incremental_search(double xl, double xu, double dx) {
38      int i, ndiv;
39      double x, testf, testdf;
40      ndiv = (int)((xu - xl) / dx);
41      for (i = 0; i <= ndiv; i++) {
42          x = xl + i * dx;
43          // Se mejoro la pregunta para el cambio, ya que antes era f(x) * f
    (x + dx) y fallaba
44          testf = f(x + 0.5 * dx) * f(x - 0.5 * dx); // pregunta con
    diferencias centradas si hay cambio de signo
45          testdf = df(x + 0.5 * dx) * df(x - 0.5 * dx); // pregunta con
    diferencias centradas si hay cambio de signo
46          if (testf < 0 || (fabs(f(x)) < TOL && testdf < 0)) {
47              n_root++;
48              R = realloc(R, n_root * sizeof(double)); // Redimensionar el
    array din micamente
49              if (R == NULL) {
50                  printf("Error al asignar memoria\n");
51                  exit(1); // Terminar si hay un error de asignacion de
    memoria
52              }
53              R[n_root - 1] = x; // Almacenar la raiz encontrada
54          }
55      }
56      printf("Raices metodo incremental:\n");
57      for (i = 0; i < n_root; i++) {
58          printf("R[%d] = %f\n", i, R[i]);
59      }
60  }
61  // Definir el metodo de Newton
62  void newton(double x0){
63      double x1;
64      int count = 0;
65      while (count < 100)
66      {
67          x1 = x0 - f(x0)/df(x0);
68          if (fabs(x1 - x0) < TOL)
69          {
70              printf("La raiz es: %lf\n", x1);
71              printf("Numero de pasos: %d\n", count);
72              break;
73          }
74          x0 = x1;
75          count++;
76      }
77  }
78  // Definir el metodo de la secante
79  void secante(double x0, double x1){
```

```
    double x2;
    int count = 0;
    while (count < 100)
    {
        x2 = x1 - f(x1)*(x1 - x0)/(f(x1) - f(x0)); //Formula del metodo de
    la secante
        if (fabs(x2 - x1) < TOL)
        {
            printf("La raiz es: %lf\n", x2);
            printf("Numero de pasos: %d\n", count);
            break;
        }
        x0 = x1;
        x1 = x2;
        count++;
    }
}
void main(){
    // Parte a
    FILE *data;//creamos un archivo para guardar los datos
    data = fopen("datos.dat","w");//abrimos el archivo en modo escritura
    double x = -2;
    while (x < 7)//calculamos los valores de f(x) para x entre -2 y 7
    {
        fprintf(data,"%lf %lf\n",x,f(x));
        x += 0.01;
    }
    fclose(data);//cerramos el archivo
    system("xmgrace datos.dat");//imprime el grafico en pantalla
    incremental_search(-10, 10, 0.1);//buscamos las raices con el metodo
   incremental
    printf("Raices encontradas:%d \n", n_root);//imprimimos el numero de
   raices encontradas
    // Parte b
    printf("Metodo de Newton\n");
    for (int i = 0; i < n_root; i++)
    {
        newton(R[i]);
    }
    printf("Metodo de la secante\n");
    for (int i = 0; i < n_root; i++)
    {
        secante(R[i] + 0.5, R[i] - 0.5);
    }
    // Parte c
    printf("Metodo Punto fijo\n");
    for (int i = 0; i < n_root; i++)
    {
        punto_fijo(R[i]);
    }
    // Liberar la memoria al final
    free(R);
    n_root = 0;//reiniciamos el contador de raices
}
```

```
131
```

Listing 2: Problema 2.

3. Resultados

3.1. Problema 1

En este problema nos centramos en estudiar la siguiente función.

$$C(V) = \frac{C_0}{(1 + |V|/\varphi)^\gamma} \tag{1}$$

3.1.1. a) Halle la carga Q del diodo, cuando la tensión entre sus extremos es de 1 V.

