PRÁCTICA 3. Programación de ficheros M

Periodo de realización: Semanas 3 y 4 del curso
Fecha límite de entrega: 25 de marzo de 2012

Se pide subir al Moodle un único fichero **`apellido_p3.pdf`** con la solución de los siguientes ejercicios (incluir tanto las instrucciones empleadas como los resultados obtenidos comentados). Para el formato de presentación, consultar el fichero **`plantilla_prácticas.pdf`**.

1. Programación de *scripts*

Ejercicio 1. Entradas (teclado) y salidas (pantalla). Se trata de programar un sencillo juego consistente en que el jugador tiene 7 oportunidades para acertar un número entero del 1 al 100. A cada intento `x`, MATLAB debe informar al jugador si el número a adivinar es mayor o menor que `x`, y cuántas oportunidades más le quedan.

Para ello, editar las instrucciones necesarias para hacer, como mínimo, lo siguiente:

1) Con ayuda de la función **`input`**, el programa pregunta el nombre al jugador y lo guarda en la variable (de clase *char*) `nombre`.

2) A continuación, con **`disp`**, el programa informa al jugador (llamándolo por su nombre) de que dispone de 7 oportunidades para acertar un número del 1 al 100.

3) El programa genera un número `n` entero y aleatorio entre 1 y 100. (Funciones: **`rand`**, **`fix`** o **`ceil`**)

4) Dentro de un bucle **`while...end`** (a ejecutar como máximo 7 veces), el programa solicita al jugador el número `x` (**`input`**), lo compara con `n` (**`if...elseif...else...end`**), e informa al jugador de si `x` es mayor o menor que `n` y le indica cuántos intentos le quedan (**`disp`**, **`num2str`**).

5) Cuando el juego termina, el programa informa al jugador si ha ganado o ha perdido (**`disp`**). En este último caso, le informa además de cuál era el número `n` (**`num2str`**).

Probar el juego (notar que con una buena estrategia es posible ganar siempre). Ilustrar el funcionamiento del script con una partida donde se gane y otra donde se pierda. □

Ejercicio 2. Forma y caracterización de la respuesta indicial de un sistema de segundo orden (prototipo). Se trata de estudiar con detalle la respuesta indicial (a un escalón unitario) del sistema de segundo orden:

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2},$$

1) Suponiendo $\omega_n = 1$, dibujar y representar en una misma gráfica la respuesta indicial para ζ = 0.0, 0.2, 0.4, 0.6, 0.8, 1, 1.2. (**`linspace`**, **`plot`**, **`step`**, **`for...end`**, **`hold on/off`**)

2) Añadir al *script* anterior las instrucciones necesarias para que además aparezcan automáticamente por pantalla los valores de la siguiente tabla:

ζ	t_D	t_r	t_r'	t_r''	t_s	t_s'
0.0						
0.2						
0.4						
0.6						
0.8						
1.0						
1.2						

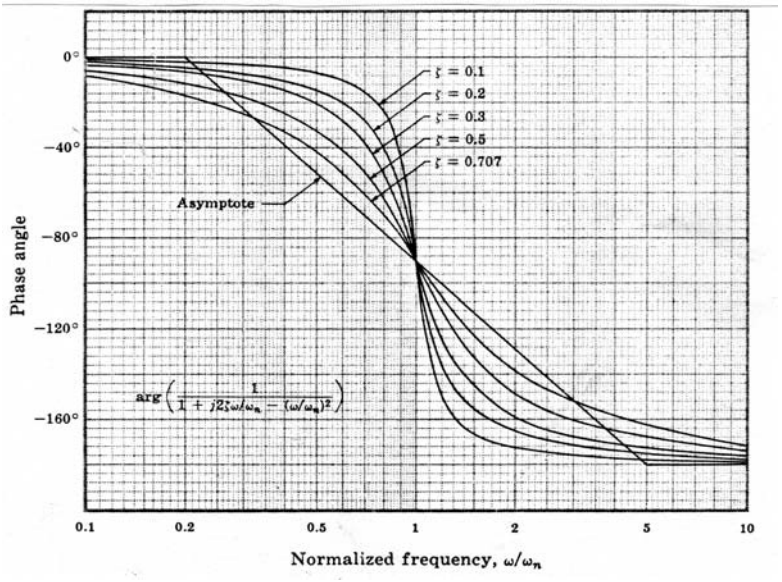
Tabla 1

donde t_D (tiempo de retardo) es el tiempo que transcurre hasta que la respuesta alcanza el 50% de su valor de régimen; t_r (tiempo de subida) es el tiempo empleado en pasar del 0% al 100% de su valor de régimen; t_r' ídem en pasar del 10% al 90%; t_r''= 1/(pendiente en t_D); t_s (tiempo de establecimiento) es el tiempo que transcurre hasta que las oscilaciones alrededor del valor de régimen no superan el 2%; t_s' ídem pero la variación considerada es del 5%. (Funciones **`find`**, **`if...else...end`**, **`>=`**, **`<=`**, **`|`**, **`&`**, **`isempty`**, **`rot90`** o **`fliplr`**, **`disp`**) □

