

Solución de Ecuaciones no Lineales aplicando Métodos Numéricos

Jaime J. Luque Quispe

Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos

j.luque@ieee.org

Abstract—This paper discusses the solution of nonlinear equations using numerical analysis. The algorithms are implemented in MATLAB. MATLAB is a language of technical computing.

I. INTRODUCCION

ESTE trabajo trata sobre la aplicación de algunas técnicas para resolver ecuaciones no lineales en una incógnita, se recurre a los conceptos básicos del cálculo y a herramientas computacionales.

Se aplicarán diversos métodos con el objetivo de calcular los ceros o raíces aproximadas de una función dada, como el método gráfico, el de bisección, iteración de punto fijo, método de Newton Raphson, Newton modificada y el de la secante.

Es imposible pensar en cálculos de ingeniería donde no estén involucradas funciones $f(x) = 0$ que para hallar sus soluciones no son aplicables las conocidas técnicas algebraicas de despejar, las fórmulas de Cardano, de Ferrari para polinomios, etc. Para estas situaciones se hace uso de los métodos numéricos con el objetivo de hallar valores aproximados de las soluciones exactas.

II. METODOS

A. Método de la Bisección

El método de bisección está basado en el Teorema del valor intermedio. Se basa en el resultado que afirma que si una función continua toma valores opuestos en los extremos de un intervalo entonces necesariamente se anula en un punto de dicho intervalo. La idea es entonces construir subintervalos de longitud cada vez menor que contengan al cero.

Procedimiento:

1. Empezamos con un intervalo $[a, b]$ y una función f que toma valores con signos opuestos en a, b

2. Se calcula $c = (a + b)/2$.
3. Se compara los signos de $f(a), f(b), f(c)$.
Si $f(a)$ y $f(c)$ tiene signos opuestos se toma $b = c$
Si $f(a)$ y $f(c)$ tiene el mismo signo se toma $a = c$
4. Se vuelve al punto 1.

Añadiendo a esto una condición que indique cuándo una solución aproximada se considera suficientemente buena, considerando también un control sobre el número máximo de iteraciones. El algoritmo es el siguiente:

```
Datos de entrada: a, b, f, eps, nmax
fa=f(a), fb=f(b)
Para i=1,2,...,nmax
c=(a+b)/2
fc=f(c)
si abs(fc)<eps entonces
    break
fin si
si fa*fc<0 entonces
    fb=fc
    b=c
sino
    fa=fc
    a=c
fin si
fin para
```

B. Método de Newton

Es un método más potente para la resolución de ecuaciones no lineales. El precio que se paga es una pérdida del carácter general de convergencia: el método converge habitualmente sólo si se arranca desde un punto situado suficientemente cerca de la raíz. Por ello suele ser habitual utilizar este método en combinación con un método menos potente pero más seguro. Así por ejemplo, el método de Bisección permite acercarse a la raíz, de forma que tras algunos pasos, lanzar el método de Newton con la aproximación dada por el método de Bisección. Aún así, no se asegura la convergencia del método. Por ejemplo

es esperable que haya problemas si la función no es derivable en el cero de la función. El método de Newton viene dado por el siguiente esquema: dado x_0 , aproximación inicial de la solución, definir:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$n = 0, 1, 2, \dots$

```
Datos xo, f, eps, nmax
Define df como la derivada de f
Hacer fxo=f(xo)
Para i=1,2,...,nmax
  xo=xo-fxo/df(xo)
  fxo=f(xo)
  si abs(fxo)<eps entonces
    break
  fin si
fin para
```

El algoritmo anterior no guarda los valores de la sucesión, simplemente el último valor calculado.

C. Método de la Secante

El Método de la Secante trata de solventar una de las principales dificultades del Método de Newton, que este requiere conocer la derivada de f . En situaciones prácticas, es posible que la derivada de f no se pueda calcular, o bien que su evaluación resulte demasiado complicada (computacionalmente). El Método de la Secante consiste en el siguiente esquema: dado x_0, x_1 dos valores suficientemente cerca de la raíz, definir:

$$x_{n+2} = x_{n+1} - \frac{x_{n+1} - x_n}{f(x_{n+1}) - f(x_n)} f(x_{n+1})$$

Se observa que a diferencia del método de Newton, se necesitan ahora dos valores iniciales, x_0 , y x_1 , para arrancar el método. A continuación se expone un algoritmo sobre el que se podría trabajar para su implementación en el ordenador.

```
Datos xo, x1, f, eps, nmax
Hacer fxo=f(xo), fx1=f(x1)
Para i=1,2,..., nmax
  x2=x1-(x1-xo)/(fx1-fxo)*fx1
  fx2=f(x2) si abs(fx2)<eps entonces
    break
  fin si
Hacer xo=x1, fxo=fx1
Hacer x1=x2, fx1=fx2
fin para
```

Se observa que con esta implementación cada iteración precisa únicamente de una evaluación de la función f ya que algunos valores de la iteración anterior se reutilizan de nuevo. El algoritmo pierde algo de su potencia (menor orden de convergencia) cuando se compara con el método de Newton, es decir, requiere de mayor número de iteraciones pero cada iteración se realiza en menor tiempo (observe que el método de Newton evalúa la función y su derivada una vez en cada iteración). De nuevo para arrancar el método es necesario que x_0 y x_1 estén cerca de la solución.

