# Cómputo Científico I

## Ejercicios 5

1. Se tiene un archivo de texto plano en el cual están escritas las fórmulas de dos funciones $f(x)$ y $g(x)$, cada una en una línea del archivo. Escriba una función de Matlab que reciba el nombre del archivo de texto plano, lo abra para lectura, lea las dos funciones y grafique $f(x) + g(x)$, $f(x) - g(x)$ y $f(x).g(x)$.

2. Escriba una función de nombre *polyintdef* que permita calcular la integral definida de una polinomio en un intervalo cerrado dado. La función recibirá como argumentos de entrada, el polinomio en forma de vector de coeficientes, el extremo izquierdo del intervalo y el extremo derecho del intervalo. El argumento de salida es el valor de la integral definida.

3. A fin de facilitar el uso de expresiones analíticas que describen el comportamiento de ciertos materiales de interés al ser calentados, se requiere transformar esas expresiones algo complejas en funciones más manejables en lo que a derivación e integración se refiere, pero siempre manteniendo tanta similitud como sea posible con las funciones originales.

   Las expresiones de las funciones en cuestión están almacenadas en archivos ascii en un formato estándar que consiste en un número que indica la cantidad de funciones que contiene el archivo y en cada una de las líneas que siguen, un número de función y la cadena de caracteres que expresan la función misma, sin espacios intermedios.

   La solución propuesta consiste en crear un grupo de funciones de Matlab capaces de extraer cualquiera de las funciones archivadas, evaluarla en un conjunto de puntos pertenecientes a un intervalo de interés y generar un polinomio de ajuste por mínimos cuadrados que la aproxime con suficiente exactitud.

   Se le pide crear un grupo de programas atendiendo a las especificaciones que se dan a continuación:

   *a*) Una función de Matlab que lea apropiadamente los archivos ascii de funciones y extraiga la que el usuario señale mediante el número que la acompaña. Si el usuario no provee el número de la función, se asumirá que se quiere trabajar con la primera función contenida en el archivo, y si el número dado por el usuario excede la cantidad de funciones en el archivo, se asumirá que se quiere trabajar con la última función del archivo.

   *b*) Una función de Matlab que evalúe la función extraída en un conjunto de $n$ puntos equidistantes de un intervalo $[xi, xf]$ (proveídos por el usuario), y cree las $n$ parejas de coordenadas $(x, y)$ que permitirán realizar el ajuste. Si el usuario no provee el intervalo, se asumirá $[-5, 5]$ y si no provee el número de puntos equidistantes, se asumirá $n = 21$.

   *c*) Una función que genere el polinomio de ajuste basado en mínimos cuadrados, probando polinomios desde el grado cero en adelante, hasta que el error absoluto acumulado en todos los puntos del ajuste no exceda el valor *maxerror* o hasta que se hayan probado todos los grados hasta el valor *maxgrad* (lo que ocurra primero). Ambos valores serán proveídos por el usuario y sus valores por defecto serán 10 y 9 respectivamente.

   *d*) Una función que escriba los resultados del ajuste en un archivo de nombre *resultados_ajuste.txt*. La información mínima que este archivo deberá contener será:

   - Fecha y hora en que se escribe el resultado.
   - Expresión (en formato Matlab) de la función que se trabajó.
   - El grado del polinomio resultante.
   - Los coeficientes del polinomio resultante.
   - La suma de los errores absolutos obtenidos en todos los puntos de ajuste, para este polinomio.

   *e*) Una función de Matlab que genere en una sola figura, tres gráficas que contengan:

   - La función que se trabajó, marcando los puntos utilizados para el ajuste.
   - El polinomio de ajuste, marcando también los puntos utilizados para obtenerlo.
   - Igual que en el ítem anterior, pero marcando los puntos utilizados para el ajuste y la gráfica en escalón del error acumulado en cada punto de ajuste.

*f*) Un *script* que será la interfaz con el usuario y como tal se encargará de invocar automáticamente a las funciones anteriores.