Ejercicio 3. Curvas normalizadas de módulo y fase. Representar las curvas normalizadas de ganancia en dB y fase de la respuesta frecuencial de un sistema de segundo orden:

$$H(j\omega) = \frac{1}{(j\omega)^2 + 2\zeta j\omega + 1}$$

La siguiente figura es una sugerencia de formato para el diagrama de fase (funciones **`logspace`**, **`bode`** o **`freqs`**, **`abs`**, **`angle`**, **`log10`**, **`hold`**, **`for...end`**, **`text`**, **`ylabel`**, **`xlabel`**).



Opcional: Repetir para un sistema de primer orden. □

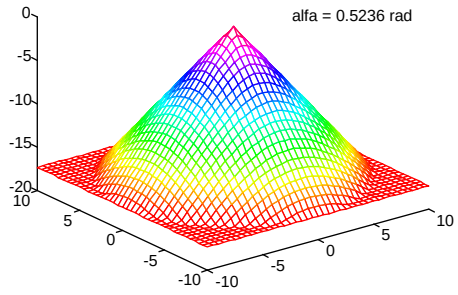
2. Programación de *functions*

Ejercicio 4. Funciones. Argumentos de entrada y salida. Se pide editar una función **conos** que represente conos y troncos de conos y cuya ayuda sea la siguiente:

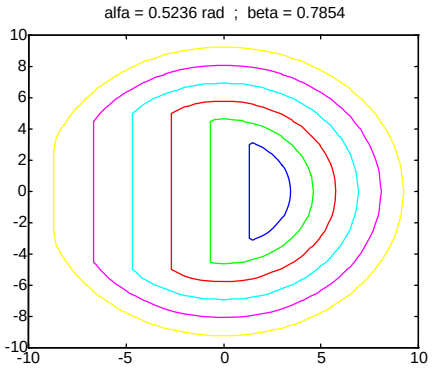
```
» help conos

CONOS:  función que representa conos y troncos de cono
z=conos(alfa)         Calcula conos de semiangulo alfa (grados)
                      Para representarlos hacer mesh(z) o contour(z)
z=conos(alfa,beta)    Calcula conos de semiangulo alfa y corte beta (grados)
                      Para representarlos hacer mesh(z) o contour(z)
conos(alfa)           Representación directa en 3D
conos(alfa,beta)
conos(alfa,'c')       Representación directa del contorno
conos(alfa,beta,'c')
```

Para verificar el correcto funcionamiento de la función se probarán las dos instrucciones siguientes:

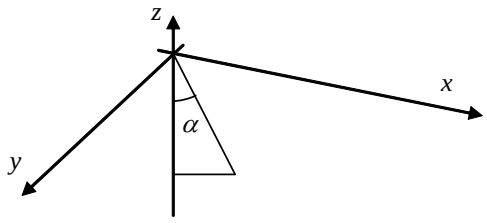
```
» conos(30)
```



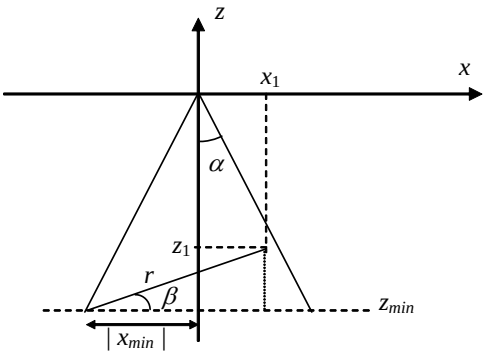

```
» conos(30,45,'c')
```



Las siguientes figuras muestran cómo generar conos y troncos de cono respectivamente:



$$\left.\begin{array}{l}\operatorname{sen}\alpha = \sqrt{x^2+y^2}\\ \cos\alpha = z\end{array}\right\} \Rightarrow z = -\frac{\sqrt{x^2+y^2}}{\operatorname{tg}\alpha}$$



$$\left.\begin{array}{l} r\cos\beta = |x_{min}| + x_1\\ r\operatorname{sen}\beta = z_1 - z_{min}\end{array}\right\} \Rightarrow z_1 = \operatorname{tg}\beta\cdot\left(|x_{min}| + x_1\right) + z_{min}$$

Representación de conos: Los pasos a seguir dentro de la función son los siguientes:

1) Generar un vector `x=-10:0.5:10;` (41 muestras) y un vector `y=x;`.

