GNU Octave

Curso Introductorio (23 de agosto de 2019)

Vero Bustamante, Fernando Danko

2do cuat. 2019

LABI – FIUBA

Temario

1. Laburando con matrices
2. Manejo de archivos e impresión por pantalla
3. Funciones
4. Gráficos Bi-dimensionales
5. Índices y Rangos
6. Graficos de superficies
7. Ecuaciones diferenciales
8. Ajuste de Curvas

Aviso sobre Interfaz gráfica

Octave (desde la versión 4.0 en adelante) posee una interfaz gráfica por defecto. Vamos a trabajar sobre ella.

Si tenés una versión anterior de Octave (a partir de la 3.8), podés iniciar la interfaz gráfica así:

En UNIX[1] Ejecutar en consola `$ octave --force-gui`

En Windows Botón derecho en el acceso directo > Propiedades. En el campo destino, agregar al final `--force-gui` y apretar Aceptar.

[1] Linux, OS X, BSD, etc.

Importancia de la ventana de Comandos

Octave se comunica con nosotros a través de la ventana de comandos.

Octave se tildó :(

Muchas veces sucede que nuestro código no ejecuta, o parece que se queda tildado el programa. Lo primero que debemos hacer es abrir el Panel *Command Window* y leer lo que diga allí.

Consideraciones

En lugar de tipear los comandos uno tras otro, conviene crear un script.

Un script es un simple archivo de texto donde estan todos los comandos que Octave debe ejecutar.

Llamamos *ejecutar* o *correr* (run) un script, a la acción de hacer que Octave ejecute los comandos listados en el script.

Consideraciones

¡Importante!

Para correr un script, es necesario que el intérprete de Octave esté trabajando en la misma carpeta donde el script está guardado.

Todo script debe terminar con un .m y no contener espacios

Desde la GUI, se puede correr un script presionando el boton ”Save File and Run”del editor. Desde consola, basta con escribir el nombre del script (sin el .m) y presionar enter.

Creando un script

Desde consola: podemos crear o editar un script utilizando el comando `edit` seguido por el nombre del script. Si no existe, creará un script nuevo.

Desde la GUI: En el panel File Browser, navegar a la carpeta adecuada, presionar el ícono de engranaje y elegir ”New File”. A continuación, escribir un nombre terminado en .m (según lo dicho anteriormente). Para editar el script, hacer doble click sobre éste.

Comentarios extra de variables

```
> a = 3*7 + 2 - 7^2
> 3*a + 17
> 3a + 17 % Esto no funciona, falta el simbolo de por
> d = 1000*a;  % El ; hace que no diga el resultado
> d  % Para mostrar el valor de una variable
> disp(d)  % Esto hace lo mismo per es más elegante (?)
```

Laburando con matrices

Transpuesta y Transpuesta conjugada (hermítica)

Para obtener la transpuesta hermítica de una matriz (o vector), se utiliza el operador `'` (*apóstrofo*[2] *simple*).

Para transponer una matriz (o vector) sin conjugar, se debe utilizar el operador `.'`

Ejemplo:

```
> A = [1-1j 2; 3 4]
> A_h = A'
1 + 1j    3
2         4
> A_t = A.'
1 - 1j    3
2         4
```

[2]Apóstrofe: Insulto que provoca y ofende.| Figura retórica que consiste en interrumpir el discurso para dirigirse con vehemencia a otra persona, generalmente con un tono patético o de lamento.

¿Vectores filas o columnas?

Pensemos en una tabla de valores donde se anota la evolución de distintas magnitudes en el tiempo: en cada columna hay distintas mediciones de un mismo dato. Ésta misma idea es en Octave: cada columna representa un distinto tipo de dato.

¿Por qué pasa? En general los datos los obtenemos de un archivo y los ponemos en una matriz en memoria.

¿Vectores filas o columnas?

Estos dos vectores columna pueden acomodarse cómodamente en una matriz, por ejemplo:

```
mediciones = [% tiempo (seg), tension (V)
              0.0      1.30
              1.0      6.40
              1.71    16.1
              2.43    31.1
              ];
```

Se pueden recuperar las filas o columnas de esta forma:

```
tiempo = mediciones(:,1); % fila: todas, columna: 1
tension = mediciones(:,2); % fila: todas, columna: 2
cond_inic = mediciones(1,:); % fila: 1, columna: todas
```

Multiplicando matrices I

Si las dimensiones de las matrices (o vectores) concuerdan, se pueden multiplicar usando el operador * De la misma forma, podemos hacer potencias de matrices cuadradas, con la definición de $\mathsf{A}^n = \underbrace{\mathsf{AAA}\cdots\mathsf{A}}_{n\text{veces}}$

```
> A = [1 0 1; 0 1 0; 1 1 1];
> A*A*A
> A^3
```

Operaciones elemento a elemento

Supongamos que medimos la tensión sobre un resistor de 8Ω y tenemos esa información en un vector columna V. Queremos obtener la potencia disipada, dada por la ecuación $P = V^2/R$ ¿Cómo podemos hacer eso?

Si realizamos la operación `V^2` en Octave, devuelve error, ya que V no es una matriz cuadrada. Pero en realidad, lo que queremos hacer es elevar *cada componente* al cuadrado, y no al vector por así decirlo.

Operaciones elemento a elemento

Para indicarle a Octave que una operación debe realizarse elemento a elemento, se antepone un punto. Por ejemplo

- Multiplicación `.*`
- División `./`
- Potencia `.^`

En nuestro caso, tendríamos que hacer:

```
> V = [10.4 7.5 12.4 9.2 7.3].'; % Datos de ejemplo
> R = 8;
> P = V.^2 ./ R
```

Operaciones elemento a elemento

Ejemplo 2: Supongamos que tenemos un vector con frecuencias angulares ω, queremos convertir esos datos al período correspondiente T según la ecuación $T = 2\pi/\omega$, tenemos entonces:

```
> w = [132.43  22.54  563.01].';  % Datos
> T = 2*pi ./ w
```

Broadcasting[3]

¡Esto no es MATLAB!

Ésta es una característica que MATLAB no poseé –y que seguramente no implementará por retrocompatibilidad– así que hay que tener cuidado si se quiere interoperabilidad.

Octave permite multiplicar matrices si sus dimesiones son *parecidas* pero no compatibles formalmente. Por ejemplo, se puede multiplicar una matriz de $n \times 3$ por un vector de 1×3 o 3×1. Lo que hace es multiplicar cada columna por el valor que dice el vector.

Ésta no es una funcionalidad sumamente empleada, pero tenía que contárselas. A veces es útil –depende de cuán loco esté cada uno–. Al menos para que sepan por qué a veces no explota algo que no tendría que funcionar.

[3] Es algo como “Ajuste automático de ancho” en español.

Manejo de archivos e impresión por pantalla

Imprimir por pantalla

```
>>a=12;

>> printf("cantidad de alumnos en el curso: %d \n",a)
cantidad de alumnos en el curso: 12

>> printf("cantidad de alumnos en el curso: %f \n",e)
cantidad de alumnos en el curso: 2.718282
```

Imprimir por pantalla

```
>> printf("cantidad de alumnos en el curso: %.2f \n",e)
cantidad de alumnos en el curso: 2.72

>> printf("cantidad de alumnos en el curso: %.1f \n",e)
cantidad de alumnos en el curso: 2.7

>> printf("cantidad de alumnos en el curso: %.f \n",e)
cantidad de alumnos en el curso: 3

>> printf("cantidad de profesores: %d y %s :% d \n",
a,"alumnos",12)
cantidad de profesores: 10 y alumnos :12
```

Levantar Archivos

```
matriz= csvread("archivo.csv");
matriz= dlmread('archivo.txt',',');
% la "," es el separador
```

Probemos en un llamado **prueba_dlm.m** con:

```
leido = dlmread('respuesta_escalon.txt',',')
leido2 = dlmread('mediciones.csv',',')
leido3 = csvread('mediciones.csv')
```

Funciones

Generalidades

Podríamos separarlas en 3 grupos:

- Funciones internas de Octave
- Funciones creadas por nosotros
- Funciones anónimas

Generalidades

Las funciones pueden tomar ninguno, uno o varios datos, los procesan, y devuelven el resultado.

