**Polinomios y el método de Newton**

* [Introduccion](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#INTRODUCTION)
* [Tips de MATLAB](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#MATLAB_TIPS)
* [El método de Horner](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#HORNERS_METHOD)
* [El método de Horner con Derivadas](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#HORNERS_METHOD_PLUS_DERIVATIVE)
* [El método de Newton-Horner](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#NEWTON_HORNER_METHOD)
* [Deflacción](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#DEFLATION)
* [Factorización de Horner](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#HORNER_FACTORING)
* [Raíces múltiples de Polinomios](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#MULTIPLE_POLYNOMIAL_ROOTS)
* [Para hacer](http://www.fim.utp.ac.pa/Members/fernando.castillo/metodos-numericos/laboratorio-parte-1/laboratorio-9#ASSIGNMENT)

**Introducción**

*Este en un proyecto complicado. Se realizará en dos seciones de Laboratorio.*

Los polinomio se encuentran en muchas aplicaciones matemáticas, por lo que es importante tener una idea de:

* ¿Como evaluar un polinomio?;
* ¿Cómo evaluar la derivada de un polinomio?;
* ¿Cómo encontrar una raíz real de un polinomio?;
* ¿Cómo encontrar varias ( o tal vez todas) raíces reales de un polinomio?;

**Tips de MATLAB**

MATLAB no permite utilizar como índice al cero en un vector. Así es que, desafortunadamente, si se quiere almacenar los coeficientes de un polinomio en un vector, el índice de un coeficiente no coincide con el exponente **x**. La convención será de la siguiente manera: (se almacenará en un vector **c**,)

**p(x) = c(1) \* x^(n-1) + c(2) \* x^(n-2) + ... + c(n-1) \* x + c(n)**

MATLAB tiene una función, que al darle los coeficientes del polinomio **c**, este lo evaluará en el punto **x.**

**px = polyval ( c, x )**

El valor de **c** será un vector, y **x** puede ser un vector también, en cuyo caso se evaluará para cada punto. También se puede mandar los velores de **c** o **x** como parte del comando:

**polyval ( [ 1, -2, 12 ], [ 0, 1, 2, 3, 4 ] )**

El comando **polyder** calcula los coeficientes de la derivadas del polinomio. Así, para evaluar la derivada en un punto, se requiere de dos pasos:

**cp = polyder ( c )   
pp = polyval ( cp, x )**

*Se deberá escribir nuestra propia versión de estas rutinas en el dia de hoy!*

MATLAB puede utilizar una colección de raíces (ya sean reales o complejas) para construir el polinomio correspondiente. El comando es **poly(v)**. Si el set de coeficiees

**v = [ 0, 1, 2, 3 ];  
c = poly [ v ];**

MATLAB responde con

**c = [ 1, -6, 11, -6, 0 ]**

**Ejercio**: Cree un polinomio con las caíces -2, 1 + 2i, 1 - 2i, and 3. Asumiendo que Ud. lo hace correctamente, los coeficientes del vector serán reales. Evalue su polinomio para las primeras raíces para verificarlo. Debe de dar cero.

MATLAB también puede resolver para las raíces, dados los coeficientes. Utilice estos comandos:

**c = [ 1, -6, 11, -6, 0 ]  
v = roots ( c )**

y vea si obtiene de vuelta los coeficientes!

**Metodo de Horner**

Suponga que alguien le solicita evaluar el polinomio

**p(x) = x^3 - 2 \* x^2 - 5 \* x + 6**

Afortunadamente para Ud., quieren saber el valor para **x = 0**. Por simple inspección se puede obtener la respuesta. Suponga ahora que se quiere saber el valor para **x = 2**? No es tan fácil calcularla. Se necesitaría de lápiz y papel o una calculadora. Pero suponga que puedo reescribir el polinomio como:

**p(x) = ( x^2 - 5 ) \* ( x - 2 ) - 4**

Asi es que, para el valor de **p(2)** se observa el resultado.

La idea importante es observar que como se factorice un polinomio como

**p(x) = q(x) \* ( x - a ) + constant**

para un argumento en especial **x=a**, se conoce que **p(a)=constante**. De esta idea, Horner determinó un método que se puede utilizar para evaluar un polinomio, su derivada, o cualquier valor deseado **a**, al reescribir el polinomio como

**p(x) = q(x) \* ( x - a ) + p(a)**

Esta es la forma escencial de la *división sintética*, en la cual siempre se esta dividiendo entre un factor lineal de la forma **x-a**. Recuerda la division sintética? Este es un método en el cual Ud. puede simplificar un polinomios en fracciones.