D. Método de Punto fijo

Sea la ecuación general $f(x)=0$. El primer paso consiste en transformar algebraicamente a la forma equivalente $x=g(x)$. Una vez que se ha determinado la forma equivalente, el siguiente paso es tantear una raíz, esto puede hacerse por observación directa de la ecuación. Se denota el valor de tanteo o valor de inicio como x_0 . Una vez que se tiene x_0 se evalúa $g(x)$ en x_0 , denotándose el resultado de esta evaluación como x_1 ; esto es; $g(x_0)=x_1$ y se procede a una segunda evaluación de $g(x)$; ahora en x_1 ; denotándose el resultado como x_2 ; $g(x_1)=x_2$. Este proceso es iterativo.

Aunque hay excepciones, generalmente se encuentra que los valores de x_0, x_1, x_2, \dots se van acercando a x' de manera que x_i está más cerca de x' que x_{i-1} , o bien mas alejado de x' e modo que cualquiera esta mas lejos que el valor anterior.

Finalmente, para determinar si la sucesión x_0, x_1, x_2, \dots esta convergiendo o divergiendo de una raíz x' , cuyo valor se desconoce, puede calcularse la sucesión $f(x_0), f(x_1), f(x_2), \dots$. Si dicha sucesión tiende a cero, el proceso iterativo converge a x' y dicho proceso se continuará hasta que $|f(x_i)| < \text{eps}$, donde eps es un valor pequeño e indicativo de la exactitud o cercanía de x_i con x' . Se toma a x_i como la respuesta y el problema de encontrar una raíz real habrá concluido. Si por el contrario $f(x_0), f(x_1), f(x_2), \dots$ no tiende a cero, la sucesión x_0, x_1, x_2, \dots diverge de x' , y el proceso deberá detenerse y ensayarse uno nuevo con una $g(x)$ diferente.

III. ESTRUCTURAS EN MATLAB

BUCLES, EL COMANDO FOR

En matlab, la estructura:

```
for j=vi : p : vf
  .....
end
```

Implementa un bucle donde las líneas de código entre `for` y `end` son ejecutadas repetidamente para cada j que varia desde un valor inicial vi a un valor final vf con un paso igual a p .

Por ejemplo,

```
for j=7:2:13; disp(j); end
7
9
11
13
```

BREAK Y CONTINUE

Asociados a for, y en general a cualquier bucle, encontramos los comandos break y continue.

Break fuerza la salida inmediata del bucle. Continue provoca el fin de la ejecución actual del bucle con el valor actual (j) y que se comience con el siguiente valor (j+1).

OPERADORES LÓGICOS Y ESTRUCTURAS DE DECISIÓN

Los operadores lógicos más básicos en MATLAB son

== igualdad, ~= desigualdad, > mayor, < menor,
>= mayor o igual, <= menor o igual, & “y” lógico, | “o” lógico

El resultado de una comparación es 1 si es verdadero, 0 si es falso:

```
>> a=3; b=2;
>> a>0
ans =
1
>> a>0 & b<3
ans =
1
>> a<0 | b<1
ans =
0
```

IF

Se utiliza para representar condiciones

```
if (condición)
    sentencia(s)
end
```

```
if (a>0 & b>0)
    disp('ab es positivo')
```

ELSE

Las sentencias dentro de else se ejecutan en caso de que la condición lógica propuesta en if no se satisfaga

```
if (condición)
    sentencia(s)
end
else
    sentencia(s)
end
```

```
if (a<0)
    disp('a es número positivo')
else
    disp('a es número negativo')
end
```

IF DE DECISIÓN MÚLTIPLE

EJEMPLO: FUNCIÓN G DEFINIDA EN INTERVALOS

```
if (x<0)
    g=x^2;
elseif (x>=0&x<1)
    g=2+log(x^2);
elseif (x>=1&x<2)
    g=x;
else
    g=sin(x/2);
end
```

Esta última instrucción es equivalente a:

```
if (x<0)
    g=x^2;
else
    if (x>=0&x<1)
        g=2+log(x^2);
    else
        if (x>=1&x<2)
            g=x;
        else
            g=sin(x/2);
        end
    end
end
```

A esta última forma de utilizar los if se llama anidar los if. Obviamente, la primera forma es más clara y concisa.

WHILE

El comando while nos permite ejecutar un bucle mientras una condición sea verdadera, a diferencia de for que para utilizarlo se tiene que definir el número de veces en que se ejecutará las sentencias contenidas en el bucle. Por ejemplo, el siguiente código:

```
dist=1;
eps= 0.0001; xo=1;
while dist>eps
    x=sin(xo);
    dist=abs(x-xo)
    disp(xo)
    x=xo;
end
```

El código anterior corresponde a la aplicación del método iterativo de punto fijo donde se intenta hallar una raíz de la función $g=x-\sin(x)$. Se muestra en pantalla los valores de cada x_0 . x_0 es un número que se actualiza en cada iteración y se va aproximando a la raíz real.

Estos comandos se utilizan cuando programamos scripts y funciones para la solución de problemas (ver ejercicios resueltos).

REFERENCIAS

- [1] A. Nieves, F. Dominguez “Métodos Numéricos Aplicados a la Ingeniería” 2nd ed. Ed. Compañía Editorial Continental, México 2003. pp. 29–117.
- [2] J. Kiusalaas “Numerical Methods in Engineering with Matlab”. Cambridge University Press
- [3] V. Dominguez “Prácticas de Matemáticas” Universidad Pública de Navarra