Funciones Básicas

Algunas son:

Redondeo	Trigonometría	Complejos	Logaritmos
• floor()	• sin()	• abs()	• log()
• ceil()	• sind()	• angle()	• log10()
• round()	• cos()	• real()	• log2()
	• tanh()	• imag()	• exp()

Actividad: Veamos que hace `sind()` usando el comando help o el comando doc seguido del nombre de la función. Por ejemplo: `help sind` o `doc angle`.

Complejos: Forma Binómica y polar

Para obtener la parte real e imaginaria, y el módulo y ángulo de un complejo, tenemos estas funciones:

```
> real(1-2 i )
1
> imag(1-2 i )
-2
> abs(1-2 i )
> angle(1-2 i )
> conj(1-2 i )
1+2 i
```

Probemos esto

```
> abs(-3)
> abs(1+1i)+100*angle(1+1i)
> exp(2-3j)
> sin(1+2j)
> exp(2)*( cos(-3) + 1i*sin(-3) )
> sin(pi)
```

Notemos esta pequeña diferencia:

```
> sin(pi)
> sind(180)
```

Las funciones trigonométricas vienen por defecto en radianes; `sind`, `cosd` y `tand` usan grados sexagesimales.

Función Max

```
> v=[1:1:5  7:-2:1]
v =    1    2    3    4    5    7    5    3    1
> max(v)
ans =   7
> v=[1  2  500j]
v =   1 +    0i       2 +    0i    0 + 500i

> max(v)
ans =      0 + 500i  % la de mayor modulo
```

La función ”min”funciona igual.

Función Mean y funciones round, ceil y floor

```
> A =    [1    2
         3    4
         1    2
         1    5
         1    7];
> mean(A)
    1.4000    4.0000
> mean(A,1)
    1.4000    4.0000

> u= mean(A,2)
    1.5000
    3.5000
    1.5000
    3.0000
    4.0000
```

Función Mean y funciones round, ceil y floor

```
> u= mean(A,2)
    1.5000
    3.5000
    1.5000
    3.0000
    4.0000

> ceil(u)'
    2    4    2    3    4
> floor(u)'
    1    3    1    3    4
> round(u)'
    2    4    2    3    4
```

Funciones para matrices

```
>> A=[1  1;1  0];

>> det(A)
ans = -1
>> trace(A)
ans =  1

>> B=[1  1; 1  1];
>> null(B) %nulo de B
   -0.70711
    0.70711
>> orth(A) %BON espacio columna de A
    0.85065    0.52573
    0.52573   -0.85065
```

Funciones para matrices

```
>> [U,S,V]=svd(A)
U =
   -0.85065   -0.52573
   -0.52573    0.85065

S =
Diagonal Matrix

    1.61803         0
          0    0.61803

V =
   -0.85065    0.52573
   -0.52573   -0.85065
```

Funciones para matrices

```
>> [q,r]=qr(A)
q =

   -0.70711   -0.70711
   -0.70711    0.70711

r =

   -1.41421   -0.70711
    0.00000   -0.70711
```

Funciones para matrices

```
>> [S,D]=eig(A)
S =

    0.52573   -0.85065
   -0.85065   -0.52573

D =

Diagonal Matrix

   -0.61803         0
         0    1.61803
```

Otros comentarios:

La función `length(v)` permite medir el largo de un vector. Aplicada a una matriz, devuelve la dimensión más grande.

`size(M)` devuelve un vector con las dimensiones de la matriz.

`size(M,1)`: cantidad de filas

`size(M,2)`: cantidad de columnas

`numel(M)`: devuelve la cantidad de elementos de M.

Funciones de polinomios

Un polinomio está representado por un vector donde figuran los coeficientes de éste. Por ejemplo el vector: $[3, 2, 1, 2]^T$ representa al polinomio $3x^3 + 2x^2 + x + 2$.

Funciones de polinomios

Las funciones mas comunes para polinomios son:

`r = roots(p): Devuelve un vector r con las raices del polinomio p`

`p = polyfit(x,y, n): Devuelve el polinomio de grado n que mejor ajusta por mínimos cuadrados el conjuntos de puntos $(x_i, y_i)$`

`y = polyval(p, x) % Devuelve el valor del polinomio para el valor x. Si x es un vector o matriz, devuelve un vector o matriz con la evaluación del polinomio p en cada elemento.`

Funciones de polinomios

d = **polyder**(p): Devuelve un vector con los coeficientes del polinomio que resulta de derivar p

s = polyint(p): Devuelve un vector con los coeficientes del polinomio que resulta de integrar p

polyout(p, "x) *% Expresa el polinomio en términos* de la variable x

Funciones de polinomios

```
b = conv(p, q) % Devuelve el vector de coeficientes que
define el polinomio b, producto de los polinomios p y q.

[b, r] = deconv(p, q) % Realiza la división de los
polinomios p y q, devolviendo el polinomio cociente b
y el polinomio resto r.
```

Forma tradicional de definir funciones

Prototipo para definir una función:

```
function resultado = nombre_funcion (argumentos)
   ... cuentas ...
endfunction % no conviene poner simplemente end
```

Multiples argumentos

Ecuación horaria

```
function posicion = obtener_posicion(t,xo,vo,a)
  posicion = xo + vo*t + 0.5*a*t.^2;
endfunction
```

Aquí simplemente se reciben varios parámetros y se calcula la posición de un MRUV.

Multiples argumentos, multiples salidas

Las funciones de Octave pueden devolver más de un valor:

Ecuación horaria

```
function [x,y] = calcular_tiro(t,xo,yo,vo,angulo,g)
  x = xo + vo*cosd(angulo)*t;
  y = yo + vo*sind(angulo)*t - 0.5*g*t.^2;
endfunction
```

Si hace:

```
>> calcular_tiro(10,0,0,10,35,9.8)
ans =  81.915
```

Solamente se obtiene el primer valor que devuelve la función, esto es, la variable x.

Multiples argumentos, multiples salidas

Ecuación horaria

```
function [x,y] = calcular_tiro(t,xo,yo,vo,angulo,g)
  x = xo + vo*cosd(angulo)*t;
  y = yo + vo*sind(angulo)*t - 0.5*g*t.^2;
endfunction
```

Si hace:

```
>> [posx, posy] = calcular_tiro(10,0,0,10,35,9.8)
posx =  81.915
posy = -432.64
```

Ahora sí, se obtienen los dos valores que devuelve la función.

Condicionales

A veces es necesario verificar si se cumplen o no ciertas condiciones. Para ello contamos con el comando `if`. En el codigo anterior, vamos a devolver y=0 si el valor resultante de y es negativo (cosa que no transpase la tierra)

Ecuación horaria más realista

```
function [x,y] = calcular_tiro(t,xo,yo,vo,angulo,g)
  x = xo + vo*cosd(angulo)*t;
  y = yo + vo*sind(angulo)*t - 0.5*g*t.^2;

  if (y < 0)
    y=0;
  endif
endfunction
```

Condicionales

Ecuación horaria más realista

```
function [x,y] = calcular_tiro(t,xo,yo,vo,angulo,g)
  x = xo + vo*cosd(angulo)*t;
  y = yo + vo*sind(angulo)*t - 0.5*g*t.^2;

  if (y < 0)
    y=0;
  endif
endfunction
```

Ahora:

```
>> [posx, posy] = calcular_tiro(10,0,0,10,35,9.8)
posx =  81.915
posy = 0
```

Iterador for

La estructura básica de un for

```
for i=primer_valor:paso:ultimo_valor
    % acá hacen cuentas
endfor
```

Calculemos un promedio de notas

El promedio de un conjunto de muestras $\{x_i\}_{i=1\ldots N}$ se define como:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

```
clc %limpio pantalla
notas = floor(10*rand(1,5)); % notas aleatorias
cantidad=size(notas,2);
total = 0;
for i=1:cantidad
    total+= notas(i);
endfor
promedio = total/cantidad
```

Iterador for

Comentarios del ejemplo anterior:

- Se podría colocar un paso
- La variable `i` se crea para recorrer todos los elementos del vector de datos.
- size(notas,2) equivale a length(notas)

Ejercicio de simulación:

- Armar una función que estime π por Montecarlo. Debe generar N vectores con distribución uniforme [0,1]×[0,1]

$$f_U(x) = \begin{cases} 1 & |x| \leq 1 \\ 0 & |x| > 1 \end{cases} \tag{2}$$

Usar x=rand(N,2) para generarlos todos juntos. Esto me genera pares de vectores en el espacio pedido.

```
> rand(3,2)
    0.231909    0.146372
    0.231932    0.059681
    0.014655    0.389704
```

En este ejemplo tengo 3 pares de vectores.

