Computación numérica
Laboratorio n° 2

J. L. López[a]

[a]*Centro de Innovación en Ingeniería Aplicada, Universidad Católica del Maule, Talca - Chile.*

Abstract

En este laboratorio trabajaremos los conceptos básicos de la computación numérica en cuanto a su aplicación en la resolución de ecuaciones no-lineales. Los problemas ejemplos incluyen (i) método de bisección (ii) Newton-Raphson, y (iii) método de la tangente, en donde se explora la convergencia de los métodos y su orden, junto con condiciones para detener el proceso iterativo.

Keywords: Iteraciones, ecuaciones no lineales, Convergencia, errores.

1. Introducción

Los métodos para resolver ecuaciones no-lineales que se aprovechan del cambio de signo de una función en la vecindad de una raíz se les conoce como métodos cerrados, o de intervalos, porque se necesita de dos valores iniciales para la raíz. Dichos valores iniciales deben encerrar a la raíz. Los métodos cerrados emplean diferentes estrategias para reducir sistemáticamente el tamaño del intervalo y así converger a la respuesta correcta.Por el contrario, los métodos abiertos siempre generan aproximaciones cada vez más cercanas a la raíz. Se dice que tales métodos son convergentes porque se acercan progresivamente a la raíz a medida que se avanza en el cálculo.

2. Raices reales de ecuaciones No-Lineales

Un método simple para obtener una aproximación a la raíz de la ecuación $f(x) = 0$ consiste en graficar la función y observar dónde cruza el eje x. Este punto, que representa el valor de x para el cual $f(x) = 0$, ofrece una aproximación inicial de la raíz. Las técnicas gráficas tienen un valor práctico limitado, ya que no son precisas. Sin embargo, los métodos gráficos se utilizan para obtener aproximaciones de la raíz. Dichas aproximaciones se pueden usar como valores iniciales en los métodos numéricos analizados en este capítulo y en el siguiente.

Email address: jlopez@ucm.cl (J. L. López)

2.1. Método de la bisección

En general, si $f(x)$ es real y continúa en el intervalo que va desde x_a hasta x_b y $f(x_a)$ y $f(x_b)$ tienen signos opuestos, es decir, $f(x_a) * f(x_b) < 0$ entonces hay al menos una raíz real entre x_a y x_b. La localización del cambio de signo (y, en consecuencia, de la raíz) se logra con más exactitud al dividir el intervalo en varios subintervalos. Se investiga cada uno de estos subintervalos para encontrar el cambio de signo. El proceso se repite y la aproximación a la raíz mejora cada vez más en la medida que los subintervalos se dividen en intervalos cada vez más pequeños. Sin embargo, queda pa tarea de establecer un criterio objetivo para decidir cuándo debe terminar el método y para lo cual el cálculo del error podría servir para detener el proceso iterativo, siempre y cuando este se encuentre por debajo de cierto nivel establecido.

2.1.1. Problema de formulación matemática

Un sistema eléctrico posee un dispositivo cuyo almacenamiento de energía se rige por la ecuación $f(t) = e^t - \pi t$. Se requiere calcular los tiempos límites para los cuales el almacenamiento electrico llega a cero. Suponga que el sistema eléctrico deja de trabajar cuando el tiempo límite logra una estabilidad menor al 1%.

Realice las siguientes actividades:

1. Resuelva el problema utilizando el método de la bisección, Newton-Raphson y el método de la secante.
2. Estudie la convergencia de las soluciones para cada método iterativo.

2.1.2. Solución

El problema requiere resolver la ecuación $e^t - \pi t = 0$, y cuyo criterio de truncamiento para método iterativo corresponde a un error absoluto, entre sucesivas iteraciones, menor al 0.1%. recuerde que el error porcentual entre iteraciones sucesivas es dado por:

$$\epsilon\% = \frac{|x_{i+1} - x_i|}{|x_{i+1}|} * 100 \tag{1}$$

Para definir el intervalo de estudio $[a, b]$ para los ceros de la función, podemos realizar un análisis gráfico y así determinar los valores de las cotas inferior (a) y superior (b). Para graficar la función escribimos las instrucciones siguientes:

```
≫ from sympy import Symbol
≫ t = Symbol('t')
≫ f = exp(t) - π * t
≫ Plot(f,(t,0,2))
```

Para resolver la ecuación mediante el método de la bisección, escribimos las instrucciones siguientes:

```
≫ def f(t): return exp(t) − π ∗ t
≫ a=0
≫ b=2
≫ e=0.1
≫ r=0
≫ S=[a]
≫ Error=[]
≫ relativo = [np.absolute((b-a)/(b))]
≫ while relativo[r]>= e:
≫ print("Error relativo =",relativo[r])
≫ r = r+1
≫ c=(a+b)/2
≫ print("La solución aproximada en la i-esima iteración es t=",c)
≫ if f(c)==0:
≫ print("La solución obtiene en t=",c)
≫ else:
≫ if f(a)*f(c)¿0:
≫ a=c
≫ else:
≫ b=c
≫ Anterior = r-1
≫ S.append(c)
≫ absoluto = np.absolute(S[r]-S[Anterior])
≫ Error.append(absoluto)
≫ print("Error absoluto =",absoluto)
≫ relativo.append(np.absolute((absoluto/S[r])*100)
≫ print("Error porcentual =",relativo)
```

Para estudiar gráficamente la convergencia del método utilizamos las siguientes instrucciones:

```
≫ plt.plot(Error)
≫ plt.plot(S)
≫ plt.plot(relativo)
```

Para resolver la ecuación mediante el método de la Newton-Raphson, escribimos las instrucciones siguientes:

```
≫ from sympy import*
≫ t=Symbol('t')
≫ f=exp(t) − π ∗ t
≫ e=0.00001
≫ c = 0
≫ r=0.5
≫ S=[r]
≫ Error=[1]
≫ relativo = [1]
≫ df = diff(g,t)
≫ while relativo[c]>= e:
≫ c = c + 1
≫ r = r- float(g.subs(t,r))/float(df.subs(t,r))
≫ S.append(r)
≫ print(r)
≫ Anterior = c-1
≫ absoluto = np.absolute(S[c]-S[Anterior])
≫ rela = np.absolute((S[c]-S[Anterior])/S[c])
≫ relativo.append(rela)
≫ Error.append(absoluto)
≫ print("Error absoluto =",absoluto)
≫ print("Error relativo =",rela)
≫ print(S)
```

donde r contiene la estimación de la raiz para $f(t) = 0$, la cual es modificada sucesivamente en cada iteración. Este proceso, al igual que con el método de la bisección, se puede utilizar para obtener los resultados y estudiar la convergencia.