El Teorema de Sturm para la búsqueda de ceros de un polinomio

Realizado por Alexander Borbón Alpízar [1]
alexborbon@costarricense.cr

Resumen

En este artículo se muestra cómo se puede utilizar el teorema de Sturm para aislar los ceros de un polinomio, además se justifica porqué es necesario usarlo, es decir, porqué es preferible utilizar este método en vez de otros más sencillos o conocidos, para mostrar cómo se utiliza el teorema se presentan paso a paso varios ejemplos. También se muestra como se pueden encontrar los valores entre los que se encuentran todos lo ceros (es decir, cómo encontrar las cotas para los ceros), se da una explicación sobre cómo se realizaría un programa de computadora con este método y, por último, se muestran tres métodos para aproximar el cero una vez que se tiene aislado.

Introducción

Uno de los objetivos principales de la teoría de ecuaciones es la búsqueda de ceros de un polinomio. Existen varios métodos relacionados con este tema, algunos tratan sobre la búsqueda de las cotas de los ceros (los valores entre los cuales se encuentran todos los ceros), otros sobre la aproximación de los ceros y hay otros, como el teorema de Sturm, que sirven para saber cuántos ceros tiene la función entre dos valores de " x" y así poder aislarlos.

El conocimiento de estos teoremas vuelven más sencilla la búsqueda de los ceros del polinomio y permiten realizar, por ejemplo, algoritmos para hacer programas de computación que realicen todo el trabajo por nosotros.

Para la aproximación de los ceros existen varios métodos, algunos de ellos son la bisección, la bisección acelerada o el método de Newton, el problema que se presenta es que se deben tener valores iniciales entre los cuales se esté seguro que se encuentra un cero, estos valores iniciales son esenciales para tener éxito en el proceso y encontrarlos no es nada sencillo, para lograrlo es que se utiliza el teorema de Sturm.

Por ejemplo, si se tiene la seguridad que un polinomio tiene un cero entre 0 y 1, se puede aplicar alguno de los tres métodos con esos valores y así encontrar ese cero; pero cómo saber que el polinomio tiene un cero entre estos valores? y cómo se puede estar seguro que es el único cero en ese intervalo?, ya que si hubieran más ceros podrían haber problemas y se tendrían que utilizar límites más cercanos, como 0 y 0.5, hasta tener un solo cero aislado.

En realidad hay varios métodos para lograr aislar los ceros, una primer forma sería usar la gráfica del polinomio, pero para ésto se debe tener una buena gráfica que permita observar todos los ceros y ésta puede cambiar para cada polinomio, además que no serviría para realizar un programa computacional, la computadora no puede ver donde están los ceros en una gráfica.

El método basado en un corolario del Teorema de Rolle se vuelve poco práctico cuando el grado del polinomio es mayor que 3 y aunque la regla de Descartes es muy sencilla, es muy común que falle al calcular el número total de ceros reales (tanto el corolario como el teorema aparecen en el anexo).

Teniendo esto en cuenta, el método más efectivo para aislar los ceros es el Teorema de Sturm, veamos este método mas a fondo, lo primero que se debe ver se el sistema de polinomios de Sturm (ésto se explicará en el siguiente apartado), éste sistema lo necesitamos para hacer evaluaciones en él ya que el número de cambios de signo entre las distintas evaluaciones nos dará la cantidad de ceros entre los valores (en las siguientes secciónes se trata este tema y se dan algunos ejemplos).

Para hacer más sencilla la búsqueda de los ceros, también hay una sección que trata sobre el cálculo de las cotas, estos valores permiten saber entre que números se encuentran todas las raíces, esto es muy importante porque se sabría exactamente donde buscar.
Al final se muestran algunas ideas para poder hacer un programa computacional, no se muestra el código paso a paso sino que se explican las funciones necesarias para hacerlo, lo que recibe cada función, lo que devuelve y una explicación general de cómo se hacen.

Sistema de polinomios de Sturm

Supongamos que se van a buscar los ceros de un polinomio $f(x)$, lo primero que se tiene que buscar al utilizar el Teorema de Sturm es un sistema de polinomios especial para poder evaluar en él, veamos cómo calcular este sistema.
El primer polinomio es el propio $f(x)$; el segundo polinomio es $f'(x)$, es decir, la derivada del polinomio $f(x)$, a este polinomio lo vamos a denotar $f_1(x)$; para el tercer polinomio se debe realizar la división de $f(x)$ entre $f_1(x)$, el residuo que se obtiene con signo contrario será $f_2(x)$; este proceso se continúa, es decir, $f_3(x)$ será el residuo con signo contrario de la división de $f_1(x)$ entre $f_2(x)$; el proceso acaba cuando se obtiene una constante.
Si los cálculos se van hacer a mano, existe una forma de hacer que no aparezcan fracciones en la división, haciéndola más sencilla, si el dividendo es $P(x) = a_n x^n + ... + a_0$ y el divisor es $Q(x) = b_m x^m + ... + b_0$, entonces multiplique $P(x)$ por b_m^{n-m+1}. Si la división la hará la computadora, no hace falta multiplicar el dividendo ya que esto no afecta el resultado del Teorema de Sturm y a la computadora le da lo mismo hacer los cálculo con fracciones o sin ellas.
Veamos un ejemplo,

EJEMPLO 1 (Sistema de polinomios de Sturm)
Calcule el sistema de polinomios de Sturm de $f(x) = x^3 - 9x^2 + 24x - 36$.

