Análisis Numérico – ITIS

Descripción de la práctica 3: Raíces por Newton-Raphson

Se trata de obtener todas las raices de un polinomio de coeficientes reales o complejos:

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0, \quad a_i \in \mathbb{C}, \forall i = 0, 1, \ldots, n$$

mediante uno de los métodos vistos en clase. Sabemos por el teorema fundamental del Álgebra (clásica) que tiene exactamente n raices, contando multiplicidades. En éste caso utilizaremos el algoritmo de Newton-Raphson que tiene como función de iteración:

$$z_{k+1} = z_k - \frac{p(z_k)}{p'(z_k)}, \quad k = 0, 1, \ldots$$

Se ha de escribir un procedimiento MATLAB con forma parecida a:

```
function x=newton_raphson([an, ..., a1, a0],maxiter,tol)
```

que devuelve el vector x con las raices obtenidas, con parámetros de entrada el polinomio, número máximo de iteraciones (por cada raiz) y tolerancia.
Para obtener los valores del polinomio y su primera derivada utilizaremos el algoritmo de Ruffini-Horner visto al comienzo del cuatrimestre, con una pequeña modificación que permite obtener ambos valores con un solo bucle:

$$\begin{array}{c|ccccc} z_k & a_n & a_{n-1} & .. & a_1 & a_0 \\ \hline & b_n & b_{n-1} & .. & b_1 & \boxed{b_0} \\ & c_n & c_{n-1} & .. & \boxed{c_1} & \end{array}$$

En la salida, son $p(z_k) = b_0$ y $p'(z_k) = c_1$.
El algoritmo de N-R arranca de una aproximación inicial z_0 (para cada raiz que se pretenda obtener) que elegiremos de uno de estos dos modos:

1. Un elemento del vector $(1+i, i, -1+i, -1, -1-i, -i, 1-i, 1)$, que se obtiene considerando los vértices y cortes con los ejes de un cuadrado centrado en $(0, 0)$ (es $\|\mathbf{x}\|_\infty = 1,\ \mathbf{x} \in \mathbb{R}^2$).

2. Un elemento del vector cuyo k-ésimo es:

   $$x_k = \cos(k) + i \operatorname{sen}(k), \quad k = 1, \ldots, N$$

   que tiene sus elementos distribuidos entre los cuatro cuadrantes del plano y están sobre la circunferencia unidad ($\|\mathbf{x}\|_2 = 1,\ \mathbf{x} \in \mathbb{R}^2$). Se puede tomar $10 \leqslant N \leqslant 50$.

Partiendo entonces de un z_0 elegido de una de estas dos formas, después de unas pocas iteraciones (2 ó 3) se mira si el proceso es convergente (con esta aprox. inicial) para lo que debe cumplirse que $|z_{k+1} - z_k| < |z_k - z_{k-1}|$. Si no es el caso, hay que tomar como aprox. inicial el siguiente elemento del vector elegido y rearrancar el proceso.
Cuando se produzca la convergencia con la tolerancia prevista ($|z_{k+1} - z_k|/|z_{k+1}| <$`tol`, y también se ha de dar que $|p(z_{k+1})|$ sea casi cero), se almacena esta raiz en el vector de soluciones y se comienza todo el proceso de nuevo con el siguiente polinomio deflacionado, el resultante de R-H: $(b_n, \ldots, b_1)$ que es de grado $n-1$. Y así sucesivamente hasta que tengamos un polinomio deflacionado de grado uno, del cual podemos ya obtener directamente la raiz.
Las raices que resulten de los polinomios deflacionados de grados $n-1, n-2, \ldots, 1$ obtenidos por R-H se han de refinar realizando algunas iteraciones de N-R con el polinomio original. Para ello, se considera cada una de ellas como aproximación inicial a la raiz verdadera y se realiza R-H con el polinomio original. Con una o dos iteraciones debería bastar para obtener la raiz con la precisión requerida.
Hay varios ejemplos para pruebas en la página web.También, en MATLAB existe la función `roots` que nos calcula las raices del polinomio que se le pasa. Y aún mejor, existe la función `poly` que nos da los coeficientes del polinomio que tiene las raices que le pasamos, p.e.

```
p1=poly([i/3, -i/2, 2+sqrt(2)*i, 1/4-sqrt(3)*i, 2/3, 3/4+i])
```

que luego le pasaríamos a `newton_raphson` para comprobar que nos calcula correctamente las raices especificadas en `poly` (`newton_raphson(p1,10,1e-14)` , por ejemplo).