Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

Tarea 1 - Método de Bairstow

La mayoría de problema técnicos y científicos pueden representarse a partir de ecuaciones, de una o más variables, lineales y no lineales. Para hallar su solución, se hace necesario hacer uso de métodos numéricos debido a la complejidad analítica de las funciones. Un caso particular de este problema es hallar solución a ecuaciones polinómicas. El método de Bairstow permite encontrar raíces de polinomios utilizando aproximaciones cuadráticas, i.e., el método busca una función cuadrática cuyas raíces sean equivalentes a las del polinomio que está siendo evaluado. Si se tiene en cuenta que un polinomio de orden n tiene la forma:

$$P(x) = \sum_{i=0}^{n} b_i x^i \qquad Ec. 1$$

Es posible factorizar dicho polinomio por uno de segundo orden, a través de un proceso de división sintética, con lo cual, $P(x)$ puede expresarse de la siguiente forma:

$$P(X) = (x^2 + px + q) \cdot Q(x) + (Rx + S) \qquad Ec. 2$$

Para aclarar la factorización, el proceso de división sintética que se lleva a cabo, consiste en dividir el polinomio de orden n por el polinomio de segundo orden de la forma $x^2 + px + q$, con lo cual aparece un cociente y un residuo, como resultado de la operación. De esta manera puede expresarse $P(x)$ como en la ecuación 2, como el divisor por el cociente más el residuo. Así, Q(x) es un polinomio de orden n-2 y el cociente de la división, y la expresión (Rx + S) es el residuo de la operación. Con la estructura de la ecuación 2, y teniendo en cuenta que R y S son funciones de p y de q, factores del polinomio de segundo orden, se puede plantear el problema de búsqueda de raíces como uno en el que las expresiones p y q, hagan que el residuo sea cero, i.e., $R(p, q) = 0$ y $S(p, q) = 0$, de modo que la raíz del polinomio de segundo orden que multiplica al cociente sea equivalente a la raíz del polinomio de orden n. Luego, al haber encontrado p y q, para que el residuo sea cero se pueden hallar dos de las raíces del polinomio de orden n con la expresión analítica

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = 0 \quad Ec. 3$$

Luego de encontrar las dos primeras raíces se sigue descomponiendo el polinomio, utilizando división sintética, hasta tener un cociente de primer o segundo orden, para el cual las raíces pueden determinarse de forma analítica con facilidad. Para la implementación del algoritmo en Matlab, se puede dividir el problema en dos funciones. Una que determine los factores p y q que hacen que el residuo sea igual a cero y otra función que permita descomponer el polinomio y determinar, a partir de valores conocidos de p y q, las raíces correspondientes al cociente actual. Con eso en mente, el proceso iterativo que debe llevarse a cabo para determinar los factores de la expresión cuadrática se puede establecer como un problema para solucionar un sistema de ecuaciones no lineales con dos incógnitas, cuyos ceros son desconocidos, lo cual puede resolverse a través del método de Newton-Raphson, un método de punto fijo donde

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad Ec.4$$

Se tienen entonces las siguientes expresiones para p y para q, que pueden resolverse con Newton-Rhapson para dos dimensiones, siendo J el Jacobiano del sistema.

$$Ec.5$$

$$p^{(n+1)} = p^{(n)} - \frac{1}{J}\left(R \cdot \frac{\partial S}{\partial q} - S \cdot \frac{\partial R}{\partial q}\right)$$
$$q^{(n+1)} = q^{(n)} - \frac{1}{J}\left(R \cdot \frac{\partial S}{\partial p} - S \cdot \frac{\partial R}{\partial p}\right)$$

$$J = \frac{\partial R}{\partial p}\frac{\partial S}{\partial q} - \frac{\partial S}{\partial p}\frac{\partial R}{\partial q}$$

Para el cálculo de las derivadas parciales es importante tener en cuenta que R y S dependen de los coeficientes b_iy por ende de p y de q como se dijo anteriormente y se describen con las expresiones que se muestran en [1], las cuales pueden resumirse a continuación:

$$Ec.6$$

$$\frac{\partial b_i}{\partial p} = c_i = -b_{i-1} - p\left(\frac{\partial b_{i-1}}{\partial p}\right) - q\left(\frac{\partial b_{i-2}}{\partial p}\right) = -b_{i-1} - pc_{i-1} - qc_{i-2}$$

$$\frac{\partial R}{\partial p} = -b_{n-2} - p\left(\frac{\partial b_{n-2}}{\partial p}\right) - q\left(\frac{\partial b_{n-3}}{\partial p}\right) = -b_{n-2} - pc_{n-2} - qc_{n-3}$$
$$\frac{\partial S}{\partial p} = -q\left(\frac{\partial b_{n-2}}{\partial p}\right) = -qc_{n-2}$$
$$\frac{\partial b_i}{\partial q} = d_i = -p\left(\frac{\partial b_{i-1}}{\partial q}\right) - b_{i-2} - q\left(\frac{\partial b_{i-2}}{\partial q}\right) = -b_{i-2} - pd_{i-2} - qd_{i-2}$$
$$\frac{\partial R}{\partial q} = -p\left(\frac{\partial b_{n-2}}{\partial q}\right) - b_{n-3} - q\left(\frac{\partial b_{n-3}}{\partial q}\right) = -b_{n-3} - pd_{n-2} - qd_{n-3}$$
$$\frac{\partial S}{\partial q} = -b_{n-2} - q\left(\frac{\partial b_{n-2}}{\partial q}\right) = -b_{n-2} - qd_{n-2}$$

