Ayudantía

SCT - Semana - 04

IPF y Método de Newton-Raphson

1. ¿Cuántas iteraciones del método de Newton-Raphson se requieren para encontrar la raíz de una función afín $f(x) = mx+n$? ¿Por qué tiene sentido el resultado anterior?

La raíz de $f(x)$ es

$$\begin{aligned} f(r) &= 0 \\ mr + n &= 0 \\ mr &= -n \\ r &= \frac{-n}{m}. \end{aligned}$$

Sabiendo, también, que la derivada de $f(x) = mx+n$ es $f'(x) = m$, se puede ejecutar el método de Newton-Raphson una vez para encontrar una "aproximación" x_1 a partir de un *initial guess* x_0:

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= x_0 - \frac{mx_0 + n}{m} \\ &= x_0 - \frac{mx_0}{m} - \frac{n}{m} \\ &= x_0 - x_0 - \frac{n}{m} \\ &= \frac{-n}{m} \\ &= r \end{aligned}$$

Se ve que Newton-Raphson convergió a la raíz $r = \frac{-n}{m}$ en una sola iteración, sin haber importado qué *initial guess* x_0 se iba a usar.

¿Por qué tiene sentido el resultado anterior? Geométricamente, si se entiende que el método de Newton-Raphson aproxima una función como una recta tangente en un punto y se busca en dónde esta recta intersecta al eje X (véase https://www.youtube.com/shorts/jN5TuHXBC1g), entonces se puede entender que, como $f(x) = mx+n$ es una recta, la recta tangente a cualquier punto es exactamente la misma función. Por lo tanto, Newton-Raphson va a converger a la raíz de la función en una sola iteración.

2. Considera la función $y = f(x) = xe^x$. Su inversa es conocida como la función W de Lambert o la función log-producto: $x = f^{-1}(y) = W(y)$. Esta función **no se puede expresar en términos de funciones elementales**. Esto provoca que no sea trivial hallar, por ejemplo, el valor a tal que $f(a) = ae^a = 2$ o, en otras palabras, $a = W(2)$.

Construye un algoritmo que, dado un entero n y un initial guess x_0, use n iteraciones del método de Newton-Raphson para aproximar aquel valor a tal que $ae^a = 2$.

Buscar el valor a tal que $f(a) = 2$ es equivalente a buscar la raíz de una nueva función $h(x) = f(x) - 2$: aquel valor a tal que $h(a) = 0$. **Es importante notar esto y realizar este cambio en la función**, porque buscar directamente la raíz de $f(x)$ aplicando Newton-Raphson es equivalente a buscar el valor r tal que $f(r) = 0$, lo cual no es lo que queremos.

Si $h(x) = f(x) - 2 = xe^x - 2$, su derivada es:

$$\begin{aligned} h'(x) &= (xe^x - 2)' \\ &= (xe^x)' - (2)' \\ &= (xe^x)' \\ &= (x)'e^x + x(e^x)' \\ &= 1 \cdot e^x + xe^x \\ &= e^x + xe^x \end{aligned}$$

Con esta información, podemos construir la iteración de punto fijo dictada por el método de Newton-Raphson:

$$\begin{aligned} x_{i+1} &= x_i - \frac{h(x_i)}{h'(x_i)} \\ &= x_i - \frac{x_i e^{x_i} - 2}{e^{x_i} + x_i e^{x_i}} \end{aligned}$$

con un *initial guess* x_0. Un buen candidato es $x_0 = 1$, pues $f(1) = 1 \cdot e^1 - 2 = e - 2 = 0.71828...$

Una posible implementación del algoritmo es:

```python
def find_a(n, x0):
    xi = x0
    for i in range(n):
        ex = np.exp(xi)  # para reutilizar calculos
        xex = xi * ex  # para reutilizar calculos
        f_xi = xex - 2  # f(xi)
        fp_xi = ex + xex  # f'(xi)
        xi = xi - f_xi / fp_xi
    return xi
```

Otra implementación más generalizable y corta, aunque sin reutilización de cálculos repetidos (calcular e^x o xe^x), es simplemente definir la función `newton_raphson` que recibe, como parámetros, la función `f`, cuya raíz se busca; su derivada `fp`; un *initial guess* `x0`; y una cantidad opcional `n` de iteraciones del método.

```python
def newton(f, fp, x0, n=10):
    xi = x0
    for i in range(n):
        xi = xi - f(xi) / fp(xi)
    return xi
```

