Fundamentos de Programación con MATLAB

Ejercicios

1. **Lectura y comprensión de código**

El siguiente código es un temporizador de cuenta atrás y está escrito en el lenguaje de scripting `Bash`. `Bash` (acrónimo de Bourne-again shell) es un intérprete de comandos y lenguaje de scripting desarrollado para sistemas operativos Unix-like, como Linux, macOS y Windows (a través de Cygwin, MinGW o WSL). Traduce el código a MATLAB.

```
#!/bin/bash

set -uo pipefail

function timer() {
  local tarea=$1
  local minutos=$2

  local segundos=$((minutos * 60))

  echo "Comenzando: $tarea"

  while [[ $segundos -ge 0 ]]
  do
    local min=$((segundos / 60))
    local seg=$((segundos % 60))

    printf "\rTiempo Restante: %02d:%02d" $min $seg

    sleep 1

    #segundos=$((segundos - 1))
  done

  printf "\nTiempo dedicado a $tarea terminado\n"
```

```
}

timer "$@"
```

2. **Integración Numérica**

Existen diversas estrategias para aproximar numericamente la integral de una función. Uno de estos métodos es la conocida **Regla de Simpson** (nombrada así en honor de Thomas Simpson):

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Para aplicar la regla de Simpson realice lo siguiente: se divide el intervalo $[a, b]$ en n subintervalos iguales (con n par), de manera que $x_i = a + ih$, donde $h = \frac{b-a}{n}$ para $i = 0, 1, 2, 3, \ldots, n$. Implementando la regla de Simpson en cada subintervalo obtenido:

$$[x_{j-1}, x_{j+1}], \quad j = 1, 3, 5, \ldots, n-1,$$

se obtiene que:

$$\int_{x_{j-1}}^{x_{j+1}} f(x)dx \approx \frac{x_{j+1} - x_{j-1}}{6}[f(x_{j-1}) + 4f(x_j) + f(x_{j+1})]$$

Sólo hace falta sumar las aproximaciones de la integral de todos los subintervalos:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(a) + 2\sum_{k=1}^{\frac{n}{2}-1} f(x_{2k}) + 4\sum_{k=1}^{\frac{n}{2}} f(x_{2k-1}) + f(b)\right]$$

Implementa la regla de Simpson antes mostrada en MATLAB.

3. Método del Gradiente Conjugado

En matemática, el **método del gradiente conjugado** es un algoritmo para resolver numéricamente los sistemas de ecuaciones lineales cuyas matrices son simétricas y definidas positivas.

Supongamos que queremos resolver el siguiente sistema de ecuaciones lineales

$$Ax = b$$

donde la matriz A es de tamaño $n \times n$, es simétrica (i.e., $A^T = A$) y definida positiva (i.e., $x^T A x > 0$ para todos los vectores no cero $x \in \mathbb{R}^n$). Denotamos la única solución de este sistema por $\overline{x}$.

El algoritmo del Gradiente Conjugado para aproximar la solución del sistema de ecuaciones lineales $Ax = b$ se muestra a continuación:

$$
\begin{aligned}
& r_0 \leftarrow b - Ax_0 \\
& \text{if } \|r0\| < tolerance \text{ then} \\
& \qquad \text{return } \ x0 \qquad // \text{ as solution of } \ Ax = b \\
& \text{end if} \\
& p_0 \leftarrow r_0 \\
& k \leftarrow 0 \\
& \text{repeat until } \ n \\
& \qquad a_k \leftarrow \frac{r_k \cdot r_k}{p_k \cdot Ap_k} \\
& \qquad x_{k+1} \leftarrow x_k + a_k p_k \\
& \qquad r_{k+1} \leftarrow r_k - a_k A p_k \\
& \qquad \text{if } \|r_{k+1}\| < tolerance \text{ then} \\
& \qquad \qquad \text{return } \ x_{k+1} \qquad // \text{ as solution of } \ Ax = b \\
& \qquad \text{end if} \\
& \qquad \beta_k \leftarrow \frac{r_{k+1} \cdot r_{k+1}}{r_k \cdot r_k} \\
& \qquad p_{k+1} \leftarrow r_{k+1} + \beta_k p_k \\
& \qquad k \leftarrow k + 1 \\
& \text{end repeat}
\end{aligned}
$$

El vector inicial x_0 puede ser una aproximación a la solución o $\mathbf{0} \in \mathbb{R}^n$.

Cree un programa que lea la matriz almacenada en `matrices_tarea.xlsx` (hoja 1) y el vector almacenado en `matrices_tarea.xlsx` (hoja 2) y que calcule la solución del sistema utilizando el método del gradiente conjugado. Primero se debe verificar que la matriz sea simétrica y definida positiva.

Sugerencias:

- Para saber si una matriz es definida positiva calcule sus eigenvalores y vea que todos los eigenvalores sean mayores que cero.
- Utilice las funciones incorporadas `issymmetric`, `all` y `eig` de MATLAB.

4. Utilice el archivo `csv` llamado `lancha2_221016211519_X1548.csv` y realice las siguientes gráficas:

Variable	**Tipo de gráfico recomendado**
Presión vs. Tiempo	Gráfico de líneas
Temperatura vs. Tiempo	Gráfico de líneas
Conductividad vs. Tiempo	Gráfico de líneas
Presión vs. Temperatura	Gráfico de dispersión

body { font-size: 11pt; /* Texto normal */ line-height: 1.6; /* Espaciado entre líneas */ } h1 { font-size: 20pt; } h2 { font-size: 16pt; } h3 { font-size: 14pt; } h4 { font-size: 12pt; } code { font-size: 10pt; /* Código inline */ } pre code { font-size: 9pt; /* Bloques de código */ }