Solución

$$f(x) = x^3 - 9x^2 + 24x - 36$$

$$f_1(x) = f'(x) = 3x^2 - 18x + 24$$

Al realizar la división de $3^2 f(x)$ entre $f_1(x)$ se obtiene [2]

$$
\begin{array}{rl}
9x^3 - 81x^2 + 216x - 324 & \underline{\mid 3x^2 - 18x + 24} \\
\underline{-9x^3 + 54x^2 - 72x} & 3x - 9 \\
-27x^2 + 144x - 324 & \\
\underline{27x^2 - 162x + 216} & \\
-18x - 108 &
\end{array}
$$

Así $f_2(x) = 18x + 108$

Ahora se realiza la división de $18^2 f_1(x)$ entre $f_2(x)$ obteniéndose [3]

$$
\begin{array}{rl}
972x^2 - 5832x + 7776 & \underline{\mid 18x + 108} \\
\underline{-972x^2 - 5832x} & 54x - 648 \\
-11664x + 7776 & \\
\underline{11664x + 69984} & \\
77760 &
\end{array}
$$

Por lo tanto $f_3(x) = -\ 77760$

Así, el sistema que se obtiene para $f(x) = x^3 - 9x^2 + 24x - 36$ es

$$f(x) = x^3 - 9x^2 + 24x - 36$$

$$f_1(x) = 3x^2 - 18x + 24$$

$$f_2(x) = 18x + 108$$

$$f_3(x) = -\ 77760$$

Veamos otro ejemplo para dejar más claro el proceso,

EJEMPLO 2 (Sistema de polinomios de Sturm)
Busque el sistema de Sturm para $f(x) = x^3 + 4x^2 - 7$

Solución

$$f(x) = x^3 + 4x^2 - 7$$

$$f_1(x) = 3x^2 + 8x$$

Al realizar la división de $3^2 f(x)$ entre $f_1(x)$ se obtiene

$$
\begin{array}{r|l}
9x^3 + 36x^2 + 0x - 63 & 3x^2 + 8x \\
\cline{2-2}
-9x^3 - 24x^2 & 3x + 4 \\
\cline{1-1}
12x^2 + 0x - 63 & \\
-12x^2 - 32x & \\
\cline{1-1}
-32x - 63 &
\end{array}
$$

Así $f_2(x) = 32x + 63$

Ahora se realiza la división de $32^2 f_1(x)$ entre $f_2(x)$

$$
\begin{array}{r|l}
3072x^2 + 8192x + 0 & 32x + 63 \\
\cline{2-2}
-3072x^2 - 6048x & 96x + 67 \\
\cline{1-1}
2144x + 0 & \\
-2144x - 4221 & \\
\cline{1-1}
-4221 &
\end{array}
$$

Por lo tanto $f_3(x) = 4221$

El sistema que se obtiene para $f(x) = x^3 + 4x^2 - 7$ es

$$f(x) = x^3 + 4x^2 - 7$$

$$f_1(x) = 3x^2 + 8x$$

$$f_2(x) = 32x + 63$$

$$f_3(x) = 4221$$

El Teorema de Sturm

TEOREMA 1 (Teorema de Sturm)
Sea $f(x)$ un polinomio de coeficientes reales tal que $f(x) = 0$ no tiene raíces múltiples. Construya el sistema de Sturm para $f(x)$. Sean a y b números reales, $a < b$ y ninguno de los dos es raíz de $f(x) = 0$. Entonces la cantidad de raíces reales de $f(x) = 0$ entre a y b es la diferencia entre el número de variaciones de signo del sistema de Sturm

$$f(x), f_1(x), f_2(x), \ldots, f_{k-1}(x), f_k(x)$$

para $x = b$ y el número de variaciones del sistema para $x = a$. Los términos que den cero deben ser descartados antes de contar los cambios de signo.

Observe que el teorema pide que la ecuación no tenga raíces múltiples, en este escrito no tomaremos muy en cuenta esta limitación, primero porque la probabilidad que una persona evalúe un polinomio así es bajo, además que el teorema funciona bien para muchos polinomios con raíces múltiples, es decir, son muy pocos los polinomios de raíces múltiples que fallan. Ahora ilustremos el teorema con un ejemplo.

EJEMPLO 3 (Teorema de Sturm)
Aisle los ceros de $f(x) = x^3 - 9x^2 + 24x - 36$.