*Metodo de Horner*: Para evaluar un polinomio con coeficientes **c** en un punto **x**, realice lo siguiente:

px = c(1)  
  
 for i = 2:n  
  
 px = px \* x + c(i)  
  
 end

No se confunda por la escritura!

**Tip para programación**: Para utilizar este método, se necesita determinar en orden **n** del polinomio. Se puede mandar esto como un argumento extra, pero Matlab es lo suficiente flexible para saber cuantas entradas hay en el vector de coeficientes? Si!. Recuerde que, por defecto, todo en MATLAB es "realmente" un arreglo bidimensional. El vector de coeficientes polinomiales es actualmente un arreglo matricial, con una (1) fila y **n** columnas. Para preguntarle a MATLAB cuantas columnas tiene un set de datos **c** se escribe:

**n = size ( c, 2 )**

(Cuando se necesita el número de *fila* de algo, por supuesto, se deberia utilizar **m = size ( c, 1 )**).

**Ejercicio**: Escriba un archivo M llamado *horner.m* de la forma

: **function px = horner ( c, x )**

el cual evalúe el polinomio en los punto **x**. Cuano termine, pruebe con el polinomio

**p(x) = x^3 - 2 \* x^2 + 3 \* x - 4**

el cual tiene los valores siguientes:

X = [-1 0 1 2 3]

P(X) = [-10  -4  -2  2  14]

**Método de Horner con derivadas**

Si se quiere utilizar el método de Newton, se necesitará además de polinomio, las derivadas.

*Método de Horner con derivadas*:

pp = 0.0  
  
px = c(1)  
  
for i=2:n  
  
pp = pp \* x + px  
  
px = px \* x + c(i)  
  
end

**Ejercicio**: De lo anterior, modifique su archivo M *horner.m* para que tenga la forma:

**function [ px, pp ] = horner ( c, x )**

y evalúe un polinomio y sus derivadas en en punto x. Cuando termine, pruebe con el polinomio siguiente:

**p(x) = x^3 - 2 \* x^2 + 3 \* x - 4**

el cual tiene los valores siguientes:

X = [ -1 0 1 2 3]  
  
P(X) = [ -10 -4 -2 2 14]  
  
P'(X) = [10 3 2 7 18]

**Método de Newton-Horner**

Esta es la forma en que se vería el algoritmo:

function root = polynew ( c,x)  
  
FATOL = ?  
  
STEPMAX = ?  
  
for step = 1 , STEPMAX  
  
px = ?  
  
pp = ?  
  
if ( | px | <= FATOL )  
  
...'bien'  
  
end  
  
if ( pp es cero )  
  
...'mal'  
  
x = x - px / pp  
  
end  
  
'Muchos pasos'

**Problema de Programación**: El archivo M de Horner retorna dos valores simultáneamente, **px** and **pp**. Cuál es la forma correcta para ejecutar ysu función de Horner, y asignar ambos valores de salida?

**Ejercicio**: Escriba un archivo M llamado *polynew.m* que lea los coeficientes de un polinomio y un punto inicial, y encuentre una raíz. Ud. necesitará invocar el archivo de Horner como parte de la solución. Comience en **x = -4** y pruebe encontrar una raíz del polinomio:

**p(x) = x^3 - 2 \* x^2 + 3 \* x - 4**

Hay por lo menos dos maneras de verificar su respuesta!

**Ejercicio**: El método de Newton es flexible lo que no lo es el de biseción. Comience en **x = 2+3i** y utilice su rutina **polynew** para encintrar una raíz del polinomio:

**p(x) = x^2 - 6 \* x + 10**

**Deflación**

Ahora el método de Newton da una manera de encontrar una raíz del polinomio. Pero el polinomio de grado **n** tiene esa cantidad misma de raices reales. Suponga que se encontró una de las raíces del polinomio, ¿hay alguna otra manera de encontrar las otras raíces?

Obiamente, hay algunas maneras sencillas para encontrarlas. Se puede, por ejemplo, utilizar un punto de arranque diferente. Si este nuevo punto hace converger a la misma raíz original, se puede seguir intentando con otro valor hasta encontrar una nueva. Como se puede ver es una forma desorganizada para hacerlo. Es más, puede ser que el polinomis tenga una sola raíz real. No hay manera.

