Elementos de Cálculo Numérico / Cálculo Numérico

Primer Cuatrimestre 2020

Décimo ejercicio computacional
Lunes 06/06/20 al Lunes 13/07/20
Recuerde subir el archivo en formato ejercicioX_NOMBREAPELLIDO.py
Recuerde enviar su código al hacer consultas

En este ejercicio construiremos dos buscadores de raices de funciones: el de Newton-Rhapson y el de punto fijo. A manera de resumen (revisar la teórica-práctica puede ser buena idea), en el de Newton-Rhapson buscamos los ceros de la función apoyandonos en la derivada de la función. Si buscamos ceros de $f(x)$, entonces nuestra iteración será:

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

donde la convergencia de esta iteración está limitada principalmente por los ceros de $f'(x_n)$.

En el método de punto fijo, buscamos escribir $f(x) = 0$ como $g(x) = x$, y empleamos iteraciones de la forma:

$$x_{n+1} = g(x_n) \tag{2}$$

En este caso, la convergencia está limitada por las características de la imagen de $g(x)$. Sin mucho más, pongamos manos a la obra:

A Construya una función que reciba un valor de x y retorne el valor de

$$f(x) = (x+2)(x+1) \tag{3}$$

(que llamaremos `funcion_f`) y una que retorne el valor de la derivada de f (que llamaremos `funcion_fprima`). Asegurese de identificar donde están los ceros de f.

B Construya una funcion `iteracion_newton_rhapson(fun_f,fun_fprima,x0,max_iter,tol)`. donde `fun_f` será la funcion a la cual queremos encontrarle el cero y `fun_fprima` su derivada, `x0` la semilla inicial, `max_iter` el número máximo de pasos a realizar, y `tol` la tolerancia con la cual esperamos encontrar el resultado.

Puede usar el siguiente modelo:

```
def iterador_newton_rhapson(funcion_g,funcion_gprima,x0,max_iter,tol):
    error = abs(## COMPLETAR ##)
    iter = ## COMPLETAR
    while ## COMPLETAR ##:
        x1 = ## COMPLETAR ##
        error = abs(## COMPLETAR ##)
        iter = iter+1
        x0 = x1
    return([x0,iter,error])
```

Checkee esta función, empleando las funciones de punto A, y las semillas `x0=1` y `x0=-10`. ¿Tiene sentido el resultado?

C Mejore la función para ser capaz de emplear una aproximación numérica de `fun_fprima`. Para esto, considere que si `fun_fprima=='NO'`, debe generar una función que la aproxime numéricamente. Puede usar el modelo:

```
def iterador_newton_rhapson(fun_f,fun_fprima,x0,max_iter,tol):
    if fun_fprima=='NO':
        def funcion(x):
            x1 = x*1.001 # Por ejemplo, aproximo usando el 100.1%
            x0 = x*0.999 # y el 99.9% de x
            f1 = # COMPLETAR #
            f0 = # COMPLETAR #
            f_prima = # COMPLETAR #
            return(f_prima)
        fun_fprima= funcion
        # sigue como en punto B #
```

Compare el resultado obtenido con el punto anterior.

D Construya un iterador de punto fijo `iterador_punto_fijo(fun_g,x0,max_iter,tol)`. En este caso, `fun_g` será la función $g(x)$ y el resto de los parámetros son análogos a los mencionados en el punto B. Puede usar el siguiente modelo:

```
def iterador_punto_fijo(fun_g,x0,max_iter,tol):
    error = abs(# COMPLETAR #)
    iter = # COMPLETAR #
    while # COMPLETAR #:
        x1 = # COMPLETAR #
        error = abs(# COMPLETAR #)
        iter = # COMPLETAR #
        x0 = x1
    return([x0,iter,error])
```

Pruebe esta función con la función f empleada en el punto A. Para esto construya $g_1(x) = -\frac{2}{x+3}$ y $g_2(x) = -\frac{x^2+2}{3}$ (¿Por qué estas funciones?) ¿Cuál converge más rápido? ¿Puede explicar la razón usando lo visto en la teórica?

E Emplee ambos iteradores para ajustar los datos del ejercicio 20 de la práctica 9. En ambos casos use la semilla `x0=1`

Los datos:

```
x = np.arange(0,5.5,.5)
y = np.array([0.756,0.561,0.407,0.372,0.305,0.24,0.219,0.209,0.21,0.194,0.140])
```

La función a minimizar es la de cuadrados mínimos, tal como se escribe en la guía:

$$\min_b F(b) = \sum_{i=0}^{10} (y_i - \frac{1}{x_i + b})^2 \tag{4}$$

Con la función $g(b)$ para el método de punto fijo:

$$g(b) = \frac{1}{y_0 + \sum_{i=1}^{10} (y_i - \frac{1}{x_i+b})(\frac{x_0+b}{x_i+b})^2} - x_0 \tag{5}$$

Y la función $f(b)$ para el método de Newton-Rhapson:

$$f(b) = \sum_{i=0}^{10} (y_i - \frac{1}{x_i + b})(\frac{1}{x_i + b})^2 \tag{6}$$

Grafique ambos resultados y compare: ¿ambos resultados son identicos?¿Difieren mucho en la cantidad de pasos?

F Extra: Empleando los datos del problema 7, realize un ajuste considerando a `X` como el logaritmo del área de un tract y a `Y` como el logaritmo de la población. Para simplificar, normalice los datos a `X_norm` e `Y_norm`:

```
X_norm = (X-np.min(X))/(np.max(X)-np.min(X))
Y_norm = (Y-np.min(Y))/(np.max(Y)-np.min(Y))
```

Realice un ajuste de cuadrados mínimos de estos datos normalizados usando una función tipo potencia de x (note que desechamos el primer punto, ya que inevitablemente es cero y dificulta el ajuste):

$$\min_a F(a) = \sum_{i=1}^{10} (y_i - x_i^a)^2 \tag{7}$$

Emplee el iterador que usted prefiera y grafique. Compare gráficamente este resultado con el obtenido en el problema 9 computacional.