Solución

Ya conocemos el sistema de Sturm para este polinomio (se calculó en la sección anterior)

$f(x) = x^3 - 9x^2 + 24x - 36$

$f_1(x) = 3x^2 - 18x + 24$

$f_2(x) = 18x + 108$

$f_3(x) = -77760$

Lo primero para aislar los ceros es revisar cuántos hay en total, así que se evalúa el sistema en $-\infty$ y $+\infty$, también evaluamos en $x = 0$ para verificar cuántos ceros positivos y cuántos negativos hay, veamos

Signos	$-\infty$	0	$+\infty$
$f(x)$	-	-	+
$f_1(x)$	+	+	+
$f_2(x)$	-	+	+
$f_3(x)$	-	-	-
Cambios	2	2	1

Como en $-\infty$ hay 2 cambios de signo y en $+\infty$ solo hay uno entonces de $-\infty$ a $+\infty$ hay 2 - 1 = 1 ceros, es decir, el polinomio solo tiene un cero en $I \quad R$.

Ahora, de $-\infty$ a 0 no hay ceros pues ambos tienen igual número de cambios de signo, así que el cero se encuentra de 0 a $+\infty$ pues al realizar la resta de cambios de signo se obtiene 2 - 1 = 1. Ahora podemos ir evaluando de uno en uno empezando desde 0 hasta encontrar en qué intervalo se encuentra.

Signos	1	2	3	4	5	6
$f(x)$	-	-	-	-	-	0
$f_1(x)$	+	0	-	0	+	+
$f_2(x)$	+	+	+	+	+	+
$f_3(x)$	-	-	-	-	-	-
Cambios	2	2	2	2	2	1

De aquí se obtiene que el cero está entre 5 y 6. De hecho, en este caso se encontró directamente que $f(6) = 0$, es decir, el único cero que tiene el polinomio es $x = 6$. Si el cero no se hubiera encontrado directamente, se hubiera tenido que utilizar algún método de aproximación en el intervalo]5, 6[.

Ahora, si el cero se encuentra en $x = 100$ tenemos que evaluar 100 veces para encontrarlo?, no hay alguna forma para tener un límite superior, algún valor del que estemos seguros que el cero se encuentra antes?

Cota superior para los ceros de un polinomio

Hay varias formas para encontrar la cota superior de un polinomio, aquí presentaremos dos, no se puede decir que una sea mejor que la otra, simplemente un método da la mejor cota para algunas funciones y el otro método es el mejor para otras.

Se debe notar que estos teoremas son para encontrar la cota superior, sin embargo, para encontrar la cota inferior simplemente debemos aplicar estos teoremas al polinomio $f(-x)$ (esta evaluación invierte el polinomio, convirtiendo los ceros positivos en negativos y viceversa), veamos estos dos teoremas

TEOREMA 2 (Cota superior de los ceros de un polinomio) Si en una ecuación polinomial de coeficientes reales

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 = 0 \qquad (a_n > 0)$$

el primer coeficiente negativo es precedido por k coeficientes positivos o cero, y si G denota el valor más grande de los valores absolutos de los coeficientes negativos, entonces cada cero real es menor de $1+\sqrt[k]{\frac{G}{a_n}}$

Una condición del teorema es que $a_n > 0$, es decir, que el coeficiente del primer término debe ser positivo, pero si esto no sucede, tan solo se debe multiplicar el polinomio por -1 ya que $-f(x) = 0$ tiene los mismos ceros que $f(x) = 0$.

EJEMPLO 4 (Búsqueda de cotas)
Busque las cotas de los ceros de la ecuación $x^3 - 9x^2 + 24x - 36 = 0$

Solución
Este es el ejemplo que utilizamos para aislar los ceros con el teorema de Sturm. El primer coeficiente negativo es -9, éste está precedido por un solo término, es decir que $k = 1$, además G es el mayor de los coeficientes de los números negativos, en este caso $G = 36$, así que la cota es $1+\sqrt[1]{\frac{36}{1}}=1+36=37$.

Para buscar la cota de los ceros negativos, calculamos $f(-x) = -x^3 - 9x^2 - 24x - 36$, como el primer coeficiente es negativo, entonces $-f(-x) = x^3 + 9x^2 + 24x + 36$, este polinomio tiene todos sus coeficientes positivos, cuando esto sucede, el polinomio no tiene ceros positivos, es decir que la cota es 0.
Por los dos hechos anteriores se puede asegurar que los ceros del polinomio $f(x) = x^3 - 9x^2 + 24x - 36$ están en el intervalo]0, 37[.
Para este ejemplo es mejor usar el segundo teorema ya que nos da una cota menor, veamos

TEOREMA 3 (Búsqueda de cotas)
Si en una ecuación polinomial con coeficientes reales

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0 = 0 \qquad (a_n > 0)$$

el valor numérico de cada coeficiente negativo se divide por la suma de todos los coeficientes positivos que lo preceden, la cota superior será el mayor de estos valores aumentado en 1.

EJEMPLO 5 (Búsqueda de cotas)
Busque las cotas de los ceros de la ecuación $x^3 - 9x^2 + 24x - 36 = 0$

Solución
Veamos el mismo ejemplo que utilizamos con el teorema anterior, calculemos las posibles valores a tomar, $c_1=\frac{9}{1}=9$, $c_2=\frac{36}{1+24}=1.44$. Como se debe tomar el mayor aumentado en uno, la cota es 9 + 1 = 10.

