UNIVERSIDAD DE MURCIA

Departamento de Matemáticas
Cálculo Numérico en una Variable
Curso 2023/24

Prof. S. Sánchez-Pedreño

Práctica 7. Puntos fijos y raíces de polinomios

11 y 15 de diciembre de 2023

Funciones ya programadas en Octave

Para realizar esta práctica necesitas descargar algunos script de Octave que he subido al Aula Virtual. Si no lo has hecho ya, descarga el archivo `Practica7_AnadirBiblioteca.zip`, situado en la carpeta `CodigoPracticas` de los Recursos del AV. Descomprímelo y guarda su contenido en la carpeta `Biblioteca` que debe estar contenida en tu carpeta de trabajo con Octave de tu ordenador personal.

Una vez realizada la tarea anterior, en `Biblioteca` dispones de las funciones siguientes.

- `puntofijo.m`, para aproximar un punto fijo de una función mediante iteración de la misma. Espera como variables la función inicial `f`, el punto inicial de la iteración `x`, `tol` y `maxit`, la tolerancia y el número máximo de iteraciones, análogos a los de la función `newton.m` de la práctica anterior, y, finalmente, `imprime`, que también estaba incluido en `newton.m` y que permite, si toma el valor `1` o `true`, escribir en la ventana de comandos los valores de las iteraciones intermedias. Devuelve la lista `[x,fval,npasos]`, que contiene la aproximación al punto fijo `x`, el valor de la función en dicha aproximación `fval` y el número de iteraciones efectuadas `npasos`.
- `puntofijoAitken.m`, espera las mismas variables que punto fijo y devuelve los mismos valores que ésta, y un cuarto valor `neval` que proporciona el número total de evaluaciones de la función.
- `deflacion.m`, para dividir un polinomio por un monomio $x-a$. La función requiere dos argumentos: `p`, el polinomio, y `a`. La función devuelve una lista `[poldef,val]`, donde `poldef` es el polinomio cociente $q(x)$ y `val` es el valor de $p(a)$, de modo que $p(x) = q(x)(x-a) + p(a)$. Naturalmente, la función implementa el método de Horner.
- `newtonPoly.m`, que implementa el método de Newton para polinomios (reales o complejos). Sus argumentos son: `p`, el polinomio al que buscamos una raíz: `x0`, el punto inicial para el método de Newton; `tol`, `maxit` e `imprime`, análogos a los ya descritos antes. La función devuelve `[x,pz,npasos]`, que son, respectivamente, la aproximación a la raíz obtenida, el valor del polinomio en dicha aproximación y el número de iteraciones que han sido necesarias para obtenerla.
- `newtonPolyAleatorio.m`. Es una función que intenta aproximar todas las raíces de un polinomio, utilizando el método de Newton, con puntos iniciales elegidos al azar dentro del disco que contiene a todas las raíces. Sus argumentos son: `p`, el polinomio; `tol` y `maxit`, la tolerancia y el número máximo de iteraciones del método; `intentos` es el número máximo de puntos iniciales elegidos al azar que, naturalmente, debe ser mayor igual que el grado de `p`; finalmente, `imprime` tiene el mismo significado que en los otros casos. Cuando, elegido un punto inicial al azar, el método no converge (de acuerdo a los valores de `tol` y `maxit`), la función elige otro punto inicial al azar, siempre que no se haya alcanzado el número máximo de intentos dado por `intentos`. El proceso que sigue la función es deflacionar el polinomio a partir de cada raíz aproximada y, después, refinar la aproximación tomando la obtenida como nuevo punto inicial del método aplicado al polinomio inicial. La función devuelve la lista `[raices,nintent]`, donde `raices` es el vector de raíces aproximadas encontradas y `nintent` es el número de puntos iniciales que se han tomado para encontrar las `raices`. Obviamente la función no garantiza la localización de todas las raíces.

Antes de utilizar cualquiera de estas funciones, asegúrate de entender qué argumentos espera, su tipo y significado, así como qué salidas produce, su significado, orden y formato.

Ejercicios

1. Tratamos de encontrar la solución de la ecuación $x^3 + x = 1000$ que se encuentra en el intervalo $[9, 10]$ mediante uno de los siguientes procesos iterativos:

$$a)\ x_{n+1} = 1000 - x_n^3, \quad b)\ x_{n+1} = \frac{1000 - x_n}{x_n^2}, \quad c)\ x_{n+1} = \sqrt[3]{1000 - x_n}$$

   Antes de recurrir a la función `puntofijo.m`, representa gráficamente en sendas ventanas las gráficas de las funciones junto con la diagonal $y = x$. A la vista de las gráficas, ¿eres capaz de de saber cuál de los tres métodos ofrecerá una convergencia más rápida? Si no lo tienes claro, dibuja en otras ventanas las gráficas del valor absoluto de las derivadas.