Ejercicio de simulación:

- Analizar cuáles de estos vectores caen dentro del círculo unitario, usar `norm(x,"rows")<1`.
- Ahora sólo resta contar cuántos vectores cumplen con esto. Usar `nnz(norm(x,"rows")<1)`.
- Finalmente la probabilidad muestral de caer dentro del círculo la calculamos como:
  `P=nnz(norm(x,"rows")<1)/N`
  $P = \frac{\text{Cant muestras de norma < 1}}{\text{Cantidad total de vectores}}$
- Analíticamente, sería: P=Caer en el circulo / P area total,
  $P = (\pi * 1^2/4)/(1) = \pi/4$.
  Por lo cual, estimamos pi=4*P

Ejercicio de simulación:

```
function mypi = estimar_pi (N)
  x=rand(N,2);
  mypi = 4*nnz(norm(x,"rows")<1)/N;
endfunction
```

Funciones anónimas

Son funciones que sirven para declarar funciones localmente, como si fuesen una variable. Pueden ser sobreesritas con facilidad.

La sintáxis es la siguiente:

```
nombre_funcion = @(lista de argumentos) expresión
```

Ejemplo

```
>> potencia = @(vec) sqrt(sum(vec.^2))
```

Eso define una funcion llamada **potencia()** que recibe un vector y devuelve la potencia (o energia) de la señal (matemáticamente, la norma euclídea).

```
>> pot = potencia([1 2 3 2.2 1 2 0 1])
pot =  4.9840
```

Funciones anónimas

A diferencia de las funciones normales, las funciones anónimas pueden ver todas las variables declaradas fuera de ellas a la hora de crearse, pero dichos valores quedan fijos una vez creadas. Es decir:

Ejemplo

```
>> k=1.38064852E-23; q=1.60217662E-19;
>> T=25+273;
>> calcIC = @(vbe, Io) Io * exp( vbe / (k*T/q) );
>> I1 = calcIC(0.6, 13e-12)
I1 =  0.18245 %A T=25ºC
>> T=60+273; %Modifico temperatura
>> I2 = calcIC(0.6, 13e-12)
I2 =  0.18245 %No cambia el resultado
```

Function handlers (a.k.a) Punteros a función

Con el símbolo @ antes del nombre de una función, se obtiene una referencia a ella, y puede ser almacenada en una variable.

```
>> seno = @sin;
```

Más aún, esa nueva variable se comporta como una función nueva.

```
>> seno(pi/4)
```

Function handlers (a.k.a) Punteros a función

Es algo común que las funciones reciban referencias a otras funciones como parámetros. Es el caso de `plotyy`.

`plotyy` permite graficar utilizando distintas funciones de ploteo para cada curva. Ejemplo:

Curva con ejes lineal y logarítmico

```
x = 0:0.1:2*pi;
y1 = sin (x);
y2 = exp (x - 1);
plotyy (x, y1, x - 1, y2, @plot, @semilogy);
```

Gráficos Bi-dimensionales

Ejemplo de gráfico de dispersión

Retomando los datos anteriores:

```
t = [0, 1.0, 1.71, 2.43, 3.14];
y = [1.3, 6.4, 16.1, 31.1, 51.0];
```

Se puede plotear la posición en función del tiempo con el comando:

```
plot(t, y);
```

Ejemplo de grafico de dispersión

Se debe obtener algo así:

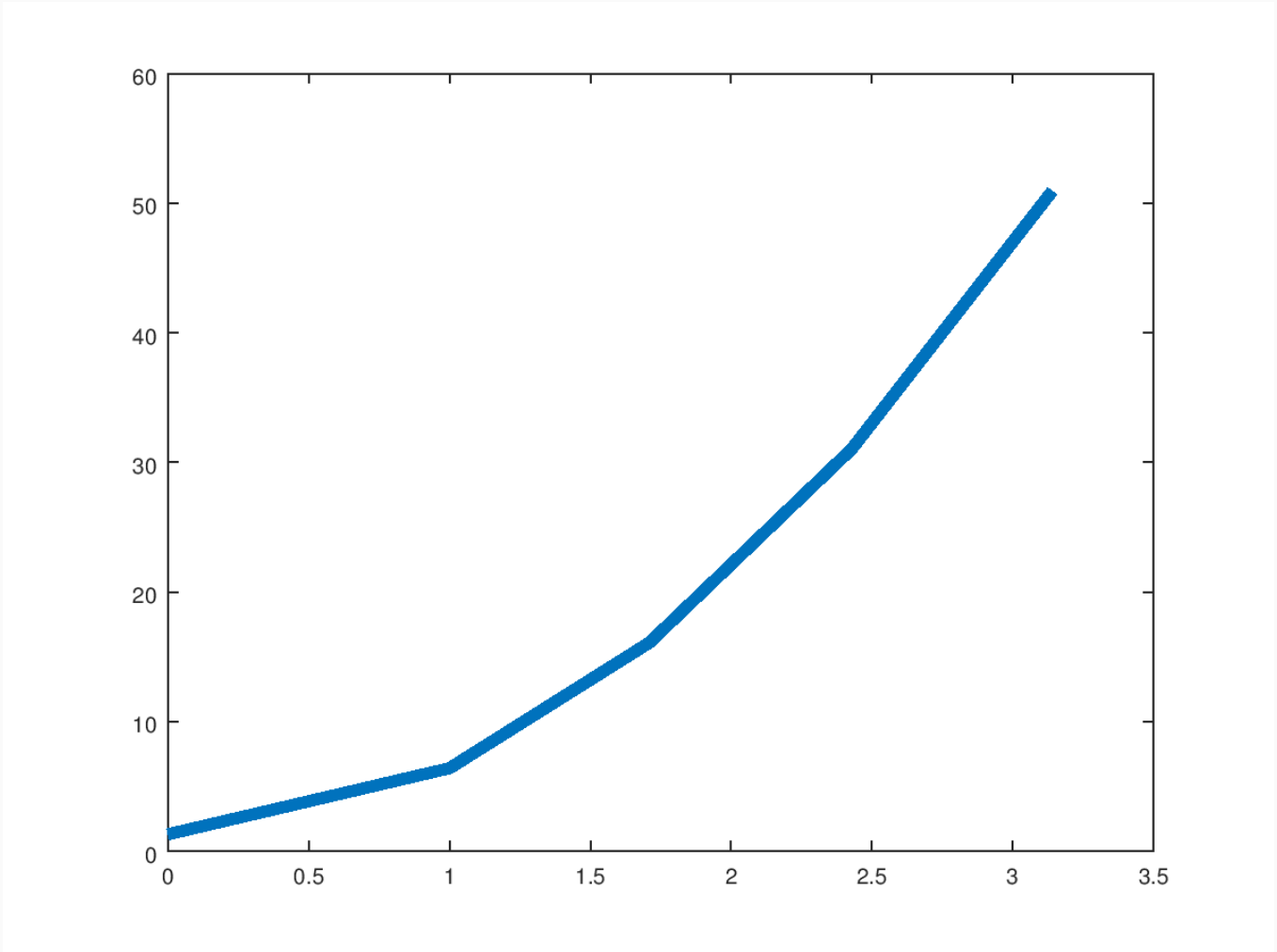


Figura 1: Una parabola media quebrada

Ejemplo de grafico de dispersión

Recordemos que Octave está ploteando una serie de puntos en el plano (x,y). Por defecto, une estos puntos con rectas, pero podemos especificar otro formato.

```
plot(t, y, 'o'); % circulos
plot(t, y, 'rx'); % en rojo, cruces
plot(t, y, 'rx:'); % en rojo, cruces, linea puntada
```

Para ver todos los formatos disponibles ejecutar `help plot`

Gráficos 2D plot()

La función `plot()` se encarga de realizar gráficos en ejes cartesianos. Puede dibujar más de una curva en el mismo gráfico y permite modificar algunas propiedades de diseño como los colores y estilos de las líneas. `plot()` puede tomar una cantidad de argumentos variable e interpreta a los vectores o matrices como datos de las curvas y a las cadenas de texto como las propiedades de diseño. La forma general de los argumentos de `plot()` son de la forma siguiente.