Para la negativa tomamos $-f(-x) = x^3 + 9x^2 + 24x + 36$, que como ya vimos no tiene soluciones positivas. Así, los ceros están en el intervalo]0, 10[, veamos que esta cota es mucho mejor que con el teorema anterior.

Observe que una vez que se tiene el intervalo en donde se encuentra el cero, se puede ir partiendo ese intervalo a la mitad y así ir aislando cada solución. Por ejemplo, se puede verificar con el teorema de Sturm en $x = 0, 5, 10$, así se eliminan 5 valores de una vez y nos quedaría el intervalo]5, 10[, luego se repite el proceso varias veces hasta tener bien aislado el cero (se puede ver que este proceso se asemeja mucho al proceso de bisección).
Veamos ahora un ejemplo completo que haga uso de todos los teoremas.

EJEMPLO 6 (Búsqueda de ceros)
Aisle todos los ceros de la ecuación $x^3 + 4x^2 - 7 = 0$

Solución

Busquemos primero las cotas de la función, comencemos con el primer teorema.

La cota superior es $1+\sqrt[3]{\frac{7}{1}} \approx 3$

Para la cota inferior usamos el polinomio $-f(-x) = x^3 - 4x^2 + 7$. La cota es $1+\sqrt[1]{\frac{4}{1}} = 5$.

Por lo tanto los ceros se encuentran en el intervalo]-5, 3[.

Con el segundo teorema se obtiene

Para la cota superior, como solo hay un coeficiente negativo, la cota es $1+\frac{7}{1+4}=2.4$. Tomemos 3 como cota superior.

Para la inferior, se utiliza el polinomio $-f(-x) = x^3 - 4x^2 + 7$ y se obtiene como cota $1+\frac{4}{1}=5$. Por lo que la cota inferior es -5.

Así, los ceros están en el intervalo]-5, 3[, este intervalo es igual al que se obtuvo anteriormente, por lo que es el que se utilizará al aplicar el teorema de Sturm.

El sistema de Sturm ya lo habíamos calculado anteriormente y se obtuvo que
$f(x) = x^3 + 4x^2 - 7$
$f_1(x) = 3x^2 + 8x$
$f_2(x) = 32x + 63$
$f_3(x) = 4221$

Primero calculamos el sistema en $-\infty$, 0 y $+\infty$ para ver cuántos ceros hay en total y cuántos son positivos y cuántos negativos.

Signos	$-\infty$	0	$+\infty$
$f(x)$	-	-	+
$f_1(x)$	+	0	+
$f_2(x)$	-	+	+
$f_3(x)$	+	+	+
Cambios	3	1	0

Por lo tanto, la ecuación tiene en total 3 ceros, hay 2 ceros negativos (en el intervalo]-5,0[pues -5 es la cota inferior) y 1 positivo (en]0, 3[pues 3 es la cota superior), empecemos a aislar cada uno de ellos, evaluemos en otros valores centrales para ir reduciendo el intervalo, empecemos con la positiva

Signos	$f(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	Cambios
$x = 1$	-	+	+	+	1

Por lo que el primer cero está entre 1 y 3.

Ahora aislemos las negativas

Signos	$f(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	Cambios
$x = -2$	+	-	-	+	2

Así que hay un cero en]-5, -2[y otro en]-2, 0[, ahora

Signos	$f(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	Cambios

$x = -1$	-	-	+	+	1

Por lo que una solución está en]-2, -1[.
Para la última solución se tiene

Signos	$f(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	Cambios
$x = -3$	+	+	-	+	2

Así que el último cero se encuentra en]-5, -3[.

Por último, si se quiere verificar estos resultados (tal vez los resultados no están muy claros), veamos una tabla completa

Signos	$-\infty$	-4	-3	-2	-1	0	1	2	$+\infty$
$f(x)$	-	-	+	+	-	-	-	+	+
$f_1(x)$	+	+	+	-	-	0	+	+	+
$f_2(x)$	-	-	-	-	+	+	+	+	+
$f_3(x)$	+	+	+	+	+	+	+	+	+
Cambios	3	3	2	2	1	1	1	0	0

Ahora ya se ve claramente que una solución está en]-4, -3[, otra en]-2, -1[y la última en]1,2[. Para terminar de encontrar los ceros sólo falta aproximarlos, en el apéndice se pueden ver algunos métodos para hacerlo.
Por último, podríamos hacer algunas reflexiones sobre cómo realizar un programa de computación con estas ideas.

Reflexiones para un programa computacional

Programa Policeros

Si lo que se quiere es realizar un programa computacional que utilice el Teorema de Sturm [4] para aproximar los ceros de un polinomio se deben seguir una serie de pasos que trataremos de enunciar a continuación, nosotros realizamos el programa en Java, sin embargo se puede programar en cualquier otro lenguaje, aquí la idea no es realizar un programa completo donde se muestre paso a paso como realizarlo, sino mostrar las ideas principales sobre cómo se debe realizar, es decir, que no se va a mostrar el código del programa completo, sino solo algunas de las funciones que son necesarias, el programa lo puede bajar al final de este artículo para analizarlo y tenerlo completo.