Un ejemplo de archivo de datos para este ejercicio se muestra a continuación

```
6
x.^4+x.^3+2*x.^2+10*x
exp(x)+x.^2+sqrt(x+2)
log(x+1)+log(x.^2)
x+exp(x)
exp(sqrt(x.^2))
exp(sqrt(x+22))
```

4. La *integración numérica* permite aproximar el área comprendida entre la curva descrita por una función continua $f(x) \geq 0$, el eje de las abscisas y sendas rectas verticales que pasan por un par de puntos $a$ y $b$ ($a < b$). Existen varias maneras de plantear la integración numérica. Una de ellas consiste en tres pasos fundamentales:

   *a*) Dividir el intervalo de integración $[a, b]$ en subintervalos determinados por un conjunto de puntos $\{x_0, x_1, x_2, \ldots, x_{N-1}, x_N\}$, no necesariamente equidistantes, tales que $a = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N = b$;

   *b*) Calcular el área de cada rectángulo cuya base es la longitud del subintervalo $[x_i, x_{i+1}]$ y cuya altura es el promedio de los valores $f(x_i)$ y $f(x_{i+1})$; y

   *c*) Sumar las áreas de todos los rectángulos descritos en el ítem anterior.

   En un archivo de texto plano se suministra una tabla de datos $\{(x_i, y_i)\}_{i=1}^{N}$, donde los $y_i$ (ordenadas) son las imágenes de los $x_i$ (abscisas) mediante una función desconocida $f$. Se le pide crear una función en Matlab, llamada *areaint*, en la cual se aproxime el área descrita más ariba. Para ello, deberá tomar en cuenta los siguientes aspectos:

   - Ud. debe escribir la fórmula matemática que representa la aproximación del área de acuerdo a la metodología numérica descrita en los pasos 1, 2 y 3;
   - El intervalo de integración $[a, b]$ está determinado por las abscisas mínima y máxima del conjunto de datos;
   - Tome en cuenta que los datos no vienen ordenados de ninguna manera específica;
   - Ud. debe decidir cuáles son los parámetros de entrada y salida de la función *areaint;*
   - Ud. debe hacer todas las verificaciones pertinentes sobre los argumentos de entrada de su función;
   - Es obligatorio comentar su función, indicando sintaxis de uso, para qué sirve, parámetros de entrada y salida con sus respectivas especificaciones; también se espera que incluya comentarios estratégicos en el código de definición de la función;
   - La función *areaint* **no** se encargará de abrir ni manipular archivos de datos.

   Además, Ud. debe crear un *script* en Matlab, de nombre *principal*, en el cual se pida el nombre del archivo de datos por pantalla, se abra este archivo, se lean los datos contenidos en él, y se invoque convenientemente la función *areaint*. Luego se reportará, también por pantalla, el resultado obtenido. Este reporte debe incluir el nombre del archivo de datos procesado, el intervalo considerado y el área calculada (ésta se debe imprimir con 15 decimales). Su *script* debe efectuar todas las verificaciones que sean pertinentes. Se debe dar al usuario la posibilidad de procesar tantos archivos de datos como desee. Decida Ud. cómo gestionar este comportamiento del *script*. Es obligatorio comentar suficientemente su programa. Tome en cuenta que los archivos de datos que serán suministrados son de texto plano y tienen, todos ellos, la siguiente estructura: a) un texto de encabezado (primera línea del archivo), en el cual se especifica a qué se refieren los datos que aparecen a partir de la segunda línea del archivo, y b) dos columnas de números (i.e. dos números, separados por espacios, en cada línea), a partir de la segunda línea del archivo.

5. En la teoría de cálculo básico de funciones reales de una variable real, se sabe que si una función es continua en un intervalo cerrado $[a, b]$ y los signos de $f(a)$ y $f(b)$ son distintos, entonces existe algún punto $x$ entre $a$ y $b$ tal que $f(x) = 0$. En otras palabras, estas hipótesis (continuidad de la función en un intervalo cerrado y

cambio de signo de la función en los extremos de éste) aseguran la existencia de al menos una solución de la ecuación no lineal general $f(x) = 0$. Se plantea el siguiente algoritmo para aproximar una solución de dicha ecuación:

**Paso 1**: Recibir una función $f$ y un intervalo cerrado $[a, b]$
**Paso 2**: Si los signos de $f(a)$ y $f(b)$ son iguales, declarar el incumplimiento de la hipótesis de cambio de signo y salir de la función
**Paso 3**: Establecer el número máximo de iteraciones, *maxiter*, en 1000
**Paso 4**: Establecer la tolerancia para convergencia, *tol*, en $10^{-12}$
**Paso 5**: Generar un número aleatorio $c$ entre $a$ y $b$
**Paso 6**: Mientras que la longitud del intervalo $[a, b]$ y el valor absoluto de $f(c)$ sean ambos mayores que *tol*, y no se haya iterado más veces que *maxiter*, repetir los pasos del 6.1 al 6.3:
  **Paso 6.1**: Si los signos de $f(a)$ y $f(c)$ son distintos, asignar el valor de $c$ en $b$
  **Paso 6.2**: Si los signos de $f(c)$ y $f(b)$ son distintos, asignar el valor de $c$ en $a$
  **Paso 6.3**: Generar un número aleatorio $c$ entre $a$ y $b$
**Paso 7**: Si se iteró *maxiter* veces, declarar la no convergencia del método;
Si no, retornar el valor de $c$ como la aproximación buscada

Cree una función de Matlab que implemente este algoritmo. Verifique el correcto funcionamiento de su función resolviendo algunas ecuaciones no lineales sencillas, como por ejemplo $x - cos(x) = 0$ en el intervalo $[0, \frac{\pi}{2}]$. Luego, utilícela convenientemente para calcular el primer *punto fijo* positivo de la función $f(x) = tan(x)$. Un valor $x$ es un punto fijo de una función $f$ si y sólo si $f(x) = x$.

6. Se entiende por *fórmula química* a cualquier combinación de los caracteres 'C', 'H', 'O', 'N', '2', '3', '4', '5' y '6', con la única restricción de que el primer caracter de una fórmula es siempre 'C', 'H' 'O' o 'N'. Por ejemplo, la cadena de caracteres 'CH3C2H4' es una fórmula química válida, mientras que las cadenas '2CH3N' y 'NO1C3' no lo son. La *cantidad de átomos* de un elemento químico en una fórmula química se obtiene de la siguiente manera:

   - Cada vez que el símbolo del elemento químico aparece en la fórmula, se cuenta el elemento tantas veces como diga el índice que aparece justo a su derecha en la fórmula. Si a la derecha del símbolo del elemento químico no hay un caracter que represente a un número, sino el símbolo de otro elemento químico, entonces el elemento se cuenta sólo una vez. La suma total de todos estos conteos es la cantiadad de átomos del elemento en la fórmula.

   Así, por ejemplo, en la fórmula química 'CH3C2H4', hay 3 átomos del elemento 'C' y 7 átomos del elemento 'H'. En la fórmula 'CNO2H' hay 1 átomo de cada uno de los elementos 'C', 'N', y 'H', mientras que hay 2 átomos de 'O'.

   Se dispone de un conjunto de *fórmulas químicas* contenidas en un archivo de texto plano de nombre *forquim.txt*, de manera que en cada línea del archivo aparece sólo una fórmula química. Se le pide escribir un programa en Matlab que lea estas fórmulas y las analice para:

   *a*) Decidir si cada fórmula es o no válida,
   *b*) Cuando un fórmula sea válida, decir cuáles elementos están presentes en la misma, indicando la cantidad correspondiente de átomos, y
   *c*) Calcular el peso molecular de la fórmula (los pesos o masas atómicas de los elementos 'C', 'H', 'O' y 'N' son 12.011, 1.0079, 15.9994 y 14.0067 respectivamente).

   Los resultados serán impresos en un archivo de texto plano de nombre *forquim_ analisis.txt*, de modo que en cada línea de éste aparezca el resultado del análisis del elemento químico que está en la línea correspondiente del archivo *forquim.txt*. Por ejemplo, si el contenido del archivo *forquim.txt* es el siguiente:

   CH3C2H4
   CNO2H
   NO1C3
   OOC6NH3

4HHC3

NC2H3O7

entonces el archivo de salida, *forquim_ analisis.txt* es:

CH3C2H4 composición: 'C' 3 átomos, 'H' 7 átomos ; peso molecular: 43.0883

CNO2H composición: 'C' 1 átomo, 'N' 1 átomo, 'O' 2 átomos, 'H' 1 átomo ; peso molecular: 59.0244

NO1C3 fórmula no válida

OOC6NH3 composición: 'O' 2 átomos, 'C' 6 átomos, 'N' 1 átomo, 'H' 3 átomos ; peso molecular: 121.0952

4HHC3 fórmula no válida

NC2H3O7 fórmula no válida

7. Las notas de las tareas entregadas por un grupo de N estudiantes de Computo Científico han sido reportadas en un archivo de texto llamado *notas.txt*. Las primeras evaluaciones (primer corte) fueron escritas por filas pero las segundas (segundo corte) se transcribieron como columnas o tablas (Fig. 1), manteniendo el mismo orden de los estudiantes. El Profesor de Cómputo requiere un archivo consolidado en el cual todas las notas de los estudiantes, tanto las del primer corte como las del segundo, se muestren por columnas, completando con *NaN* aquellas evaluaciones faltantes, y que además contenga el promedio de las notas por cada alumno (Fig. 2). Las tareas faltantes de cada estudiante siempre son las últimas.