Función plot - argumentos

plot(x1 , y1 , 'propiedad1' , 'valor1' , 'propiedad2' , 'valor2' , x2 , y2 , 'propiedad3' , 'valor3' , ...)

Cada valor hace referencia a la propiedad inmediatamente anterior, y cada propiedad modifica el par de datos x , y anterior.

Gráficos 2D plot()

Hay formas más humanas de escribir el formato:

```
plot (t , y , 'color' , 'r' , 'linestyle', ':')
```

Aquí modificó el color de la línea a rojo y la dibujó punteada. Las propiedades que se pueden modificar para cada curva son: “linestyle”, “linewidth”, “color”, “marker”, “markersize”, “markeredgecolor”, y “markerfacecolor”.

Para más info sobre las propiedades poner `"help plot"` en la consola de Octave.

Gráficos 2D plot()

Por último, y porque sin etiquetas en los ejes y título del gráfico el TP rebota...

```
> plot ( t , x);
> title ( 'Título del gráfico');
> xlabel ( 'Magnitud eje X y [unidades]');
> ylabel ( 'Magnitud eje Y y [unidades]');
> legend( 'Descripción de la curva')
```

Graficando varias curvas juntas

Generalmente es necesario graficar distintas curvas superpuestas para compararlas. Hay muchas formas de hacer esto.

Esta es la más elegante:

```
> plot (t1, x1, t2, x2, t3, x3); % etc
```

Y podemos decorar cada curva de la forma:

```
> plot (t1, x1, 'color', 'r', 'linewidth', 3, ...
        t1, x1, 'color', 'b', 'linestyle', '--', ...
        t3, x3, 'color', 'k', 'linestyle', ':' ); % etc
```

Nota: Con los tres puntos le decimos a Octave que el comando sigue en la próxima linea.

Graficando varias curvas juntas

Otra forma, un poco más ~~enferma~~ rebuscada es pasar todos los datos de las absisas en una matriz, como columas o filas según sean compatibles las dimensiones.

Por ejemplo:

```
> t = linspace(0, 10, 100)';
> x = [t, t.^2/10, t.^3/100, t.^4/1000];
> plot (t, x);
> legend ("Cuadrática", "Cúbica", "Cuarta");
> xlabel ("t");
> ylabel ("t^n / 10^n");
```

Esto puede ser útil cuando la matriz x la generamos automágicamente.

Graficando varias curvas juntas

La forma que se usa el 99 % de las veces es con el comando **hold**.

Con **hold on** le decimos a Octave que grafique una curva sobre la otra, sin borrar la anterior.

```
> t = linspace(0, 10, 100)';
> x = t.^2;
> td = (0:10)';
> xd = td.^2 + 2*randn(length(td), 1); % ruido
> hold on;
> plot (t, x, 'b');
> plot (td, xd, 'or', 'markersize', 7);
> legend ('Modelo', 'Mediciones');
```

Usos y costumbres al plotear

El “estándar de facto” a la hora de hacer curvas es el siguiente:

```
> close all; % cierro todos los plots
> % bla bla bla
> figure; % creo una nueva figura, abre una ventana
> hold on;
> plot (...); % realizo el primer plot
> plot (...); % realizo los siguientes
> legend (...);
> xlabel (...); ylabel (...);
> print ('-dpng' 'miplot'); % Guardo el plot
```

Ejes (semi)logarítmicos

Si es necesario usar una escala logarítmica para alguno de los ejes (o ambos), es exactamente igual que antes, sólo que hay que usar `semilogx()`, `semilogy()` o `loglog()` en lugar de `plot()`.

Bug al usar ejes logarítmicos y hold

En caso de querer usar `hold on` junto con ejes logarítmicos (o semi), primero es necesario plotear la primer curva y *después* usar el comando `hold on`. Caso contrario, `hold on` crea ejes lineales.

Es decir:

```
> figure; % creo una nueva figura, abre una ventana
> semilogx(...); % primer plot
> hold on;
> semilogx(...); % realizo los siguientes
```

Ejes dobles independientes

Si queremos graficar dos curvas con ejes independientes, podemos usar el comando **`plotyy()`**. La sintaxis es:

```
plotyy (x1, y1, x2, y2);
plotyy (x1, y1, x2, y2, @TipoPlot1, @TipoPlot2);
```

En TipoPlot debemos poner alguna de las funciones de ploteo que ya conocimos (u otras). Esto es un ”puntero a función”. Ya lo veremos luego ;) pero básicamente le decimos a Octave con qué función debe graficar cada curva con sus respectivos ejes.

Ejes dobles independientes

Este ejemplo de la documentación de GNU Octave aclara los tantos:

```
> x = 0:0.1:2*pi;
> y1 = sin (x);
> y2 = exp (x - 1);
> ax = plotyy (x, y1, x - 1, y2, @plot, @semilogy);
> xlabel ("X");
> ylabel (ax(1), "Axis 1"); % Especifico el eje a rotula
> ylabel (ax(2), "Axis 2");
```

Más funciones de ploteo

Hay otras funciones, por ejemplo, para hacer histogramas (hist), gráficos discretos (stem), paretos (pareto), graficos polares, etc.

Para más info, escribir: `doc plot` en consola.

Datos en tabla

Una forma conveniente de presentar los puntos a plotear es mediante una tabla, por ejemplo:

```
mediciones = [% tiempo (seg), posición (m)
              0.0      1.30
              1.0      6.40
              1.71    16.1
              2.43    31.1
              3.14    51.0
              ];
```

Aquí se puede plotear de la siguiente forma:

```
plot(mediciones(:,1), mediciones(:,2));
```

Índices y Rangos

Rangos

La funcionalidad de los dos puntos es generar un rango.

Un rango es de la forma: `inicio:paso:fin` O bien: `inicio:fin` entendiéndose el paso igual a 1 en este caso.

Por ejemplo:

```
> 1:5  % no tiene paso, se toma como 1
1  2  3  4  5
> 1:2:10  % se saltea de a 2
1  3  5  7  9
> 15:-1:10  % paso negativo, disminuye en 1 hasta 10
15  14  13  12  11  10
```

Rangos

Los rangos se pueden utilizar para crear vectores. Notemos que podemos usar numeros no enteros:

Por ejemplo:

```
> v = 1:0.1:1.5
1  1.1  1.2  1.3  1.4  1.5

> 2.*v
2  2.2  2.4  2.6  2.8  3

> 10.^v  % Útil para escalas exponenciales
10.0000  12.5893  15.8489  19.9526  25.1189  31.6228
```

Revirtiendo un vector

¿Cómo puedo hacer para “dar vuelta” un vector? Visto lo anterior es muy sencillo:

```
> v = [0.1  0.2  0.3  0.4  0.5];
> v_rev = v(end:-1:1) %end indica el final del vector
0.5  0.4  0.3  0.2  0.1
```

Y puedo obtener solo las coordenadas pares e impares:

```
> x = [1  0  3  0  9  0  -10];
> x_par = x(2:2:end)
1  3  9  -10
> x_impar = x(1:2:end)
0  0  0
```

Para matrices es *exactamente* igual. (Tarea: Jugar con ese caso).

Índices en matrices

Sea A una matriz de $n \times m$. Por ejemplo:

```
> A = [  1  2  3  4
        −1  0  1  0
         3  9  8  −1];
```

Podemos obtener el elemento de la fila `i` y columna `j` de la forma `A(i,j)`

Ejemplo:

```
> A(3,2)
9
```

Los índices empiezan en 1

A diferencia de lenguajes como C o Java, los índices empiezan por 1 y deben ser positivos.