Un primer problema que se nos puede presentar es cómo leer el polinomio, leer un polinomio como una tira de caracteres y transformarlo a algo que la computadora entienda no es algo trivial, a esto se le llama parsear una expresión y ésto será algo de lo que escribiré en otra ocasión, ahora se debe buscar una forma sencilla, se puede poner una caja de texto para cada coeficiente que se quiera leer y hacer el polinomio de un tamaño fijo (otra forma es preguntar el tamaño primero y después construir las cajas de texto que se requieran).

La mejor forma de guardar el polinomio es en un arreglo de números, en la posición cero se guarda el coeficiente que está solo (es decir que va multiplicado por x^0), en la posición 1 se guarda el coeficiente que acompaña a x^1 y así consecutivamente hasta guardar todo el polinomio.

Una vez que se ha guardado el polinomio deseado, se necesitan varias funciones para trabajar con él. La primera función será la que evalúe un valor de x en el polinomio, esta función debe recibir el arreglo del polinomio y devolver la evaluación, para esta simplemente hay que sumar los término multiplicando cada uno por x^n, donde n es la posición en el arreglo (hay otras formas mejores para hacerlo, pero la idea aquí no es mostrar la mejor forma, sino la más simple). Veamos el código:

```
/*La funcion que sigue es la que evalua el polinomio*/
public double f(Vector fun, double x) {
```

```
    double total = (double)((Double)fun.elementAt(0)).doubleValue();
    for(int i = 1; i < fun.size(); i++){
      total += ((double)((Double)fun.elementAt(i)).doubleValue())*Math.pow(x, i*1.0);
    }
    return total;
  }//Fin de la funcion f
```

Una segunda función calcula la derivada de un polinomio, esta también recibe un arreglo con el polinomio y devuelve otro arreglo con su derivada, es muy sencillo pues cada valor se debe multiplicar por su posición en el arreglo y se elimina el primer término (el que está multiplicado por x^0).

```
  //Funcion que deriva un polinomio
  public Vector derivePoli(Vector fun) {
    Vector derivado = new Vector();
    for(int i = 1; i < fun.size(); i++)
      derivado.addElement(new Double((double)((Double)fun.elementAt(i)).doubleValue()*i));
    derivado.trimToSize();
    return derivado;
  }//derivePoli
```

La tercera función es la que calcula la división de dos polinomios, esta no es tan sencilla, se deben recibir dos arreglos que contengan el dividendo y el divisor (son polinomios) y se puede hacer que solo devuelva el cociente, el residuo o ambos, en este caso sólo se necesita el residuo, así que se declara un arreglo para guardarlo, se realiza la división entre los últimos coeficientes del cociente y el divisor (recuerde que los guardamos empezando con el que acompaña a x^0, luego x^1,etc), este valor se multiplica por cada elemento del divisor y se le suma a su correspondiente elemento del dividendo, estos valores se van guardando en el arreglo declarado.

```
  //Funcion que divide dos polinomios y devuelve el residuo
  //Si no se puede dividir devuelve el vector vacío
  public Vector dividaPolis(Vector dividendo, Vector divisor) {
    Vector dividendoTemp = new Vector();
    Vector cocienteTemp = new Vector();
    Vector residuoTemp = new Vector();

    dividendoTemp = (Vector)dividendo.clone();

    //el grado del dividendo debe ser mayor o igual que el grado del divisor
    if(dividendo.size() >= divisor.size()){
      //Cantidad de veces que se debe hacer el algoritmo
      for(int i = 0; i <= (dividendo.size()-divisor.size()); i++){

        if(residuoTemp.size() > 0)
        residuoTemp.removeAllElements();

        cocienteTemp.addElement(new Double((double)((Double)dividendoTemp.elementAt(
        dividendoTemp.size()-1)).doubleValue()/(double)((Double)divisor.elementAt(
        divisor.size()-1)).doubleValue()));
        //Hace la división de los mas grandes para ir llenando el cociente

        for(int j = 2; j <= dividendoTemp.size(); j++){
          if (j <= divisor.size())
            residuoTemp.addElement(new Double((double)((Double)dividendoTemp.elementAt(
            dividendoTemp.size()-j)).doubleValue()-((double)((Double)divisor.elementAt(
            divisor.size()-j)).doubleValue()*(double)((Double)cocienteTemp.elementAt(i)
            ).doubleValue())));
          else
            residuoTemp.addElement(new Double((double)((Double)dividendoTemp.elementAt(
            dividendoTemp.size()-j)).doubleValue()));
        }//for

        residuoTemp = volverVector(residuoTemp);
        dividendoTemp = (Vector)residuoTemp.clone();
      }//for

      cocienteTemp = volverVector(cocienteTemp);
      residuoTemp = quitaCeros(residuoTemp);
```

```
    if(residuoTemp.size() == 0)
      residuoTemp.addElement(new Double(0.0));
  }//if

  residuoTemp.trimToSize();
  cocienteTemp.trimToSize();

  return residuoTemp; //Aquí se puede retornar el cociente o el residuo cambiando la palabra
}//dividaPolis
```