Para lograr este objetivo se despejó la carga Q, de la siguiente ecuación:

$$V = \frac{Q}{V} \qquad\qquad Q = V \times C \tag{2}$$

Sustituyendo los valores pertinentes, la carga es:

$$Q = 5{,}260885 \times 10^{-11} C$$

3.1.2. b) Represente gráficamente en xmgrace la carga Q en función de V para valores de la tensión entre 0 y 5 V.

Se graficó la ecuación para la carga (ecuación 2), en un rango de voltaje de $[0v, 5v]$. Se obtuvo el siguiente gráfico.

3.1.3. c) Para un valor $Q = 1010C$, determina con precisión de millonésimas el valor de V correspondiente, utilizando el método de Newton.

Para lograr este objetivo se manipularon las ecuaciones 1 y 2 para llegar a la ecuación siguiente:

$$C_0 V - Q(1 + |V|/\varphi)^\gamma = 0 \tag{3}$$

Con esto, se sustituyó el valor de Q por el pedido por el problema y nos quedó un problema de raíces nuevamente. Nuestra variable ahora es el voltaje. Por el método de Newton el resultado es el siguiente:

$$V = 2{,}566887V$$

Al método de Newton le tomó 3 iteraciones para llegar a este resultado.

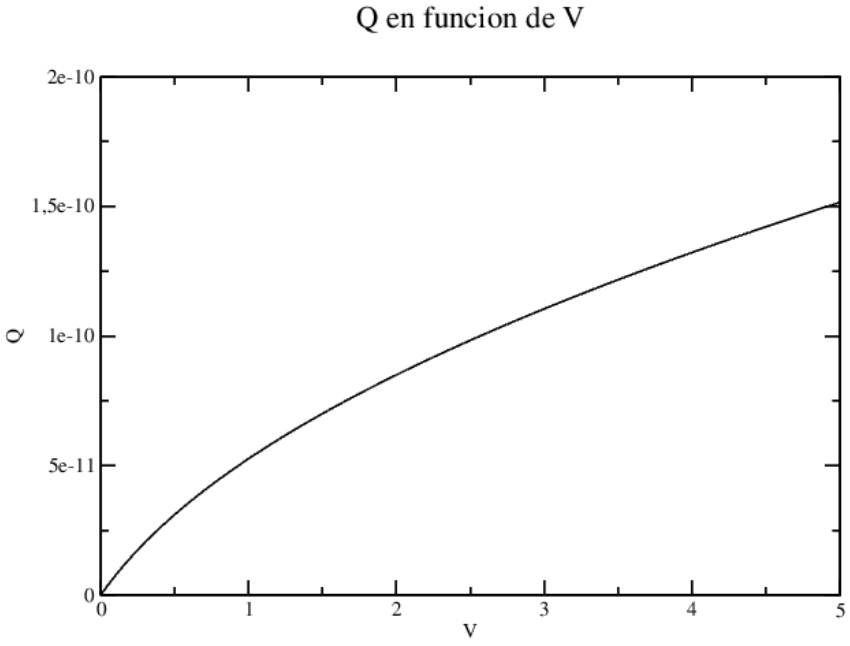

Figura 1: Carga un función del Voltaje

3.1.4. Resuelva el problema por el método de Punto Fijo.

Se hizo el mismo procedimiento de la sección anterior, pero ahora se usó el método del punto fijo. El resultado es el siguiente:

$$V = 2{,}566887V$$

Al método de punto fijo le tomó 17 iteraciones para llegar a este resultado.

3.2. Problema 2

La función a estudiar fue:

$$8x - \cos(x) - 2x^2 = 0 \tag{4}$$

3.2.1. a) Representación gráfica y estimación de raíces por método incremental

Se uso `xmgrace` para representar gráficamente el archivo `.dat` que arrojo el código 4.