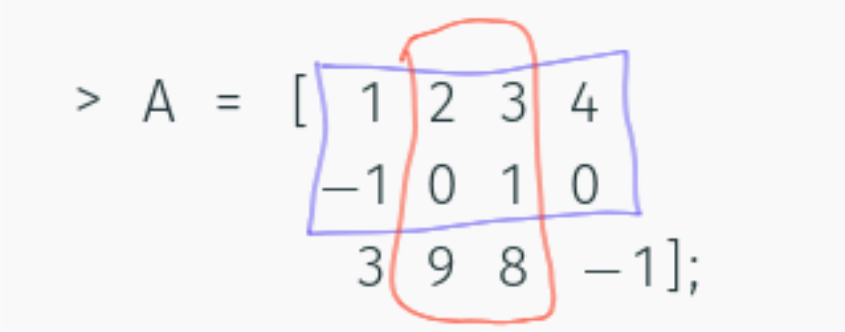
Índices en matrices - submatrices

Ejemplo, sea:

```
> A = [ 1  2  3  4
       −1  0  1  0
        3  9  8  −1];
```

Podemos jugar con lo que aprendimos de rangos:

```
> A(1:2, 2:3)
2   3
0   1
```



Esto indica que obtiene las filas de 1 a 2 y las columnas de 2 a 3.

Índices en matrices - submatrices

Una forma cómoda para especificar que queremos todos los valores de filas o columnas es poner simplemente un `:` en la dimensión que no queremos poner restricciones:

```
> A = [ 1 2 3 4
       -1 0 1 0
        3 9 8 -1];

> A(:,2)
2
0
9

> A(1:2, :)
 1 2 3 4
-1 0 1 0
```

Índices en matrices - end

Muchas veces no tenemos en mente cual es el largo o ancho de una matriz, para ello, podemos valernos de la palabra reservada **end**, que al verla Octave la convierte automáticamente por el tamaño de la dimensión en la cual se use.

```
> A = [ 1 2 3 4
       -1 0 1 0
        3 9 8 -1];
> A(1, end) % primer fila y última columna
4
> A(end-2, end-1) % antepenúltima f. y anteúltima c.
3
> A(:, end-1) % todas las filas y anteultima columna
3
1
8
```

Seguimos indexando

Podemos usar vectores para extraer sólo determinados indices:

```
> A = [ 1 2 3 4
       -1 0 1 0
        3 9 8 -1];

> A([1 3], 1:2)
1 2
3 9
```

Seguimos indexando

Sea

```
> A = [ 1  2  3  4
       −1  0  1  0
        3  9  8  −1];
```

¿Qué hace esto?

```
> A([1  1  2  1], :)
```

Seguimos indexando

Sea

```
> A = [ 1  2  3  4
       −1  0  1  0
        3  9  8  −1];
```

¿Qué hace esto?

```
> A([1  1  2  1], :)   % Selecciono las columnas
   1   2   3   4
   1   2   3   4
  −1   0   1   0
   1   2   3   4
```

Técnicas avanzadas de indexado[4]

En lugar de un rango, se puede poner una condición. Por ejemplo:

```
> v = [8.1  9.0  1.2  9.1  6.3  0.9  2.7];

> v(v>=4)
8.1  9.0  9.1  6.3

> v(v.^2 < 2*v)
1.2  0.9

%puedo aplicarle una función por fuera
%para saber cuántos cumplen
> length (v (v >=4))
ans =  4
```

[4]Ver: https://www.gnu.org/software/octave/doc/interpreter/Advanced-Indexing.html

Técnicas avanzadas de indexado

Si disponemos de dos vectores del mismo largo, podemos usar uno como condición del otro. Ejemplo:

```
> t = 0:0.1:10; % Un par de datos, cualquier cosa.
> x = 10 - 3.*t; % x(t)

> x(t>9.5) % x(t) dado que t>9.5
-18.8   -19.1   -19.4   -19.7   -20.0
```

Nota: ¿ese ”dado qué”no les suena de algún lado? ¿Para qué suponen que se usa algo así?

Técnicas avanzadas de indexado: operadores lógicos

`&` Operador “y” lógico (*and*)

`|` Operador “o” lógico (*or*)

`~` Operador “no” lógico (*not*)[5]

```
> 1 | 0
ans = 1
> 1 & 0
ans = 0
> ~1
ans = 0 % variable tipo lógica
> t=1:2:9
    1     3     5     7     9
> t(t>3 & t<9)
    5     7
```

[5] Los operadores && y || operan sobre escalares y devuelven escalares

Técnicas avanzadas de indexado

¿Y si quiero saber qué índices de un vector **v** son los que cumplen que $a < v < b$?

Para eso existe el comando `find()`.

```
> notas = 2:1:10;
 2 3 4 5 6 7 8 9 10

> indices = find ( notas>=4 & notas<=7)
3   4   5   6
```

Técnicas avanzadas de indexado

Si bien estas dos expresiones hacen lo mismo, la segunda es mucho más recomendable (menos costosa computacionalmente):

```
> x( find ( t>2.2 & t<3) )   % No

> x(   t>2.2 & t<3 )         % Sí
```

< Mencionar usos de los índices en la práctica >

Ejercicio

Aprobados en Álgebra 2

Construya una función que reciba un archivo por parámetro y devuelva la cantidad de aprobados. El prototipo es el siguiente:

```
function [aprobados,n_alumnos]=aprobar(archivo_notas)
 // cosas
endfunction
```

Luego armar un script **aprobados_algebra_ultima_fecha** donde utilice esta función. La salida debe verse como:

```
> CANTIDAD DE GENTE QUE SE PRESENTÓ AL EXAMEN: ??
> CANTIDAD DE GENTE QUE APROBÓ EL EXAMEN: ??
> PORCENTAJE DE APROBADOS:  ??.?? %
```

Graficos de superficies

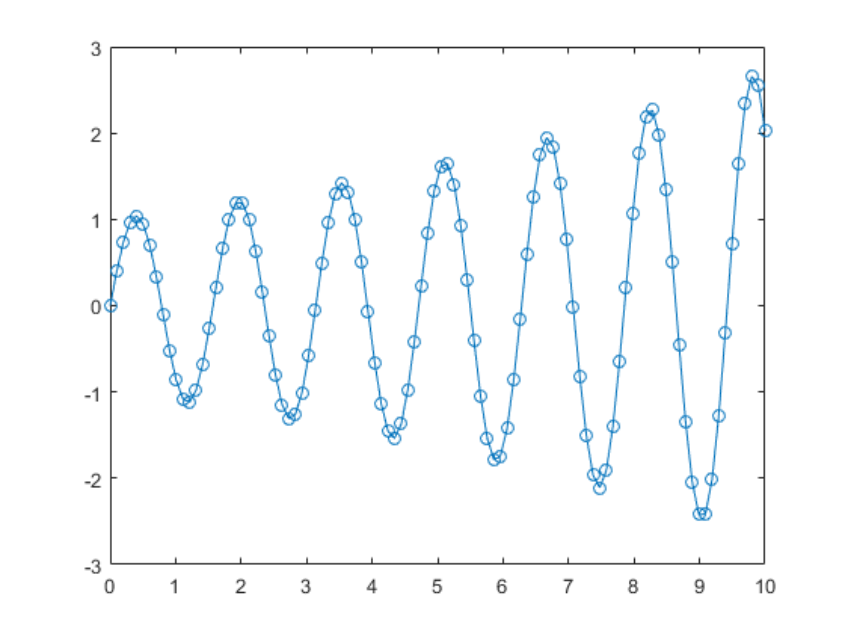
Superficies

A la hora de graficar una función de dos variables, necesitamos posicionarnos en un espacio tridimensional.

El dominio lo tomamos sobre el plano xy, y la grafica de la función sobre el eje z.

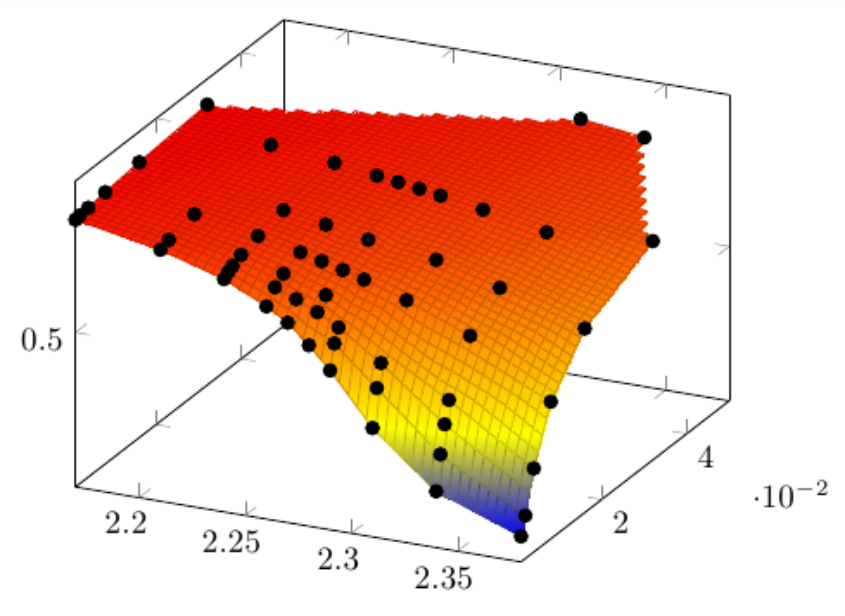
Superficies

Para un gráfico de una función, necesitamos discretizar el dominio, generalmente con linspace. Octave se encarga de llenar con lineas los valores intermedios para representar el plot.



