UNIVERSIDAD DE MURCIA

Departamento de Matemáticas
Cálculo Numérico en una Variable
Curso 2021/22

Prof. S. Sánchez-Pedreño

Práctica 8. Raíces de polnomios

13 de diciembre de 2021

El objetivo general de esta práctica es el de realizar distintas experiencias con la aproximación de raíces de polinomios mediante el método de Newton.

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 `Practica8Anadir_aBiblioteca.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.

- `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` 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`, escribir en la ventana de comandos los valores de las iteraciones intermedias. 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 determinar 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.
- `newtonPolyOneIt.m`, es una función peculiar concebida para el ejercicio 3 de esta práctica. Su función es obtener una única iteración del método de Newton sobre el polinomio `p` a partir del punto `x0`. Estos son sus argumentos: `p` y `x0`, y la función devuelve el punto `x1` obtenido como iteración de `x0`.

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. 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 `Ejercicio1.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 (esto lo puedes obtener «de un vistazo»).

   b) Utilizando que el polinomio es de coeficientes reales, ejecuta en el script una llamada al método de Newton para polinomios, con una condición inicial x_0 que asegure la convergencia a una raíz real de $p(z)$ (recuerda el teorema 4.38 de las notas). 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 deflacionando. 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.

   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,0:0.5:10)
   ```

   ¿Adivinas dónde están las raíces y cuántas raíces reales hay?

   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.

2. *Búsqueda de raíces de forma aleatoria*

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

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

   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...

3. *Siguiendo la pista de las iteraciones en el método de Newton*

   El objetivo de este ejercicio es visualizar la marcha de las sucesivas iteraciones de un punto mediante el método de Newton así como verificar experimentalmente la veracidad del teorema 4.36.

   Consideramos el polinomio $p(z) = z^3 - 1$ cuyas raíces son las tres raíces terceras de la unidad, es decir

   $$z_1 = 1, \quad z_2 = e^{\pi i/3} = \cos\frac{\pi}{3} + i\,\mathrm{sen}\,\frac{\pi}{3}, \quad z_3 = e^{2\pi i/3} = \cos\frac{2\pi}{3} + i\,\mathrm{sen}\,\frac{2\pi}{3}.$$

   a) Dibuja con una marca de asterisco en color magenta las tres raíces. Utiliza la función
   `set(1,"position",[100,300,1120,870])`
   que cambia la posición y tamaño de la figura `1` (si es demasiado grande y se sale parcialmente de la pantalla utiliza valore menores en el último vector). Establece en la figura los límites en abscisas y ordenadas para que muestre el cuadrado $[-4,4]\times[-4,4]$.

   b) Declara un vector como `[0.6+0.78*i,0.6+0.75*i,0.6+0.9*i]`: serán tres condiciones iniciales para el método de Newton.
   Así mismo declara un vector de colores igual a `color=["r","g","b"]`.

   c) Para cada una de las condiciones iniciales anteriores haremos 15 iteraciones del método, utilizando la función `newtonPolyOneIt.m`. En cada iteración vamos a dibujar el nuevo punto obtenido y un segmento que une el punto anterior con el nuevo (todo ello en el color `color(j)`).

   d) Además con cada nuevo punto dibujamos una circunferencia (en color negro, `"k"`) de centro dicho punto y radio $3|x_k - x_{k-1}|$, siendo x_k y x_{k-1} respectivamente el nuevo punto y el anterior (y 3 es el grado del polinomio). Según el teorema 4.36, cada uno de estas circunferencias debe contener en su interior a una de las raíces del polinomio: podemos verificarlo visualmente.

   e) Para que el dibujo no se vaya haciendo incomprensible, antes de dibujar un nuevo punto, volvemos a repintar la circunferencia anterior en color blanco (lo que la hará desaparecer... aunque dejará algunos puntos blancos por medio...)

   f) Si te apetece y tienes tiempo puedes variar los puntos iniciales...

   Observaciones sobre Octave. Los siguientes detalles serán útiles:

   - Octave dibuja bien los números complejos (aunque he detectado algún fallo... o eso me parece). Así pues, si `x` es un vector de números complejos puedes hacer
     `plot(x)`
     o bien
     `plot(real(x),imag(x))`
   - La orden `waitforbuttonpress()` detiene la ejecución hasta que se pulsa el ratón sobre la ventana gráfica: esto nos permitirá dibujar paso a paso.
   - Para especificar un color puedes usar un código del tipo:
     `plot(x0,"o","color",color(j),"markersize",2)`
     donde, además, se está especificando que los puntos se dibujen con una marca de tipo `"o"` y el tamaño de la marca (`"markersize"`) sea igual a `2`.
   - También te puede interesar
     `plot(x,y,"linewidth",1)`
     que indica un grosor de línea igual a `1`.
   - Recuerda que `clear var` elimina la variable `var`... por si te viene bien.