   Escriba un *script* en Matlab que permita crear un archivo, denominado *notas_ modificadas.txt*, con las características descritas en el párrafo anterior. Ejecute su *script* para el archivo dado en la figura 1 y para otro que le será suministrado, de nombre *notas2.txt*.

```
%lista de notas, tareas CO2111, grupo 1
%número de estudiantes en el archivo
5
%cantidad total de tareas en el trimestre
14
%cantidad de tareas entregadas por cada estudiante en el primer corte
5 7 2 4 7
%notas de Pedro Pérez
2.5 1.5 3.0 4.5 4.0
%notas de Carlos Gruber
4.5 4.0 4.0 4.5 5.0 4.0 5.0
%notas de María Juárez
1.5 2.0
%notas de Alfredo López
0.5 1.0 4.0 3.5
%notas de Jazmín Herrera
4.5 4.0 4.0 4.5 5.0 4.0 5.0
%notas de las tareas entregadas por cada estudiante en el segundo corte
P.Perez C.Gruber M.Juarez A.Lopez J.Herrera
  4.0  |   3.5  |   3.0  |  3.0  |   4.0
  1.0  |   2.0  |   2.0  |  2.5  |   3.0
  4.5  |   1.0  |        |  3.5  |   4.5
  3.5  |        |        |  3.0  |   5.0
       |        |        |  3.5  |   4.5
       |        |        |  3.0  |   5.0
       |        |        |       |   4.5
```

Figura 1: Archivo *notas.txt*

```
%lista de notas, tareas CO2111, grupo 1
 P.Pérez  C.Gruber  M.Juárez  A.López  J.Herrera
2.500000  4.500000  1.500000  0.500000  4.500000
1.500000  4.000000  2.000000  1.000000  4.000000
3.000000  4.000000  3.000000  4.000000  4.000000
4.500000  4.500000  2.000000  3.500000  4.500000
4.000000  5.000000     NaN    3.000000  5.000000
4.000000  4.000000     NaN    2.500000  4.000000
1.000000  5.000000     NaN    3.500000  5.000000
4.500000  3.500000     NaN    3.000000  4.000000
3.500000  2.000000     NaN    3.500000  3.000000
   NaN    1.000000     NaN    3.000000  4.500000
   NaN      NaN        NaN      NaN     5.000000
   NaN      NaN        NaN      NaN     4.500000
   NaN      NaN        NaN      NaN     5.000000
   NaN      NaN        NaN      NaN     4.500000

         ***** P R O M E D I O S *****

3.166667  3.750000  2.125000  2.750000  4.392857
```

Figura 2: Archivo resultante de procesar los datos en *notas.txt*.

8. Cree una función en Matlab que reciba como entrada una cadena de caracteres y que devuelva otra cadena de caracteres que resulte de: 1) eliminar todos los espacios en blanco que estén al principio y al final de la cadena original, y 2) sustituir cualquier grupo de dos o más espacios en blanco consecutivos que aparezcan en la cadena original, por un solo espacio en blanco.

9. Un programa de DACE ha generado un archivo de texto plano en el cual se lee: *"En la sección 4 del curso de Cómputo Científico I del trimestre septiembre-diciembre 2012, se inscribieron 11 estudiantes cuyas notas en el primer examen parcial fueron las siguientes: 7.3, 8.4, 7.3, 5.1, 2.9, 1.8, 3.0, 4.0, 7.6, 3.9 y 5.6. No se había retirado ningún estudiante al momento de imprimir este reporte."*. En realidad, Ud. no tiene ninguna información previa sobre la forma en la que está estructurado el mensaje que está escrito en el archivo. Sólo sabe que: (i) el mensaje está escrito en la única línea existente en el archivo, es decir, no hay saltos de línea en el archivo, y (ii) el mensaje contiene las notas de los estudiantes de cierto curso dictado en la Universidad, y las mismas están escritas como números de punto flotante dentro del mensaje, siendo los *únicos* números de punto flotante contenidos en el mensaje. Se le pide escribir una función de Matlab que cumpla las siguientes especificaciones:

   *a*) Recibe un argumento de entrada que corresponde al nombre del archivo generado por DACE.