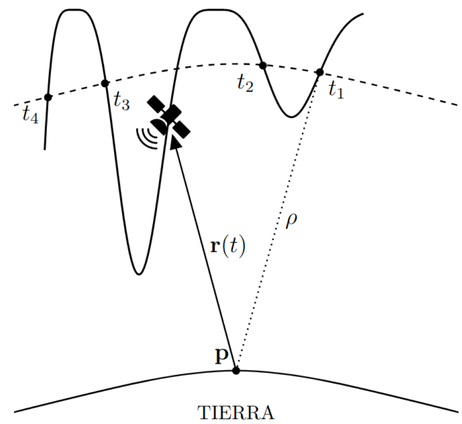
```python
>>> f = lambda x: x*np.exp(x) - 2
>>> fp = lambda x: np.exp(x) + x*np.exp(x)
>>> x0 = 1
>>> newton(f, fp, x0)
np.float64(0.8526055020137254)

```

3. Un satélite orbita alrededor de la Tierra, siguiendo una órbita descrita por un vector posición en función del tiempo: $\mathbf{r}(t) = (x(t), y(t))$.

Cuando este satélite está cerca de un punto de transmisión en la superficie terrestre, cuya posición es $\mathbf{p} = (p_x, p_y)$, pueden comunicarse y transmitir datos.

Si la distancia entre el satélite y el punto de transmisión es mayor a una distancia crítica ρ, entonces no hay transmisión de datos. Si la distancia entre el satélite en $\mathbf{r}(t)$ y el punto de transmisión en $\mathbf{p}$ es $\|\mathbf{r}(t) - \mathbf{p}\|_2 = \sqrt{(x(t) - p_x)^2 + (y(t) - p_y)^2}$, entonces la condición de transmisión es que $\|\mathbf{r}(t) - \mathbf{p}\| < \rho$.

A medida que el satélite orbita la Tierra, va entrando y saliendo de la zona crítica. Los tiempos t en los que entra o sale de la zona crítica se denotan $t_1, t_2, t_3, t_4, ..., t_k, ...$



Dado el último tiempo t_k donde el satélite entró en (o salió de) la zona crítica, el desafío es encontrar el próximo tiempo t_{k+1} donde volverá a salir (o entrar).

Los científicos que estudian este sistema conocen una estimación de la duración de tiempo Δ_k que transcurrirá entre t_k y t_{k+1}. Con esta estimación, pueden aproximar $t_{k+1} \approx t_k + \Delta_k$.

La idea es usar un algoritmo para refinar esta aproximación. En particular, usarán una función $f(t)$ cuyas raíces son los tiempos t_k y usarán métodos de búsqueda de raíces para encontrar la raíz t_{k+1} tal que $f(t_{k+1}) = 0$.

La función $f(t)$ que están utilizando es

$$f(t) = \left(\|\mathbf{r}(t) - \mathbf{p}\|_2^2 - \rho^2\right)^2,$$

sobre la cual están aplicando el **método de Newton-Raphson**, debido a que saben que converge cuadráticamente en la mayoría de los casos.

Sin embargo, **su método no está convergiendo cuadráticamente, sino linealmente: los científicos estiman una tasa** $S = \frac{1}{2}$. ¿Por qué no hay convergencia cuadrática? ¿Cómo corregirías el algoritmo para lograr convergencia cuadrática? Implementa, en Python y con ayuda de NumPy, este algoritmo corregido para aproximar el próximo t_{k+1} con convergencia cuadrática, dados los parámetros ρ, Δ_k, p_x, p_y y t_k, las funciones $x(t)$ e $y(t)$ para representar las coordenadas del satélite, sus derivadas $x'(t)$ e $y'(t)$, y la función `Newton1D(f, fp, x0, m=1)` que aplica el método de Newton sobre la función `f`, cuya derivada es `fp`, con *initial guess* `x0` y con la capacidad opcional de especificar la multiplicidad `m` de la raíz que se busca aproximar.

Si el método de Newton-Raphson no está convergiendo cuadráticamente, sino linealmente, esto indica que la derivada de f en la misma raíz también vale 0. Es decir, si t_{k+1} es una raíz de f tal que $f(t_{k+1}) = 0$, también se está cumpliendo que $f'(t_{k+1}) = 0$ y esta es la causa de la convergencia lineal de Newton-Raphson.