Se podría hacer otra función para simplificar algunos cálculos, esta recibe un polinomio y le quita los ceros que se encuentran al final, es decir, que el coeficiente que acompaña a la variable de mayor grado sea diferente de cero.

```
//Procedimiento que elimina el número de grado mayor si es 0
public Vector quitaCeros(Vector fun){
  Vector sinCeros = new Vector();
  boolean salir = false;  //para ver si encontró un valor de cierto grado o si no poner 0
  int i = fun.size()-1;

  sinCeros = (Vector)fun.clone();

  while(i >= 0 && salir == false){
    if((double)((Double)sinCeros.elementAt(i)).doubleValue() == 0.0)
      sinCeros.removeElementAt(i);
    else
      salir = true;
    i--;
  }
sinCeros.trimToSize();
return sinCeros;
}//quitaCeros
```

Otra función que puede ser necesaria es la que calcula $-f(x)$, es decir, una función que le cambie el signo a todos los coeficientes del polinomio. También es necesaria la que calcula $f(-x)$, esta función debe cambiarle el signo a los campos impares (1, 3, 5, 7, ...).

```
//Función que le cambia el signo a todas las entradas del polinomio
public Vector cambieSignoPoli(Vector fun) {
  Vector signoCambiado = new Vector();

  for(int i = 0; i < fun.size(); i++)
    signoCambiado.addElement(new Double(-1*(double)((Double)fun.elementAt(i)).doubleValue()));

  signoCambiado.trimToSize();
  return signoCambiado;
}//cambieSignoPoli

//Funcion que le cambia el signo a las potencias impares del polinomio
public Vector cambiaSignoImpares(Vector fun){
  Vector nuevafun = new Vector();

  for(int i = 0; i < fun.size(); i++){
    if(i%2 == 0)
      nuevafun.addElement((Double)fun.elementAt(i));
    else
      nuevafun.addElement(new Double(-1*((double)((Double)fun.elementAt(i)).doubleValue())));
  }

return nuevafun;
}//cambiaSignoImpares
```

Para el teorema de Sturm puede ser útil realizar una función que devuelva el signo de algún valor que recibe, si es negativo puede devolver -1, si es positivo devuelve 1 y si es cero entonces devuelve ese valor, es decir 0.
Se deben hacer las funciones que reciben un polinomio y devuelven la cota superior, esta tambiés es sencilla pues tan solo es de buscar los valores negativos y seguir las fórmulas.

```
//Busca la cota superior de lo ceros positivos de una funcion
```

```
public double cota(Vector fun){
  double valorCota = 0, mayorCoef;

  mayorCoef = Math.abs((double)(((Double)fun.elementAt(0)).doubleValue()));
  for(int i = 1; i < fun.size()-1; i++)
    mayorCoef = Math.max(mayorCoef, Math.abs((double)(((Double)fun.elementAt(i)).doubleValue())));

  if(Math.abs((double)(((Double)fun.elementAt(fun.size()-1)).doubleValue())) == 0)
    valorCota = 0;
  else
    valorCota = 1+(mayorCoef/Math.abs((double)(((Double)fun.elementAt(fun.size()-1)).doubleValue())));

return valorCota;
}//cota
```

Falta también la función (esta ya usa las anteriores) que calcula el sistema de Sturm para un polinomio dado, esta es bastante simple si ya tenemos la funciones anteriores. Otra función es la que evalúa un valor en el sistema de Sturm y devuelve la cantidad de cambios de signo que hubieron.

```
/*La función que sigue es la que hace el teorema de sturm
primero deriva el polinomio, divide el polinomio entre la derivada
y recoge el residuo, luego divide la derivada entre el residuo
y nuevamente recoge el residuo y sigue dividiendo los dos últimos
residuos para obtener otro residuo hasta que ya no se pueda.
Luego evalúa las funciones en -infinito y +infinito y con la resta
de los cambios de signo se sabe cuantos ceros hay...*/
public Vector sturm(Vector fun) {
  boolean salir = false;
  int cerosPositivos = 0, cerosNegativos = 0;
  Vector funciones = new Vector();
  Vector raices = new Vector();
  Vector cotas = new Vector();

  if(fun.size() > 1){

    fun = quitaFallo(fun);

    if(fun.size() > 1){

      //Se buscan las funciones del sistema de sturm
      funciones.addElement(fun);
      funciones.addElement(derivePoli(fun));
      int i = 0;
      while(!salir && i < (fun.size()-2)){
        funciones.addElement(cambieSignoPoli(dividaPolis((Vector)funciones.elementAt(i),
        (Vector)funciones.elementAt(i+1))));
          if(((Vector)funciones.elementAt(funciones.size()-1)).size() == 1){
            if(((double)(((Double)(((Vector)funciones.elementAt(funciones.size()-1))).
            elementAt(0))).doubleValue()) == 0){
              salir = true;
            }
          }
          i++;
       }
       //Se buscan los intervalos con el sistema de sturm

       cerosPositivos = numeroCerosPositivos(funciones);
       cerosNegativos = numeroCerosNegativos(funciones);

       cotas = busqueCotas((Vector)funciones.elementAt(0));
       raices = unaVectores(busqueRaices(funciones, cerosNegativos, (double)(((Double)cotas.
       elementAt(0)).doubleValue()), (double)(((Double)cotas.elementAt(1)).doubleValue())),
       busqueRaices(funciones, cerosPositivos, (double)(((Double)cotas.elementAt(2)).doubleValue()),
       (double)(((Double)cotas.elementAt(3)).doubleValue())));

       if(falloSturm)
         raices.addElement(new Double(0.0));
     }
     else
```

```
        raices.addElement(new Double(0.0));

    }

  raices.trimToSize();
  return raices;
  }//sturm
```