En cambio, se puede tratar de utilizar una metodología conocida como la deflaxción. La dea de la deflación es bien simple: si **p(x)** tiene **n** raíces, y se conoce una de ellas, **r1**, entonces para buscar las otras se puede dividir el polinomio por el factor x**-r1** y considerar utilizar el polinomio **p2(x)** que resulta para obtener otra raíz:

**p2(x) = p(x) / ( x - r1 )**

El polinomio **p2(x)** tiene todas las raíces del polinomio original excepto **r1**. Si se aplica nuevamente el método y se encuentra la raíz **r2** y se divide el polinomio **p2(x)** entre **r2**,se obtendría el polinomio **p3(x)** y asi sucesivamente hasta encontrar todas las raíces.

Si los cálculos son lo suficientemente exactos, se podrán calcular todas las raíces del polinomio original.

**Factorización de Horner**

Al asumir que **r1** es una raíz de **p(x)**, se puede utilizar una variante del método de Horner para determinar el polinomio **p2(x)** de tal manera que

**p(x) = p2(x) \* ( x - r1 ) + p(r1)**

*Método de Horner con factorización del residuo*: Dados los coeficientes c del polinomio  **p(x)**, determine los alores de **b**:

b(1) = c(1);  
for i = 2:n-1  
b(i) = c(i) + r1 \* b(i-1)  
end  
rem = c(n) + r1 \* b(n-1)

El polinomio **p2(x)** esta definido por los coeficientes de **b** y el residuo **rem** es  el valor de **p(r1)**.

**Ejercicio**: De lo anterior, defina un archivo llamado, factorh*orner.m*, con la forma:

**function [ b, rem ] = factorhorner ( c, r1 )**

Antes de continuar, verifique que el proceso funcione.

**Ejercicio**:

* Defina **c4 = poly ( [ 1, 2, 3, 4 ] )**;
* Verifique que **polyval ( c4, [1, 2, 3, 4] )** sea cero.
* Factorice la raíz **x = 1** utilizando su código **factorhorner**, y almacene los coeficientes resulantes como **c3**;
* Revise los lalores de **c3**, con **polyval ( c3, [1, 2, 3, 4] )**;
* Ahora, trabaje con **c3**, para encontrar **c2**, y **c1**, cada vez descartanto una raíz.

**Polinomios con Múltiples raíces:**

Ahora se verá como se puede visualizar un método para obtener algunas o todas las raíces reales del polinomio **p(x)**. Primero, se hace realmente fácil encontrar la primera raíz **r1** por el método de Newton. Para encontrar las otras raíces se aplca el método de factorización de Horner para **p(x)**.

**p(x) = p2(x) \* ( x - r1 ) + p(r1)**

Ya que **p(r1)** es supuestamente 0, o casi cero, se puede asumir la tarea para encontrar el polinomio **p2(x)**. Si se pueden pasar los coeficientes del puevo polinomio al método de Newton, con este se encuentra la siguiente raíz, hasta completar todos los pasos

Esquema de la rutina.

function roots = mulpolynew ( c, x )  
  
 Asigne las raíces a un vector vacío;  
 Haga **n** el tamaño del vector de coeficientes.  
  
 Realice un lazo **n-1** veces.  
  
 Llame **polynew** para encontrar una raíz;  
  
 Si el valor de la funciónIf en la nueva raíz es muy grande, algo anda mal,  
 asi es que imprima una bandera de aviso y regrese.  
  
 Añada la raíz al vector de raíces;  
   
 Llame la función factorhorner para obtener los nuevos coeficientes del   
 polinomio reducido;  
  
 end

**Ejercicio**: Haga una nueva versión de esta rutina llamada, *mulpolynueva.m*, tque trabaje apropiedamente. Pruebe en el polinomio siguiente:

**x^2 - x - 6**

¿El programa funciona correctamente? Ahora utilice el siguiente polinomio:

**x^3 - 3\*x^2 + 4\*x - 12**

 ¿Que resulta?

**Para hacer en el Laboratorio**

Ejecute su programa **mulpolynueva** con:

diary ex5  
c = [ 1, -5, -47, -39, 90, 0 ]  
x = 5  
mulpolyneva ( c, x )  
diary off

y envíeme estos resultado en un archivo llamado *ex5*, con su nombre completo y C.I.P.