   *b*) Extrae las notas de todos los estudiantes del curso.

   *c*) Calcula y retorna como argumentos de salida, la cantidad de estudiantes del curso, un vector de notas y el promedio de éstas.

   *d*) ¿Cree Ud. que es posible extraer alguna otra información del mensaje? Explique.

10. Se le pide crear las siguientes funciones:

   *a*) Una función que lea el archivo de texto que contiene la sopa de letras, extraiga la matriz de letras y la almacene en una variable de nombre **sopa**.

      1) La entrada de esta función será el nombre del archivo de texto.
      2) La salida será la matriz **sopa**.

   *b*) Una función que realice la búsqueda de la palabra deseada en las filas de la matriz **sopa**. La búsqueda se llevará a cabo en ambos sentidos (de izquierda a derecha y de derecha a izquierda).

      1) Las entradas de esta función serán la matriz **sopa** o sus filas (lo que más le convenga a usted) y la palabra a buscar.

2) La salida será la posición (componentes) de las letras de la palabra buscada sobre la matriz. El orden en que se reportarán las componentes, será el de las letras de la palabra leída correctamente.

*c*) Una función que realice la búsqueda de la palabra deseada en las columnas de la matriz sopa. La búsqueda se llevará a cabo en ambos sentidos (de arriba hacia abajo y viceversa).

1) Las entradas de esta función serán la matriz **sopa** o sus columnas (lo que más le convenga a usted) y la palabra a buscar.
2) La salida será la posición (componentes) de las letras de la palabra buscada sobre la matriz. El orden en que se reportarán las componentes, será el de las letras de la palabra leída correctamente.

*d*) Una función que cree o actualice, según aplique, un archivo de texto de nombre *resultados.txt* , donde se escribirán las palabras encontradas y la ubicación de sus letras en la matriz **sopa**.

1) La entrada será la palabra que se buscó en la matriz **sopa** y la posición de sus letras en perfecto orden (sólo si se encontró).
2) Esta función no tendrá salida en forma de variable; sin embargo, de ella se obtendrá el archivo actualizado hasta la última búsqueda realizada. En cada actualización se escribirá la palabra buscada, y de encontrarse en **sopa**, la ubicación de sus letras.

*e*) Una función de nombre **interfaz6** que será la única que el usuario deberá invocar. Esta función utilizará las anteriores para ubicar cada palabra en la sopa de letras y anotar los resultados en el archivo especificado. La función podrá recibir como argumento la palabra a buscar; pero si ésta no se incluye como argumento al invocar la función, ésta no fallará y, por el contrario, invitará al usuario a introducir la palabra a través del teclado.

Cada vez que se complete la búsqueda (vertical y/u horizontal) de una palabra y se reporte en *resultados.txt*, la función **interfaz6** dará oportunidad al usuario de introducir una nueva palabra para ubicarla en la misma sopa de letras. Esto se repetirá hasta que el usuario introduzca la palabra **"end"**.

- Usted decida cuál o cuáles deben ser las entradas de la función **interfaz6**.
- Igual que en el caso anterior, no existe una variable de salida pero sí se obtendrá como producto final el archivo *resultados.txt* actualizado tras cada búsqueda. SÓLO SE REQUIERE UNA OCURRENCIA DE LA PALABRA, ES DECIR, CUANDO SE ENCUENTRE UNA VEZ EN EL ARCHIVO, NO ES NECESARIO SEGUIR BUSCANDO ESA PALABRA.

EJEMPLO

Dado el archivo **sopita.txt** con el siguiente contenido:

```
4 5
dando
estoy
uneje
mplox
```

y la palabra **"yo"** para ser buscada, su programa extraerá la información necesaria y guardará en la matriz **sopa** de cuatro filas y cinco columnas, lo siguiente:

```
sopa = ['dando'
        'estoy'
        'uneje'
        'mplox']
```

Luego buscará la palabra **yo**, encontrándola por ejemplo en las posiciones [sopa(2,5), sopa(1,5)] <primero la posición de la "y" y luego la de la "o">, porque se realizó primero la búsqueda vertical. Y posteriormente escribirá los resultados en el archivo de salida como:

```
palabra: yo
posiciones: (2,5) (1,5)
```

Luego preguntará al usuario por otra palabra y se repiten estos pasos. Si la siguiente palabra no estuviera en la sopa de letras, por ejemplo: **caja**, la actualización del archivo de salida produciría:

**palabra: yo**

**posiciones: (2,5) (1,5)**

**palabra: caja**

**posiciones:**

Y esto se repetirá hasta que el usuario introduzca la palabra **"end"**.