En esta situación, se dice que la raíz tiene multiplicidad mayor a 1. Específicamente, si todas las derivadas de f hasta la $m-1$ valen 0 en la raíz, es decir, $f(t_{k+1}) = f'(t_{k+1}) = f''(t_{k+1}) = ... = f^{(m-1)}(t_{k+1}) = 0$, entonces se dice que la raíz tiene multiplicidad m.

Una corrección al algoritmo tradicional de Newton-Raphson para lograr convergencia cuadrática, cuando la multiplicidad es m, es:

$$x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_i)}$$

Sabiendo que la tasa de convergencia lineal es $S = \frac{m-1}{m}$, un valor $S = \frac{1}{2}$ indica que la multiplicidad de la raíz t_{k+1} de f es $m = 2$.

En caso de no tener la tasa S, se podría, de todos modos, afirmar que la multiplicidad es, como mínimo, 2, como hicieron algunos estudiantes. El siguiente análisis, si bien no permite afirmar que la multiplicidad es $m = 2$, sino $m \geq 2$, puede servir de referencia para problemas similares. Por lo tanto, se dejará aquí, en caso de necesitarlo.

Se puede definir $g(t)$ como todo lo que se está elevando al cuadrado en $f(t)$:

$$\begin{aligned} g(t) &= \|\mathbf{r}(t) - \mathbf{p}\|_2^2 - \rho^2 \\ &= (x(t) - p_x)^2 + (y(t) - p_y)^2 - \rho^2 \end{aligned}$$

tal que $f(t) = (g(t))^2$. Si $f(t_{k+1}) = 0$, entonces $g(t_{k+1}) = 0$. Esto tiene sentido, pues t_{k+1} es el tiempo en que la distancia entre el satélite y el punto de transmisión es igual a ρ: $\|\mathbf{r}(t) - \mathbf{p}\|_2 = \rho$.

Si $f(t) = (g(t))^2$, entonces

$$f'(t) = 2g(t)g'(t),$$

donde $g'(t) = 2(x(t) - p_x)x'(t) + 2(y(t) - p_y)y'(t)$. Debido a que $g(t_{k+1}) = 0$, se da que $f'(t_{k+1}) = 0$: la multiplicidad de la raíz t_{k+1} es mayor a 1.

¿Será la segunda derivada también igual a 0 en t_{k+1}?

$$\begin{aligned} f''(t) &= 2g'(t)g'(t) + 2g(t)g''(t) && (1) \\ &= 2(g'(t))^2 + 2g(t)g''(t) && (2) \end{aligned}$$

Como $g(t_{k+1}) = 0$, se obtiene $f''(t_{k+1}) = 2(g'(t_{k+1}))^2$, lo cual solo puede ser 0 bajo una condición muy específica: que la velocidad del satélite sea nula o perpendicular a su posición relativa respecto del punto de transmisión, dado que $g'(t)$ se puede expresar como el producto punto entre ambos vectores: $g'(t) = (\mathbf{r}(t) - \mathbf{p}) \cdot \mathbf{r}'(t)$. En este caso, el satélite pasaría tangencialmente por la frontera crítica. En general, $f''(t_{k+1}) \neq 0$ y la multiplicidad de la raíz t_{k+1} es simplemente 2, pero, si se da la condición específica descrita anteriormente, la multiplicidad sería mayor.

Sabiendo que la multiplicidad es 2, el algoritmo corregido sería:

$$x_{i+1} = x_i - 2\frac{f(x_i)}{f'(x_i)}$$

donde $f(x) = (g(x))^2$, con $g(x) = (x(t) - p_x)^2 + (y(t) - p_y)^2 - \rho^2$. El *initial guess* sería $x_0 = t_k + \Delta_k$, la aproximación a t_{k+1} que propusieron los científicos. Idealmente, $\lim_{n\to\infty} x_n = t_{k+1}$.

El algoritmo luce así:

```python
def find_tkp1(rho, delta_k, p_x, p_y, t_k, x, xp, y, yp):
    # Funcion g tal que f(t) = (g(t))^2.
    g = lambda t: (x(t) - p_x)**2 + (y(t) - p_y)**2 - rho**2

    # Como f'(t) = 2g(t)g'(t), debemos definir g'(t).
    gp = lambda t: 2*(x(t) - p_x)*xp(t) + 2*(y(t) - p_y)*yp(t)

    f = lambda t: g(t)*g(t)
    fp = lambda t: 2*g(t)*gp(t)
    x0 = t_k + delta_k

    tkp1 = Newton1D(f, fp, x0, m=2)
    return tkp1
```