Así pues, las derivadas parciales pueden calcularse de forma recursiva hasta determinar p y q. Finalmente, con p y q se determinan analíticamente las raíces con la ecuación 3 o para el caso de la ecuación de orden 1 con la ecuación 7.

$$\frac{y - b}{m} = x \quad Ec.7$$

A continuación se resume el macro algoritmo del método de Bairstow según lo descrito anteriormente:

1. Factorización por polinomio de segundo orden utilizando división sintética.
2. Resolver p y q tal que la expresión $Rx + S = 0$ utilizando el método de Newton.

Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

3. Utilizar p y q para solucionar sistema de ecuaciones de segundo orden de manera analítica, la solución será una raíz del polinomio.
4. Repetir desde el paso 1 para el cociente resultante de la operación, si el cociente es de primer orden, resolver analíticamente y parar. Si el cociente es de segundo orden resolver directamente de forma analítica como en el paso 3 y parar.

Implementando el ejemplo 2.1 del libro texto, en Matlab se verifica el funcionamiento del algoritmo, y se evidencia que un polinomio de orden n tiene n raíces, que en caso de ser complejas están acompañadas por su conjugado, i.e. el conjugado de una raíz compleja también es una raíz del polinomio. Es importante tener en cuenta que este algoritmo está limitado a resolver sistemas con ecuaciones polinómicas, ya que el proceso de división sintética solo converge a una solución factible para operaciones entre polinomios.

bairstow([1 -4 25 30 -185 428 -257 -870], 0.001, 200)

ans = -1.0000 + 0.0000i 2.0000 + 0.0000i 1.0000 + 2.0000i 1.0000 - 2.0000 2.0000 + 5.0000i 2.0000 - 5.0000i -3.0000 + 0.0000i

Anexos – Código en Matlab

Función principal

```
function r = bairstow(polinomio,error,N) % N es el numero de iter máximo
% polinomio tiene los coeficientes b_i del polinomio
n = size(polinomio,2);
q = polinomio;
r = zeros(1,n-1);

for i = 1:floor(n/2)
   % Se revisa si el orden del polinomio es par o impar
   % de esta manera se determina cómo calcular las raices
   % en el úlimo caso, es decir, cuando el cociente, producto
   % de la descompocisión es de orden 2 o 3.
   if (size(q,2) <= 3)
      f = q;
      q = 0;
   else
      % Si el cociente es de orden par se calcula el nuevo cociente
      % y los factores p y q de la expresión de segundo orden, que
      % para efectos de código se almacenan en un vector f, donde,
      % f(1) = 1, f(2) = p, f(3) = q
      [q,f] = bairstow1(q,error,N);
   end
   % Se calculan las raices de forma analítica para el caso de una raices
   % i.e., el caso de cociente de orden impar
   if size(f,2) == 2
      r(2*i-1) = -f(2)/f(1);
   % Se calculan las raices de forma analítica para el caso de dos raices
   % i.e., el caso de cociente de orden par
   else
      disc = f(2)^2 - 4*f(1)*f(3);
```

```
      r(2*i-1) = (-f(2) - sqrt(disc))/(2*f(1));
      r(2*i)   = (-f(2) + sqrt(disc))/(2*f(1));
   end;
end;
r = r'; % Se retorna un vector con las n raices del polinomio del orden n
```

Función auxiliar

```
function [q,f] = bairstow1(p,error,N) % error -> error admisible
% Se calcula de forma iterativa los valores de p y q hasta conseguir
error
% admisible, de modo que (Rx + S) tienda a cero.
n = size(p,2); % orden del polinomio
r = rand; % aproximación inicial al valor de r
s = rand; % aproximación inicial al valor de s
for i = 1:N
   d = [1 -r -s];                % divisor
   % deconv resuelve el proceso de división sintética
   [q1,r1] = deconv(p,d); % q1: cociente, r1: residuo
   % calculo los coeficientes b_i para el sistema de dos ecuaciones
   b0 = r1(n) + r*r1(n-1);
   b1 = r1(n-1);
   p2 = [q1 b1 b0];
   % se cálcula de nuevo el cociente con división sintética a partir de
la
   % nueva solución de p.
   [q2,r2] = deconv(p2,d);
   % cálculo de expresiones con derivadas parciales
   c1 = r2(n-1);
   c2 = q2(n-2);
   c3 = q2(n-3);
   % Definición del sistema de ecuaciones 2x2, para hallar R y S
   % (Rx + S)
   a = [c2 c3; c1 c2];
   b = [-b1; -b0];
   sol = a\b; % solución del sistema
   % Método de Newton-Rhapson
   % Se calculan nuevos valores, sumo valor missmatch
   r = r + sol(1);
   s = s + sol(2);
   if (norm(sol) < error)
      break;
   end;
end;
% A partir de la última solución de residuo (Rx + S)
q = q1; % se retorna el último cociente
f = d; % se retorna el último divisor
```

Referencias

[1] S. Rosloniec, Fundamental Numerical Methods for Electrical Engineering, Warsaw: Springer, 2008.

Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388