A fin de que pueda probar sus programas, se le proporciona el archivo de texto *test1.txt* , que contiene una sopa de letras bajo el formato mostrado en el ejemplo dado arriba. En cuanto a las palabras a ubicar, a continuación se le sugieren unas cuantas, pero puede utilizar las que usted quiera:

**SALMON PERCA TRUCHA CARPA GOBI REIR PIRATA SONREIR LAPA PALA LAS SAL.**

11. Convierta el pseudocódigo que se muestra a continuación en una función de Matlab. Las entradas, salidas y valores por defecto, deberán extraerse del pseudocódigo mismo.

> **Paso 0**: Leer una función $f : \mathbb{R} \to \mathbb{R}$, un intervalo cerrado $[a, b]$, donde $f$ es continua, un número máximo de iteraciones *maxiter* y una tolerancia $tol >= 10^{-4}$. Si la tolerancia no cumple esta condición, el programa informará al usuario del error y terminará.
> **Paso 1**: Si $|a - b|$ <1 % de la tolerancia, se considerará que la integral es nula y se devolverá el valor 0. Si no ocurre lo anterior pero ocurre que $a - b > tol$, se intercambiarán los valores de las variables $a$ y $b$.
> **Paso 2**: Hacer $divs = 100$ , $k = 1$ , $Int_{k-1} = 0$ , $err_k = 10^{10}$
> **Paso 3**: Mientras se cumpla que $err_k > tol$ y que $k <= maxiter$ ejecutar los pasos desde el 3.1 hasta el 3.6.
>
> **Paso 3.1**: Dividir el intervalo $[a, b]$ en *divs* partes iguales.
> **Paso 3.2**: Hallar el área de los trapecios formados en cada división del intervalo $[a, b]$ cuyos cuatro vértices coinciden con los extremos de cada división del intervalo y los valores funcionales para esos extremos.
> **Paso 3.3**: Calcular $Int_k$ =suma de todas las áreas calculadas en el paso 3.2
> **Paso 3.4**: Calcular $err_{k+1} = |Int_k – Int_{k-1}|$
> **Paso 3.5**: Incrementar *divs* en 10 % y redondearlo al entero más cercano.
> **Paso 3.6**: Incrementar $k$ en una unidad y regresar al paso 3.
>
> **Paso 4**: Devolver el valor de la integral contenido en $Int_k$

Luego, cree una interfaz con el usuario, que se encargará de invocar la función representada en el pseudocódigo anterior, para integrar el producto de dos funciones provenientes de un archivo ascii cuyo formato es el mismo que se usó en el ejercicio 3.

Las entradas que el usuario proveerá a la interfaz serán:

(i) Los números de las dos funciones cuyo producto se integrará.

(ii) El intervalo de integración.

(iii) La tolerancia.

La salida será: el valor de la integral o el mensaje de error que se podría generar en el paso 0.

12. Se tiene un conjunto de datos $(x_i, y_i)$, $i = 1, 2, \ldots, n$, contenidos en un archivo de texto, los cuales están escritos en dos columnas, la primera corresponde a las abscisas (los $x_i$) y la segunda a las ordenadas (los $y_i$). En otro archivo de texto hay un conjunto de numeros enteros positivos.

Se le pide escribir una funcion en Matlab, de nombre *polyaprox*, bajo las siguientes especicaciones:

- Los parámetros de entrada son: el vector de abscisas, $x$, el vector de ordenadas, $y$, y un vector *grados* de números enteros positivos.
- No hay parámetros de salida.
- Se debe verificar la consistencia de los parámetros de entrada, es decir, que $x$ e $y$ sean vectores de números, que $x$ e $y$ tengan la misma longitud, y que *grados* sea efectivamente un vector de números enteros positivos. En caso de que alguna de estas condiciones falle, se imprimirá un mensaje alusivo en pantalla y se detendrá la ejecucion de la función mediante el comando `error` de Matlab.
- Para cada entero positivo $k$ contenido en el vector *grados* se debe:

- Calcular el polinomio de ajuste, $p$, de grado $k$, para los datos dados.
- Calcular e imprimir en pantalla el error cuadrático asociado al polinomio de ajuste $p$; la fórmula de dicho error es $\sum_{i=1}^{n} (y_i - p(x_i))^2$. Debe indicar, en el mensaje por pantalla, cuál es el grado del polinomio de ajuste que corresponde al error cuadrático impreso.