En este programa se presentan algunas funciones de las que no hemos hablado, por ejemplo, el número de ceros positivos y negativos, estas funciones simplemente se evalúa el sistema de Sturm de cero a infinito para saber los ceros positivos y de menos infinito a cero para los negativos. busqueRaices es la función que se encarga de ir acotando las raices y buscarlas. La función quitaFallo lo que hace es eliminar la multiplicidad del cero

```
  //Función que busca los intervalos donde hay ceros
  public Vector busqueRaices(Vector fun, int numCeros, double minimo, double maximo){

    Vector raiz = new Vector();
    int cerosEncontrados = 0;
    double ci = minimo, cs = maximo, cotaux;

    cotaux = (ci+cs)/2;
    if(numCeros == 1)
      raiz.addElement(new Double(miRedondeo(buscarCero((Vector)fun.elementAt(0), ci, cs))));
    else{
      while(cerosEncontrados < numCeros){
        if(Math.abs(cambiosDeSigno(fun, cotaux)-cambiosDeSigno(fun, ci)) == 0){
          ci = cotaux;
          cotaux = (cs+cotaux)/2;
        }
        else if(Math.abs(cambiosDeSigno(fun, ci)-cambiosDeSigno(fun, cotaux)) > 1)
          cotaux = (ci+cotaux)/2;
        else if(Math.abs(cambiosDeSigno(fun, ci)-cambiosDeSigno(fun, cotaux)) == 1){
          raiz.addElement(new Double(miRedondeo(buscarCero((Vector)fun.elementAt(0), ci, cotaux))));
          ci = cotaux;
          cotaux = (cs+ci)/2;
          cerosEncontrados++;
        }
      }//while
    }//if
  raiz.trimToSize();
  return raiz;
  }//busqueraices

  //Funcion que quita el fallo de Sturm, especificamente, la multiplicidad del cero
  public Vector quitaFallo(Vector fun){
    Vector nuevaFun = new Vector();
    falloSturm = false;
    int i = 0;

    while((double)(((Double)fun.elementAt(i)).doubleValue()) == 0.0){
      i++;
    }

    if(i > 0){
      falloSturm = true;
      while(i < fun.size()){
        nuevaFun.addElement((Double)fun.elementAt(i));
        i++;
      }
    }
  else
    nuevaFun = fun;

  return nuevaFun;
  }//quitaFallo
```

Por último, se debe hacer el programa principal que maneje todas las funciones anteriores, es decir que debe leer el polinomio, acomodarlo, calcular el sistema de sturm e imprimir los ceros.

```
  public boolean action(Event evt, Object arg){
      if (evt.target instanceof Button){
          if (evt.target == panel.b1){
              panel.lista.removeAll();
              leaParametros();
              funcion = leePoli(formulaPoli);
              panel.info.setText("");
              if(funcion.size() > 0){
                  funcion = acomodaPoli(funcion);
                  raicesPol = sturm(funcion);
                  imprimaCeros(raicesPol);
              }
              lienzo.repaint();
          }
      }//if del botón
      return true;
  }//action
```

Hay que hacer la funcion que reciba el polinomio y los valores iniciales y aproxime el cero, es decir, la funcion que realice el método de bisección, bisección acelerada o el de Newton.

```
//Buscar cero es la función que aproxima el cero, utiliza el método de bisección
public double buscarCero(Vector fun, double a, double b){
  double xmax = b, xmin = a;
  double xn = a;
  boolean cero = false;

  if(f(fun, xmin) == 0)
    xn = xmin;
  else if(f(fun, xmax) == 0)
    xn = xmax;
  else{
    int i = 0;
      while(!cero && i < 1000){
        xn = (xmin+xmax)/2;
        if(Math.abs(f(fun, xn)) == 0)
          cero = true;
        else if(f(fun, xn)*f(fun, xmin) < 0)
          xmax = xn;
        else
          xmin = xn;

        i++;
      }//while
    }
  //Si este método no funciona se hace la llamada al método newton
  if(Math.abs(f(fun, xn)) > 0.000001)
    xn = newton(fun, a, b);
return xn;
}//buscarCero

//Buscar cero es la función que aproxima el cero, utiliza el método de Newton
public double newton(Vector fun, double a, double b){
  double xn = (a+b)/2;
  Vector derivada = derivePoli(fun);
  boolean cero = false;

  int i = 0;
  while(!cero && i < 500){
    xn = xn-(f(fun, xn)/f(derivada, xn));
    if(Math.abs(f(fun, xn)) == 0)
      cero = true;
    i++;
  }//while
return xn;
}//buscarCero
```