2) Aplicar **`meshgrid`** para obtener `xx` e `yy` y, a partir de éstos, obtener los valores `z` del cono.
3) Verificar que la columna central de `xx` (la 21-ésima) es la que contiene el valor central de `x`. Ídem con la fila central de `yy`. A fin de representar el "suelo", forzar que todos los valores de z cuyo valor sea inferior a `z(1,21)` (o bien `z(21,1)`) tomen el valor `z(1,21)` (o bien `z(21,1)`). (Funciones **`find`**, **`for...end`**)
4) Representar el cono con **`mesh`** y/o el contorno con **`contour`**. Poner un título (**`title`**) indicando el valor de α en radianes (**`num2str`**).

Representación de cortes de conos:

1) Modificar la función para que también dé como resultado cortes de conos.
2) Utilizar **`nargin`**, **`nargout`** o **`varargin`**, **`varargout`** para que el resultado de la función sea uno u otro según el número de argumentos con que se la invoca. □

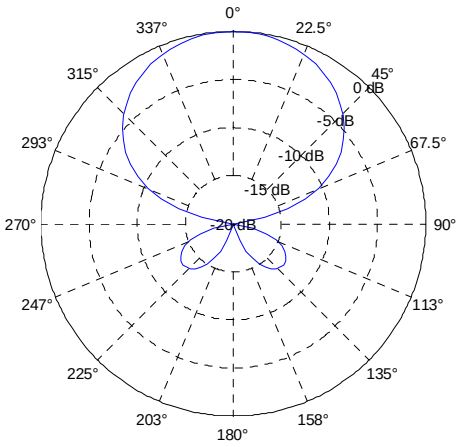
Ejercicio 5. Diagrama polar en dB.

1) Generar y representar en coordenadas polares (**`linspace`**, **`abs`**, **`sin`**, **`polar`**) la expresión:

$$y = \left| \frac{\sin(2\theta)}{2\theta} \right| \quad \text{para} \quad -\pi \le \theta \le \pi$$

2) Representar en coordenadas cartesianas $20\log_{10}(y)$ en función de θ (**`log10`**, **`plot`**).
3) Crear una función de nombre **`polardB.m`** que permita representar $20\log_{10}(y)$ en polares y que genere exactamente el mismo gráfico de la figura. Los argumentos de entrada deben ser "**`theta`**", "**`y`**" y "**`min_db`**". Este último parámetro corresponde al valor "central" de la representación. El resultado de la ejecución para **`min_db=-20`** es el que se muestra en la figura.
4) Ejecutar de nuevo la función para **`min_db=-40`**.

(Funciones: **`linspace`**, **`exp`** o **`cos`**/**`sin`**, **`plot`**, **`axis`**, **`text`**, [**`patch`**], **`hold`**)



3. Funciones que llaman a funciones

Ejercicio 6. Solución de ecuaciones trascendentes.

1) Se quiere resolver la ecuación $\frac{\phi - \sin\phi}{2} = \frac{A}{r^2}$ con A = 0.0472 y r = 2. Para ello crear una función con la ecuación a resolver, $y = \frac{\phi - \sin\phi}{2} - \frac{A}{r^2}$, cuyo argumento de entrada sea ϕ y cuyo argumento de salida sea y. A continuación, aplicar la función **`fzero`** o la función **`fsolve`**.

2) Opcional: Considerar las dos curvas definidas por

$$p_2 - c_2 = \frac{1}{\alpha\left(1-\left(\frac{e^{x_2\beta-\alpha p_2}}{e^{x_2\beta-\alpha p_2}+e^{x_1\beta-\alpha p_1}}\right)\right)}$$

$$p_1 - c_1 = \frac{1}{\alpha\left(1-\left(\frac{e^{x_1\beta-\alpha p_1}}{e^{x_2\beta-\alpha p_2}+e^{x_1\beta-\alpha p_1}}\right)\right)}$$

donde $c^T = \begin{pmatrix} c_1 & c_2 \end{pmatrix} = \begin{pmatrix} 2.8724 & 2.8839 \end{pmatrix}$, $x^T = \begin{pmatrix} x_1 & x_2 \end{pmatrix} = \begin{pmatrix} 20 & 19 \end{pmatrix}$, $\alpha = 0.172188$ y $\beta = 0.4$. Las coordenadas de representación son, en ambos casos (p_1, p_2). Se trata de obtener el punto de cruce de ambas curvas numéricamente. A continuación, verificar el resultado representando ambas curvas con **`ezplot`**. □

