

**Trabajo de Algebra 2**

Integrantes: Félix Pérez

Hernán Puelles

Fecha: 17/06/2014

**Introducción**

En este trabajo explicaremos como funciona nuestro programa que realiza la aproximación de mínimos cuadrados para sistemas de ecuaciones y para la aproximación de puntos a funciones lineales, cuadráticas y cúbicas.

Decidimos programar utilizando el lenguaje de programación Python ya que es el lenguaje que habíamos utilizado más recientemente y con el cual manejábamos mejor algunas librerías que serían de utilidad.

A pesar de que conocíamos en python funciones que realizaban directamente las operaciones que requeríamos, decidimos programar de todas formas cada una de las operaciones necesarias de la forma en que las realizaríamos nosotros, a costo de eficiencia en el código, con el objetivo de cumplir con la finalidad del trabajo, el cual era aprender a calcular y entender cómo funcionan los mínimos cuadrados.

Python Version: 3.4.1 x32

Modulos: Numpy 1.8.1 (librería matemática)

Matplotlib: 1.3.1 (librería grafica)

Archivos:

* MinCuad.py:

Incluye menú de usuario interactivo y opciones graficas.

* MinCuad\_func.py:

Incluye sólo las funciones de operaciones algebraicas con comentarios de cada función.

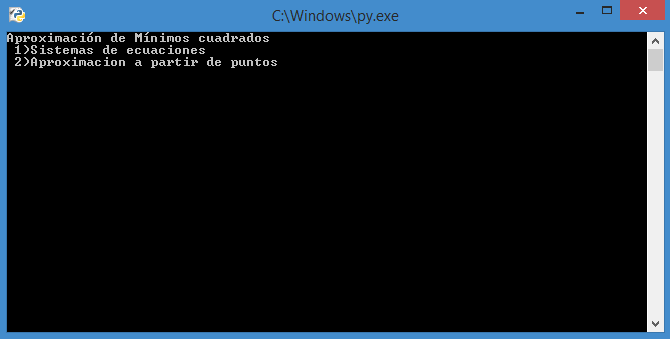
Funcionamiento

El usuario elige si desea calcular sistemas de ecuaciones o aproximar a puntos:

-Para sistemas de ecuaciones deberá ingresar la cantidad de filas y columnas, luego ingresar la matriz de coeficientes seguida de la matriz de soluciones.

Si se pueden calcular los mínimos cuadrados, el programa calculará dicha solución.

-Para puntos, el usuario ingresa los puntos que desea aproximar, dependiendo de la cantidad de coordenadas "x" distintas el usuario tendrá la opción de calcular la aproximación a una recta, a ecuación cuadrática y/o cúbica, además el programa puede calcular cual es la ecuación que mejor se aproxima.



Especificaciones

Para sistemas de ecuaciones:

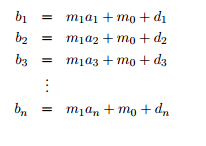
Dada una matriz de coeficientes "A" de orden (nxm) y una matriz de soluciones "B"orden (nx1), el sistema A\*X=B tiene solución única mediante mínimos cuadrados si y sólo si el rango de la matriz A es igual a m, en otras palabras, el número de filas linealmente independientes de A es igual al número de columnas, en definitiva

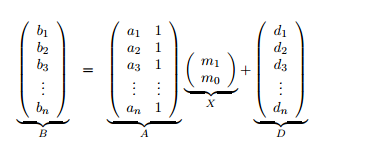
A es una matriz cuadrada y podemos entonces calcular su inversa.

Por lo que la solución única mediante mínimos cuadrados es:



Para puntos:

Sea un conjunto de puntos (ai,bi) con i= [1,2....n] Si existen al menos m+1 puntos con coordenadas x distintas construimos un sistema de la forma:

Que es equivalente a:

Con lo cual resolvemos mediante mínimos cuadrados utilizando la formula previamente descrita:

* Si m=1 entonces obtenemos la ecuación de la recta
* Si m=2 una ecuación cuadrática
* Y si m=3 una ecuación cúbica

Implementación

Como ya hemos explicado el funcionamiento del método de aproximación de mínimos cuadrados, ahora es necesario describir como realizamos cada paso en nuestro código, para ello creamos funciones para poder simular la solución:



A continuación se describirán las funciones utilizadas para sistemas de ecuaciones :

* MULT: Multiplicación de matrices, primero verifica que 2 matrices A(nxm) y B(pxq) sean multiplicables es decir m=p, luego crea una matriz de dimensiones (nxq) y va llenando cada posición [i][j] con el resultado de la sumatoria de los productos entre cada elemento de las filas[i][..] de la matriz Ay las columnas[..][j] de la matriz B.
* TRANSP: #Transpuesta de una matriz, dada una matriz de dimensiones(nxm) creamos una matriz transpuesta tmatriz de dimensiones(mxn) y llenamos cada una de las posiciones de tmatriz[i][j] con el valor de matriz original en la posición matriz[j[i], es decir invertimos las filas y columnas.
* DET: #Determinante de una matriz, dada una matriz de 2x2 [[a b],[c d]], realiza el producto de la diagonal principal menos la diagonal secundaria, es decir: [(a\*d)-(b\*c)]. Para matrices mayores aplicamos teorema de Laplace, utilizando la primera fila calculamos la suma de cada elemento multiplicado por el determinante (de forma recursiva) de su matriz menor complementaria, o submatriz, con el correspondiente signo (-1)^(i+j), en este caso i siempre es 0 ya que usamos la primera fila y como nuestros arreglos están basados en index calculamos de la forma (-1)^(j+2).
* COFACTORES: esta función nos permite obtener la matriz de cofactores(cmatriz), substituyendo en cada termino de matriz[i][j]", por el termino cofactor cmatriz[i][j], que corresponde al determinante de la submatriz con signo correspondiente a la posición en que se encuentre.
* INVERSA: #Para obtener la matriz inversa, si está es de dimension 2x2 se aplica la formula: (1/det)\*[[d -b],[-c a]], si las dimensiones de la matriz son mayores a 2x2, se realiza el producto entre: (1/(det(matriz))\*adjunta(matriz), donde la adjunta corresponde a la transpuesta de la matriz cofactores

Nota: en el ingreso de datos del programa principal comprobamos que la matriz (A.t\*A) que necesita ser invertida tiene efectivamente matriz inversa pero de todos modos hemos incluido aqui los condicionantes necesarios.

* SUBM: Función Submatriz, devuelve la matriz resultante de eliminar una fila "i" y columna "j".
* MINCUAD: #Teniendo todas las funciones necesarias podemos resolver la ecuación de mínimos cuadrados :

X≈ [(A.t \* A)^(-1)] \* A.t \* B

Cabe destacar que está ecuación puede resolverse sólo si existe la inversa de "A.t\*A" En el programa principal determinamos que esta condición se cumple al momento de ingresar los datos de todas formas hemos incluido la condición en está versión para que la función funcione por sí sola. Usamos matrix\_rank del modulo álgebra lineal de numpy para obtener el rango, y shape[1] las columnas.

A continuación se describirá las funciones utilizadas para la aproximación por puntos:

* APROX: #Dado un conjunto de puntos de la forma [(a,b)...(an,bn)] Primero obtenemos las coordenadas (x,y) por separado en x1, y1 respectivamente, formamos la matriz A de la forma [x 1]\*n filas en A1, calculamos los mínimos cuadrados entra la matriz construida A1 y la matriz de coordenadas "y", la matriz solución que obtenemos correspondiente a los valores de la pendiente y coeficiente de posición, estos valores los podemos evaluar en la ecuación de la recta usando las coordenadas x originales, obteniendo las coordenadas "y" correspondientes a la recta aproximada, teniendo esto podemos obtener la suma de los cuadrados de los residuos d1=sum((recta-y1)\*\*2) De forma similar operamos para encontrar la ecuación cuadrática y cúbica que más se aproxima, construyendo matrices de la forma [x^2 x 1]\*n filas y [x^3 x^2 x 1]\*n filas respectivamente.

Nota: para poder encontrar una ecuación de grado m necesitamos m+1 puntos, esta condición está determinada

en el ingreso de datos del programa principal donde la variable “grf” determina las ecuaciones que se pueden y desean calcular.