Ya para finalizar, se deben mostrar en pantalla los ceros obtenidos.

```
//Imprime los ceros en una lista
public void imprimaCeros(Vector raices){
  if(raices.size() == 0)
    panel.lista.addItem("No hay ceros reales.");
  else
    for(int i = 0; i < raices.size(); i++)
      panel.lista.addItem("x = "+(double)((Double)raices.elementAt(i)).doubleValue());
}//imprimaCeros
```

Espero que estas reflexiones hayan servido para que puedan realizar un programa computacional eficaz y eficiente, cualquier duda se pueden comunicar a mi correo y con mucho gusto les puedo contestar y ayudar con algún problema que tengan. Más adelante voy a escribir como hacer un parseador simple para leer un polinomio y otro más complicado sobre como leer cualquier expresión metemática y evaluar un valor en ella. El programa que se puede bajar tiene mucho más que solo la búsqueda de ceros, pues grafica el polinomio y marca los ceros en la gráfica.

Apéndice

Métodos para aproximar los ceros de un polinomio

Aunque existen muchos métodos para aproximar los ceros de un polinomio, aquí solo vamos a mencionar tres muy simples y conocidos: el método de la bisección, el método de bisección acelerada y el método de Newton. Si se quiere investigar más sobre estos métodos o buscar más se puede investigar en algún libro de Métodos Numéricos.

Método de bisección

Para poder utilizar este método se necesitan tener dos valores iniciales a y b tales que $f(a)$ y $f(b)$ tienen signo contrario. Una vez que se cumple esto, entonces tome $c=\frac{a+b}{2}$ y evalúe $f(c)$, si $f(c) = 0$ entonces ya se encontró que el cero es $x = c$, sino si $f(a)$ y $f(c)$ tienen signo contrario, entonces tome $a = a$ y $b = c$ y repita el proceso, sino tome $a = c$ y $b = b$. El proceso se repite hasta que $f(c)$ sea menor que un valor pequeño predeterminado, puede ser $f(c) < 0.00001$.

Método de bisección acelerada

Este método es similar al anterior, se necesitan dos valores iniciales a y b tales que $f(a)$ y $f(b)$ tengan diferente signo. La diferencia está en que ahora se toma $c=a-\frac{f(a)(b-a)}{f(b)-f(a)}$. Si $f(c) = 0$ entonces ya se encontró que el cero es $x = c$, sino si $f(a)$ y $f(c)$ tienen signo contrario, entonces tome $a = a$ y $b = c$ y repita el proceso, sino tome $a = c$ y $b = b$. El proceso se termina cuando $f(c)$ es menor que un valor pequeño predeterminado, talvez $f(c) = 0.00001$.

Método de Newton

Para este método solo se necesita un valor inicial a, el siguiente valor que se toma es $c=a-\frac{f(a)}{f'(a)}$. Si $f(c) = 0$ entonces ya encontramos el cero, es $x = c$, sino se continúa el proceso hasta que $f(c)$ sea menor que un valor pequeño predeterminado, puede ser $f(c) < 0.00001$

El problema que presentan estos métodos es que los valores iniciales deben ser muy buenos, el primer método es muy lento, el segundo es un poco más rápido y el tercero es el más sencillo y puede ser el más rápido, sin embargo, puede pasar que se pierda y nunca encuentre el cero. Si se realizan en un programa de computadora es conveniente que se ponga un número máximo de iteraciones por si nunca se llega a un resultado satisfactorio.

Corolario del Teorema de Rolle y la Regla de Descartes

TEOREMA 4 Corolario del Teorema de Rolle
Entre dos raíces reales consecutivas a y b de $f(x) = 0$, hay un número impar de raíces reales de $f'(x) = 0$, una raíz de multiplicidad m es contada m veces.

TEOREMA 5 La Regla de Descartes
El número de raíces reales positivas de una ecuacón real es igual al número v de sus variaciones de signo o es menor que v por un número entero positivo par. Una raíz de multiplicidad m se cuentan m veces.

Dickson, L. *New First Course in the Theory of Equations*. Jhon Wiley and Sons, New York, 1962.

Alexander Borbón[1]
Instituto Tecnológico de Costa Rica, estudiante del CINVESTAV-IPN, México.

obtiene[2]
La función original se multiplica por 3^2 para poder realizar la división sin fracciones de por medio, como 3 es el primer coeficiente del divisor entonces con la fórmula se obtiene $3^{3-2+1} = 3^2$

obteniéndose[3]
En este caso se multiplica el dividendo por 18^{2-1+1}

Sturm[4]
Quiero agradecerle a Christian Páez, Mauricio Gamboa y Evelyn Agüero, mis compañeros y amigos, ya que entre los cuatro realizamos el programa PoliCeros, los extractos que siguen a continuación fueron tomados de ahí y el programa completo se puede utilizar y bajar al final de este artículo.