Superficies

En el caso de un plot en dos dimensiones, necesitamos una cuadrícula (malla o mesh), sobre la cual evaluar la función:



Superficies

Para crear el dominio de nuestro campo escalar, debemos hacer el producto cartesiano entre los valores de x e y. Para ello, usamos meshgrid:

```
x = linspace(-1,1,100);
y = linspace(-2,2,500);

[xx,yy] = meshgrid(x,y);

surf(xx,yy, (xx.^2)+(yy.^2)/2)
```

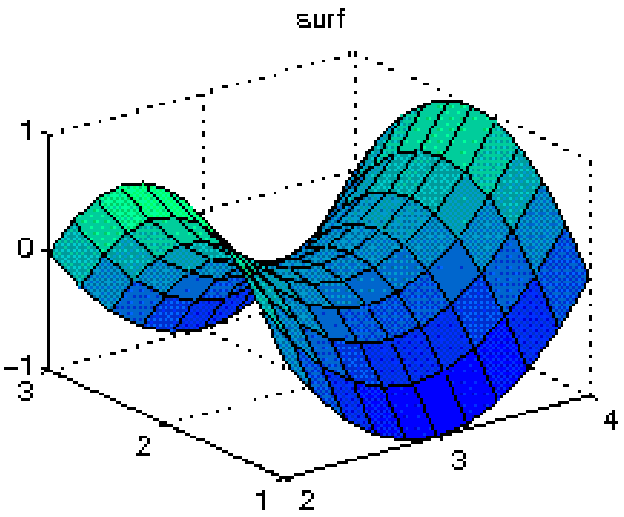
Superficies

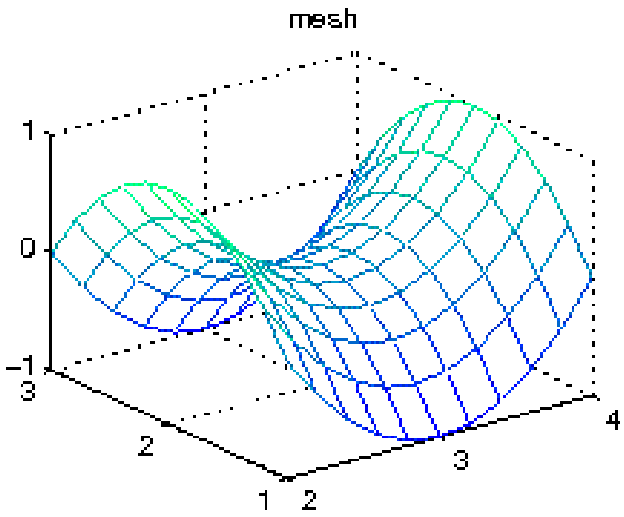
Cabe destacar que los xx e yy que devuelve meshgrid son matrices (por comodidad al representarlos).

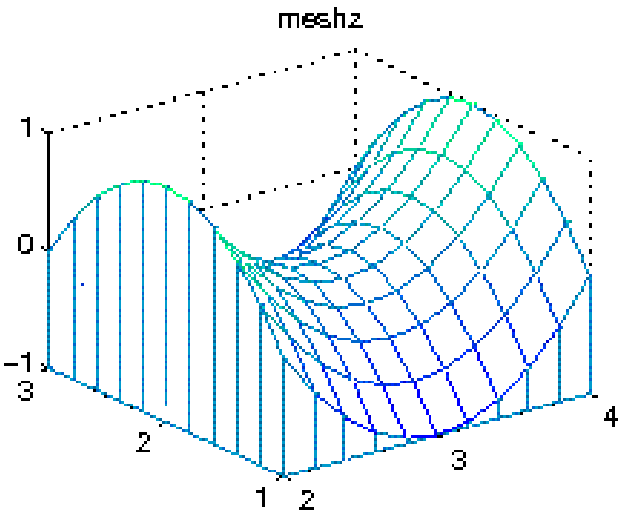
Sin embargo, muchas funciones no se portan bien al recibir como parametro una matriz, por lo cual, podemos vectorizar el resultado:

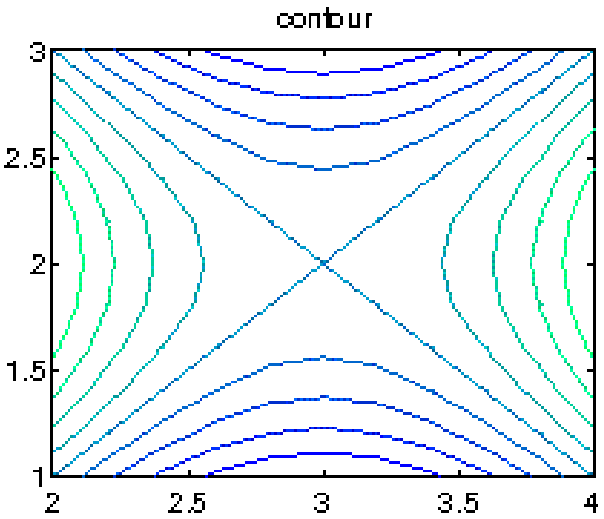
```
x = linspace(-1,1,100);
y = linspace(-5,5,500);

[xx,yy] = meshgrid(x,y);

xxv = xx(:);
yyv = yy(:);

surf(xxv,yyv, alguna_f(xxv,yyv))
```