   Finalmente aproxima la solución buscada con una tolerancia de 10^{-14}. ¿Cuántos iteraciones han sido necesarias?

   Para terminar repite la aproximación al punto fijo de la misma función utilizando el método de aceleración de Aitken, con la misma tolerancia. Compara el número de iteraciones y el número de evaluaciones de la función con las efectuadas en el método anterior.

2. Este ejercicio está concebido para ir experimentando y, en función de los resultados de ciertos cálculos, ir añadiendo código en el script, hasta tener la respuesta completa. Se trata de encontrar, de forma «manual», las cinco raíces de un polinomio de grado cinco. Iremos construyendo poco a poco un script `Ejercicio2.m`.

   Considera el polinomio: $p(z) = z^5 - 5.6z^2 + 10z - 32$.

   a) Calcula el radio ρ de un disco $D(0, \rho)$ que contenga a todas las raíces de p.

   b) Ejecuta en el script una llamada al método de Newton para polinomios con una condición inicial x_0 que proporcione la convergencia a una raíz real de $p(z)$. Utiliza una tolerancia del orden de 10^{-10} y un número máximo de iteraciones que consideres oportuno.

   c) Una vez obtenida una raíz real, haz la deflación del polinomio respecto a dicha raíz. Llamemos $p_1(z)$ al polinomio deflacionado. Si intentas localizar otras raíces reales con otros valores iniciales reales para el método de Newton aplicado a $p_1(z)$ verás que no lo consigues... el polinomio sólo tiene una raíz real. Puedes dibujar la gráfica de este polinomio para intuir que no tiene raíces reales.

   d) Para obtener una idea aproximada de condiciones iniciales que nos permitan obtener alguna raíz compleja incluye en el script el código para dibujar las curvas de nivel de $|p(z)|$; el código siguiente permite obtener la imagen:

```
x=linspace(-3,3,200);
y=linspace(-3,3,200)';%% Cuidado: no olvides el apóstrofo final: es el vector traspuesto
z=abs(polyval(p,x+i*y));%% p es nuestro polinomio
contourf(x,y,z,0:1:15)
```

   e) A la vista del gráfico obtenido, intenta encontrar el resto de raíces. Procede de la forma siguiente: iniciando en el polinomio deflacionado $p_1(z)$ obtenido antes, aproxima una nueva raíz con Newton y luego refínala aplicando el mismo método al polinomio original $p(z)$ con la aproximación obtenida como punto inicial. Después vuelve a deflacionar $p_1(z)$ para obtener $p_2(z)$ y reiterar el proceso.

   f) Una vez obtenidas las cinco raíces vamos a reconstruir el polinomio inicial. Para ello utilizamos la función de Octave:

```
poly(x)
```

   que a partir del vector `x` devuelve los coeficientes del vector cuyos ceros son las componentes de `x`. Escribe el polinomio obtenido a partir de las aproximaciones de las raíces obtenidas y compara los coeficientes de éste y del original: es una forma de estimar la bondad de las aproximaciones.

3. *Búsqueda de raíces de forma aleatoria*

   Utiliza la función `newtonPolyAleatorio.m` para buscar las raíces de los polinomios siguientes:

$$p_1(x) = x^5 - 5.6x^2 + 10x - 32 \text{ (el mismo polinomio del ejercicio anterior)}$$
$$p_1(x) = 3x^{11} - 12x^{10} - x^9 + 23x^8 - x^7 + 32x^5 - 123x^3 + 12x^2 - x + 15$$
$$p_2(x) = 5x^{14} - 0.5x^{11} + 2x^{10} + 0.8x^9 + 2x^8 - x^7 + 32x^5 - 13x^3 + 2x^2 - x - 1$$

   Si la función no devuelve todas las raíces cambia los valores del número de intentos aleatorios, del número máximo de iteraciones y/o de la tolerancia.

   Una vez obtenidas todas las raíces reconstruye los polinomios con los ceros aproximados (como en el último apartado del ejercicio anterior) y compáralos con los polinomios originales: para compararlos mejor, calcula el máximo del valor absoluto de las diferencias de los coeficientes. Atención si el polinomio inicial no es mónico...