Ejercicio 7. Solución de ecuaciones diferenciales. Atractor caótico. Las ecuaciones de estado de un atractor caótico vienen dadas por las ecuaciones de Lorentz:

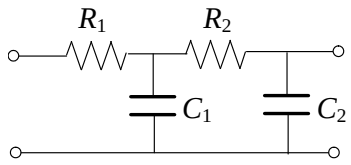
$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{pmatrix} = \begin{bmatrix} -\beta & 0 & y_2 \\ 0 & -\sigma & \sigma \\ -y_2 & \rho & -1 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

se pide:

1) Editar una función que contenga dichas ecuaciones de estado. Tomar σ = 10, ρ = 28 y β = 8/3.

2) Aplicar la función **`ode23`** con condiciones iniciales CI = (0 0 ε)^T, (utilizar la variable predefinida **`eps`**), $t_{inicial}$=0s y t_{final}=100s. Representar el resultado en 3D (**`comet3`**). □

Ejercicio 8. Optimización paramétrica de un filtro. Se desea realizar un filtro pasa-bajas con ω_b = 1000rad/s. Para ello se monta el circuito de la figura con $C_1 = C_2 = 1nF$.



Por el método de mallas, se obtiene que la función de transferencia del filtro es

$$H(s) = \frac{\frac{10^{18}}{R_1 R_2}}{s^2 + \left(\frac{10^9}{R_2} + \frac{10^9}{R_1}\right)s + \frac{10^{18}}{R_1 R_2}}$$

Se trata de determinar los valores de R_1 y R_2 que ajustan de manera óptima la respuesta real del filtro a la ideal deseable. Como criterio de ajuste se utiliza $J = \sum e_i^2$ siendo e_i el error o diferencia entre la atenuación ideal y la real:

$$e_i = \left|A(\omega_i) - 1\right| \text{, para } \omega_i < 1000 \quad \text{y} \quad e_i = \left|A(\omega_i)\right| \text{, para } \omega_i \geq 1000$$

donde $A(\omega)=|H(j\omega)|$ es el módulo de la respuesta frecuencial. (Notar que el módulo de la respuesta ideal vale 1 en la banda [0,1000] y vale 0 para frecuencias superiores a 1000rad/s).

En concreto se pide:

1) Editar una función que, a partir del argumento de entrada [R_1 , R_2], calcule el valor de J. **Nota:** Calcular los errores e_i para las frecuencias ω_i = 0, 100, 500, 800, 900, 1000, 1100, 1200 y 1500. (Funciones: **`function`**, **`freqs`**, **`for...end`**, **`if...else...end`**, **`abs`**).

2) Optimización numérica: Con ayuda de **`fminsearch`** determinar los valores óptimos de R_1, R_2 y J.

3) Opcional: Optimización gráfica: Representar (en 3D) el criterio J en función de R_1 y R_2 (**`meshgrid`**, **`mesh`**). Indicar el par (R_1 , R_2) que minimiza J en el diagrama de contorno (**`contour`**, **`clabel`**, **`text`**).

4) Opcional: Representar el módulo del filtro pasa bajas ideal y compararlo con el módulo del filtro obtenido. □

4. Otras aplicaciones

Ejercicio 9. Interpolación. Considerar el fichero **`sonda.m`** de la intranet de la asignatura (Hanselman). Se pide:

1) Representar los datos con **`mesh`**.

2) Con ayuda de la función **`interp2`** obtener los puntos intermedios y representar la curva suavizada resultante. □

Ejercicio 10. Procesado de audio. Considerar los archivos **`audio1.wav`** y **`audio2.wav`** de la intranet de la asignatura. Se pide:

1) Cargarlos en MATLAB con **`wavread`** y escucharlos con **`sound`**. Opcional: ¿Identifica a los intérpretes y temas?

2) Obtener y representar en una misma figura los espectros. Para obtener los espectros usar cualquier método del apéndice del Tema 3 (periodograma, Blackman-Tukey, Welch, Barlett,…). ¿Cuál de los dos espectros se asemeja más al ruido blanco?

3) Opcional: Realizar algún tipo de manipulación de la señal (filtrado, *scrambling*,…) y escuchar el resultado. □