- Se deben hallar los polinomios de ajuste $p1$ y $p2$, cuyos errores cuadráticos asociados son, respectivamente, el valor mínimo y el valor máximo de todos los errores cuádraticos calculados en el ciclo anterior.
- Usando los comandos `subplot` y `plot`, se deben graficar, en una misma ventana, el polinomio $p1$ en el cuadrante de la izquierda y el polinomio $p2$ en el cuadrante de la derecha, incluyendo los datos dados en ambos gráficos. Es importante que los polinomios sean graficados con trazo continuo de color azul, mientras que los datos deben ser graficados usando un asterisco rojo. Agregue un título a ambos gráficos, donde se especifique el grado del polinomio de ajuste correspondiente y el error cuadrático asociado.

13. La determinación de la cantidad de calor requerido para elevar la temperatura de un material es un problema usual en la ingeniería química. La característica necesaria para llevar a cabo este cálculo es la capacidad calorífica $c$. Este parámetro representa la cantidad de calor requerida para elevar una unidad de temperatura en una unidad de masa.

    La capacidad calorífica varía con la temperatura. Más aún, se ha reportado que a temperaturas superiores a 0°C la variacion de $c(T)$ puede representarse mediante expansiones polinómicas en $T$ de diferente orden (grado), esto es:

$$c(T) = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + \cdots + a_n T^n \tag{1}$$

    Por otra parte, cuando se requiere calcular cambios entálpicos, entre otros, resulta conveniente estimar la capacidad calorífica media, $\bar{c}(T)$, entre dos temperaturas dadas $T1$ y $T2$. Matemáticamente, ésta puede calcularse mediante la relación integral:

$$\bar{c}(T) = \frac{\int_{T_1}^{T_2} c(T)dT}{T_2 - T_1} \tag{2}$$

    El archivo *dataSO3.txt* contiene los registros de la capacidad calorífica del trióxido de azufre, $SO3$, en unidades molares consistentes y en función de la temperatura $T$ en grados centígrados. Se ha empleado a un Ingeniero Químico o a un Matemático Aplicado (Ud.) para que proponga una expresión polinómica mediante la cual puedan ser modelados los valores reportados en el archivo mencionado anteriormente. Para ello, Ud. deberá proceder siguiendo los pasos listados a continuacion:

    *a)* Aproxime tres polinomios de diferentes órdenes a los datos contenidos en el archivo, empleando para ello la función `polyfit` de Matlab; almacene los coeficientes resultantes.

    *b)* Grafique mediante el comando `plot` y de forma discreta los puntos $T$ y $c(T)$ contenidos en el archivo de datos.

    *c)* Genere una partición $TT = 27 : 0.5 : 1227$ y evalúe en ésta cada uno de los polinomios obtenidos en *a)*. Emplee el comando `polyval` para realizar tales evaluaciones.

    *d)* Grafique los polinomios de aproximación contra los puntos originales y cualitativamente diga cuál aproximación le parece más precisa. Identifique mediante una leyenda cada uno de los polinomios graficados según su grado.

    *e)* Para el polinomio escogido en el ítem anterior, evalúe las temperaturas $427°C$, $627°C$, $927°C$ y calcule el error de aproximación con respecto a la capacidad calorífica correspondiente reportada en el archivo de datos. Escriba sus conclusiones.

    *f)* Calcule la capacidad calorífica media entre las temperaturas $327°C$ y $927°C$ para el polinomio escogido en *d)*. Emplee para ello las funciones de integración por rectángulos programada por Ud. previamente (si no ha programado ninguna, se le recomienda hacerlo).

    *g)* Para los tres polinomios calculados anteriormente estime el valor de la temperatura para la cual se tiene una capacidad calorífica de $c = 18.00$ (unidades molares). Emplee para ello el método numérico descrito en el ejercicio 5 (si no lo ha programado, ¿qué espera?).