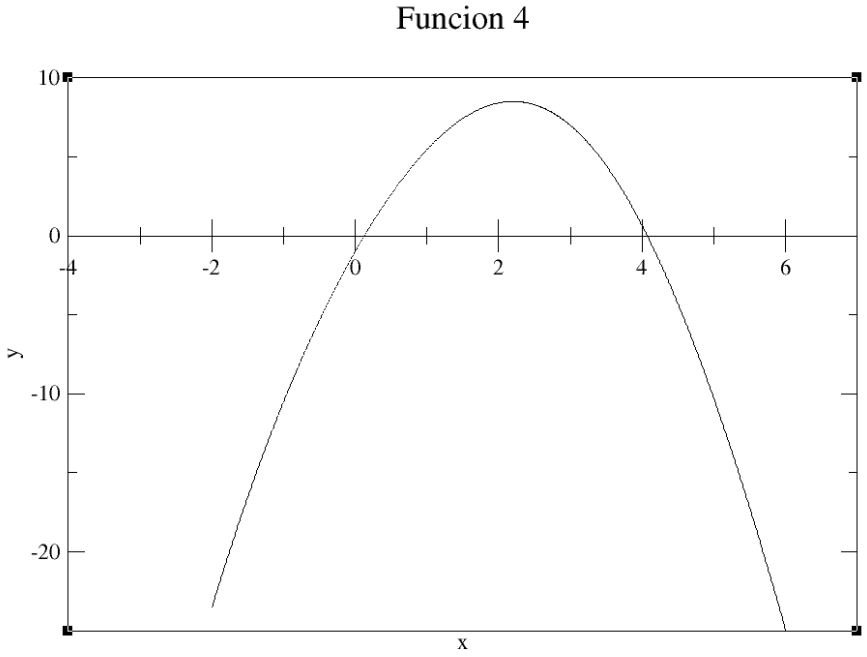

Figura 2: Representación gráfica de la función 4

En la figura 2 se puede notar que hay solo dos raíces, una cercana a cero y la otra cercana a 4. Con el método incremental se puede estar mas seguro de donde se encuentran las raíces.

`n_root`	Raíz
1	0.1
2	4.1

Cuadro 1: Estimación de raíces por el método incremental.

3.3. b) Resuelva la ecuación utilizando los métodos Newton y secante.

Se usaron las estimaciones del método incremental como parámetros para las funciones `newton(double xo)` y `secante( double x0 , double x1 )`. En el caso de la función para

el método secante, se uso una sola estimación de la raíz en ambos argumentos de la función con la salvedad que estaban desfasados medio paso, es decir $R[i] \pm 0{,}5$. Esto arrojo los siguientes resultados.

`n_root`	Raíz	Pasos	Metodo
1	0.128077	2	Newton
2	4.073225	2	Newton
1	0.128077	4	Secante
2	4.073225	4	Secante

Cuadro 2: Comparación de los métodos de Newton y secante con una tolerancia de 10^{-5}

Con esto podríamos determinar que el método de Newton converge mas rápido que el método de la secante. Esto se debe a que el método de Newton tiene una convergencia cuadrática. Esto significa que el número de dígitos correctos en la aproximación de la raíz aproximadamente se duplica en cada iteración, siempre que la aproximación inicial esté cerca de la raíz y la función sea suficientemente suave. En cambio, la convergencia del método de la secante es superlineal.

3.4. c) Resuelva el problema por el método de Punto Fijo.

En este problema preciso, posiblemente debido a que las raíces de la función están relativamente cerca una de la otra. No se pudo encontrar una función $g(x)$ que con la que se pueda obtener las dos raíces. Así que se encontraron 2 funciones g que encuentra una raíz cada una.

$g(x)$	Ruiz	Pasos
$\frac{\cos(x)+2x^2}{8}$	0.128077	2
$-\frac{\cos(x)}{2x-8}$	4.073225	3

Cuadro 3: Raíces a través del método del punto fijo.

Las funciones necesarias para este calculo son:

$$g(x) = \frac{\cos(x) + 2x^2}{8} \tag{5}$$

$$g(x) = -\frac{\cos(x)}{2x - 8} \tag{6}$$

Estas dos calculadas a partir de la función 4.