Hay varias funciones para plotear

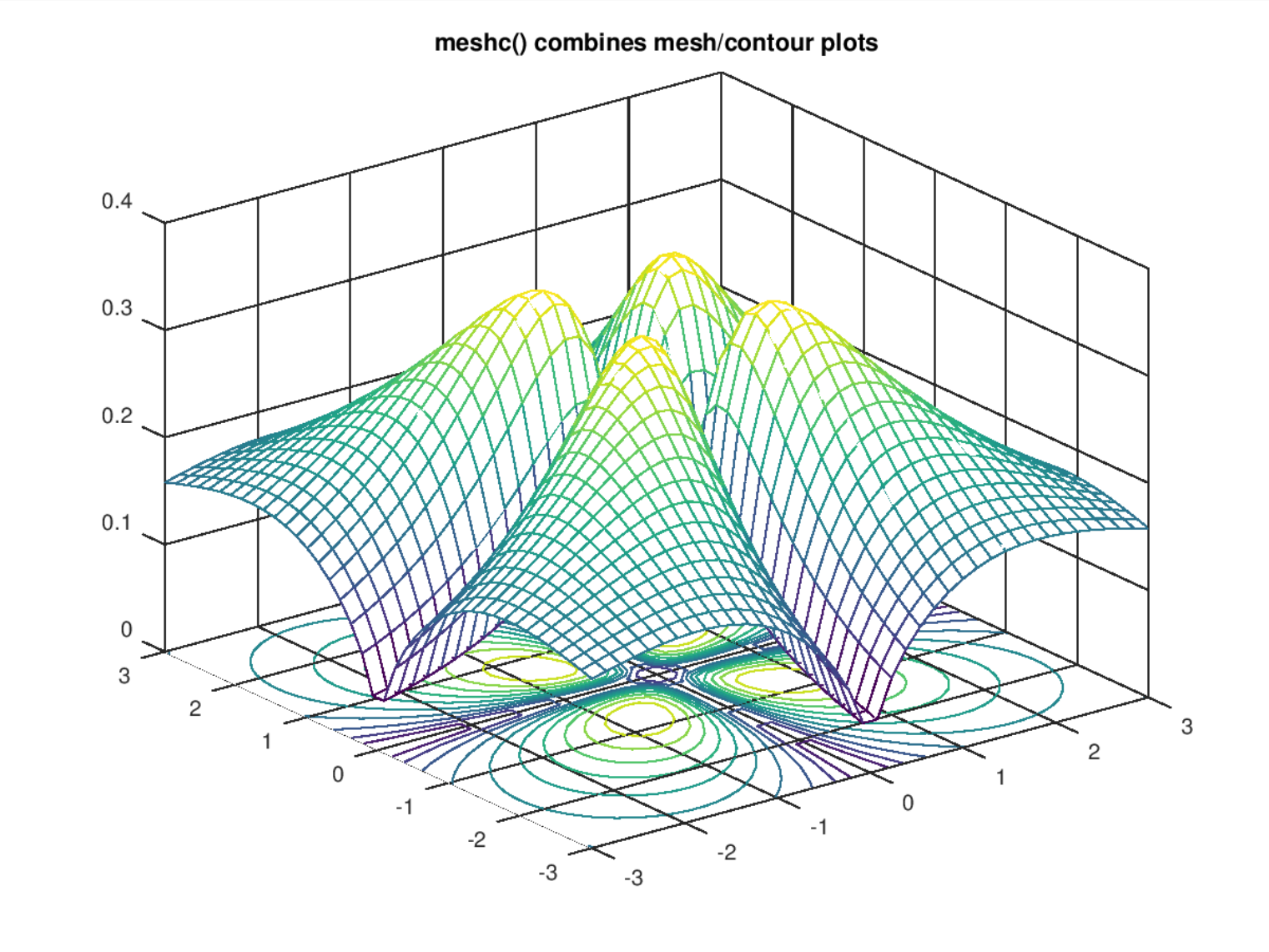




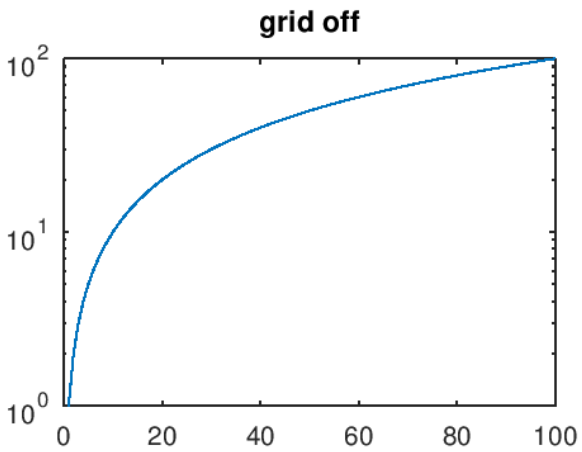


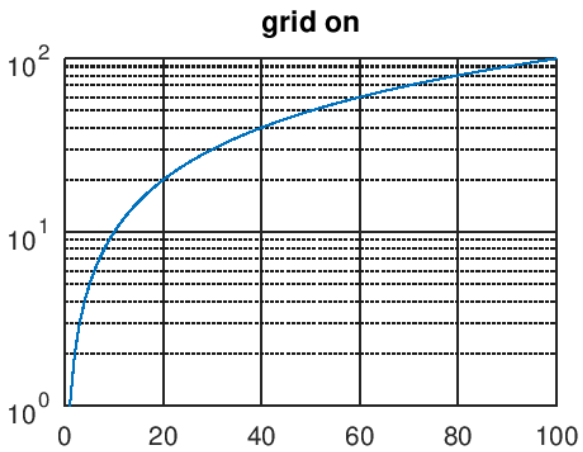


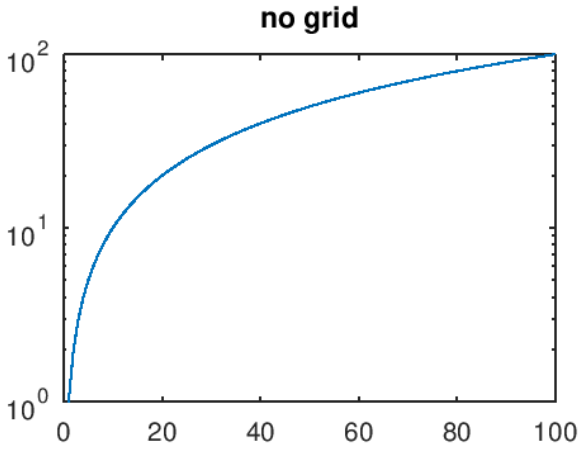
Hay varias funciones para plotear

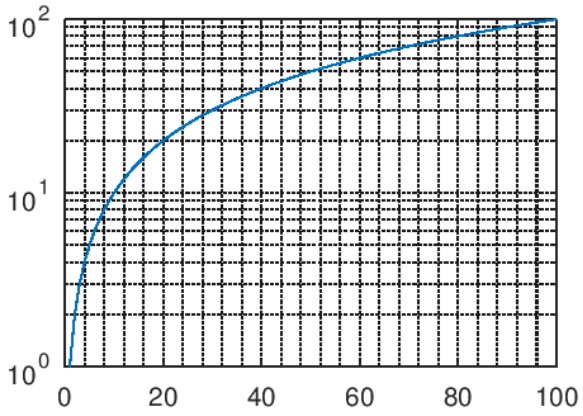


Notas sobre grid

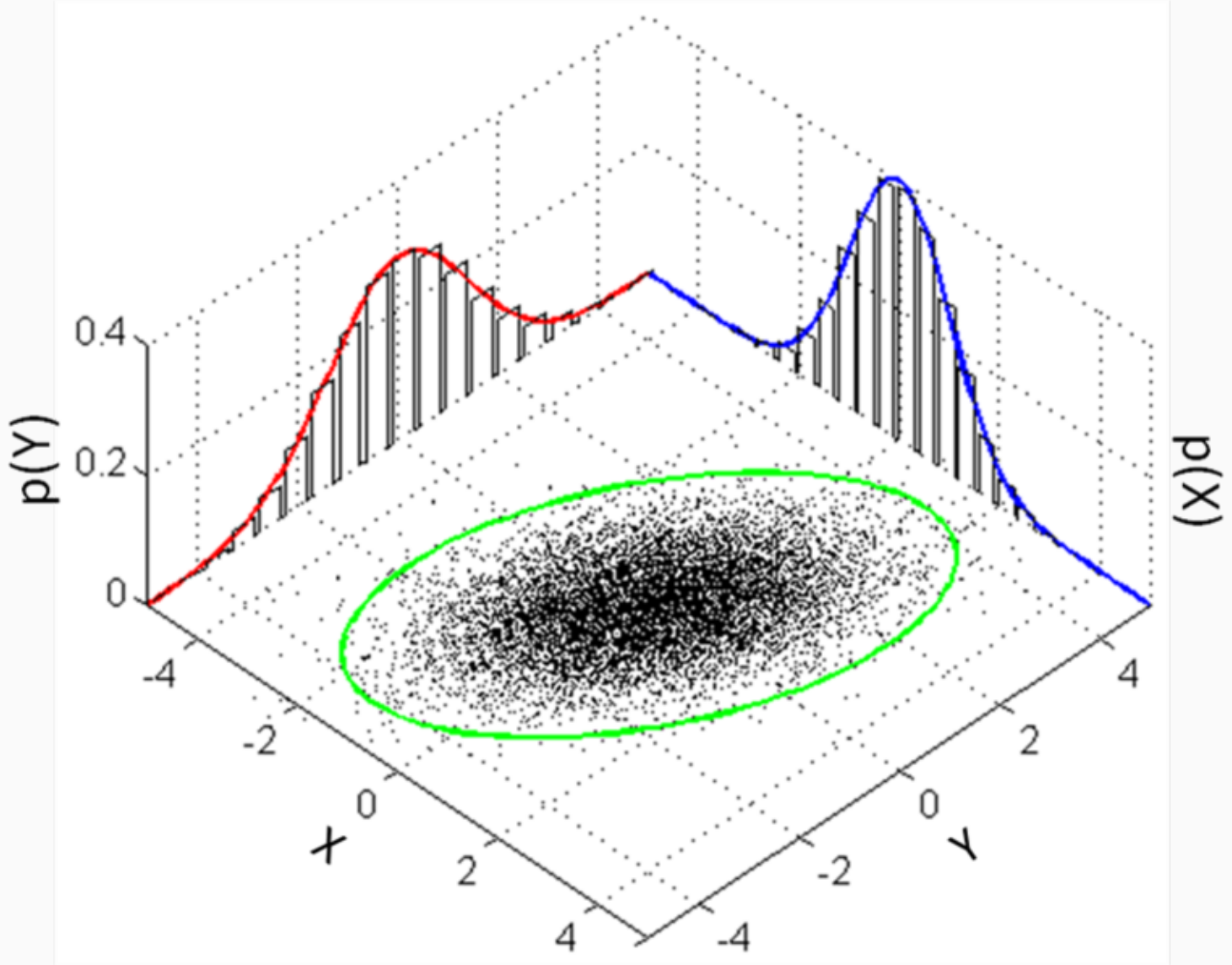








Tarea: Mezcla de normales bivariadas



Ecuaciones diferenciales

Ecuación Diferencial Ordinaria

Una Ecuación Diferencial Ordinaria Lineal tiene la forma:

$$a_0(t)x + a_1(t)\frac{dx}{dt} + a_2(t)\frac{d^2x}{dt^2} + \ldots + a_n(t)\frac{d^nx}{dt^n} + b(t) = 0 \tag{3}$$

O con otras notaciones:

$$a_0(t)x + a_1(t)x' + a_2(t)x'' + \ldots + a_n(t)x^{(n)} + b(t) = 0 \tag{4}$$

$$a_0(t)x + a_1(t)\dot{x} + a_2(t)\ddot{x} + \ldots + a_n(t)\overset{(n)}{\dot{x}} + b(t) = 0 \tag{5}$$

Donde $x = x(t)$

Una de las ventajas del cálculo numérico es poder resolver cómodamente ecuaciones diferenciales a coeficientes no constantes.

Ecuación de primer orden

Una Ecuación Diferencial Ordinaria Lineal de primer orden se escribe algebráicamente como:

$$\frac{dx}{dt} = f(t, x)$$

Donde toda la información sobre la ecuación diferencial está contenida en $f(t, x)$. Por lo tanto, a la hora de escribir una ED en Octave, la vamos a expresar a través de f.

Para resolver esta ecuación diferencial, se necesita definir una condición inicial: $x(0) = x_0$

Forma Algebraica ED 1 orden

Por ejemplo, un tanque cilíndrico de sección S que tiene una abertura en su parte inferior de sección s se va vaciando. Usando la Ecuación de Torricelli para modelar la altura del nivel de agua, se llega a la expresión:

$$h'S = -s\sqrt{2gh}$$

Esta ecuación se puede llevar a la forma algebráica:

$$h' = -k\sqrt{2gh}$$

Donde $k = s/S$ es el cociente entre la sección de la abertura y la sección del tanque

Forma Algebraica ED 1 orden

A partir de:

$$h' = -k\sqrt{2gh}$$

Definimos la ecuación diferencial de 1º orden como:

$$f(t, h) = -k\sqrt{2gh}$$

Por lo tanto, en Octave definimos la función f para valores particulares de k y g como:

```
g=10;  k=0.1
f  = @(t,h) -k*sqrt(2*g*h)
```

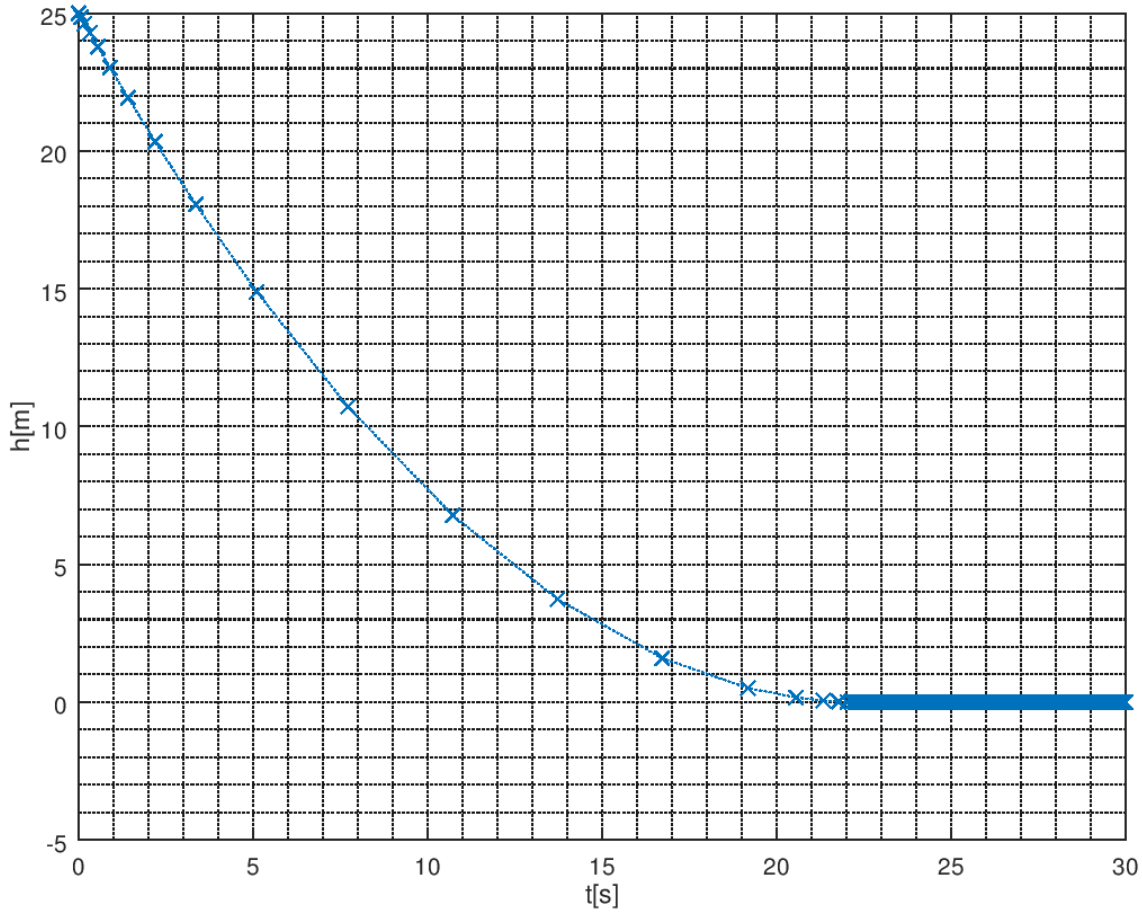
Forma Algebraica ED 1 orden

Para resolver la ecuación diferencial, vamos a usar el método **ode45**, que funciona bien cuando las ecuaciones diferenciales no son rígidas[6]:

```
g=10; k=0.1; h0=25; tfinal = 25;
f = @(t,h) -k*sqrt(2*g*h);

[t,h] = ode45(f, [0, tfinal], h0);
plot(t,h);
```

[6] Un sistema rígido es uno que tiene componentes que cambian rápidamente (transitorios), junto con componentes de cambio lento.

Problemas con ode45 (I)



Notar la gran cantidad de puntos en la parte recta del grafico ($h \simeq 0$).

Problemas con ode45 (I)

Al analizar los valores de h calculados se ve algo interesante:

```
>> h
%[...]
   10.71588 +  0.00000i
    6.77401 +  0.00000i
    3.73213 +  0.00000i
%[...]
    0.00003 +  0.00000i
    0.00001 -  0.00000i
    0.00000 +  0.00001i  % altura imaginaria
%[...]
   -0.00135 +  0.00001i  % altura negativa
   -0.00135 -  0.00001i
```

Problemas con ode45 (I)

El problema es que Octave no sabe que la altura debe ser un número positivo, y sobre todo, real.

Octave estima la derivada de h cerca de 0 como $-k\sqrt{2gh}$. El problema es que si prueba poniendo un h negativo, ¡su derivada es imaginaria!

Obviamente, por esta razón Octave sufre mucho, y para que las cosas le cierren tiene que tomar muchos muchos puntos, a fin de que no le diverga todo al frutear con complejos.

Notar cómo un caso bastante trivial se puede complicar por razones ajenas a la programación.

Solución al Problema (I)

Una vez encontrado el problema, la solución es bastante sencilla.

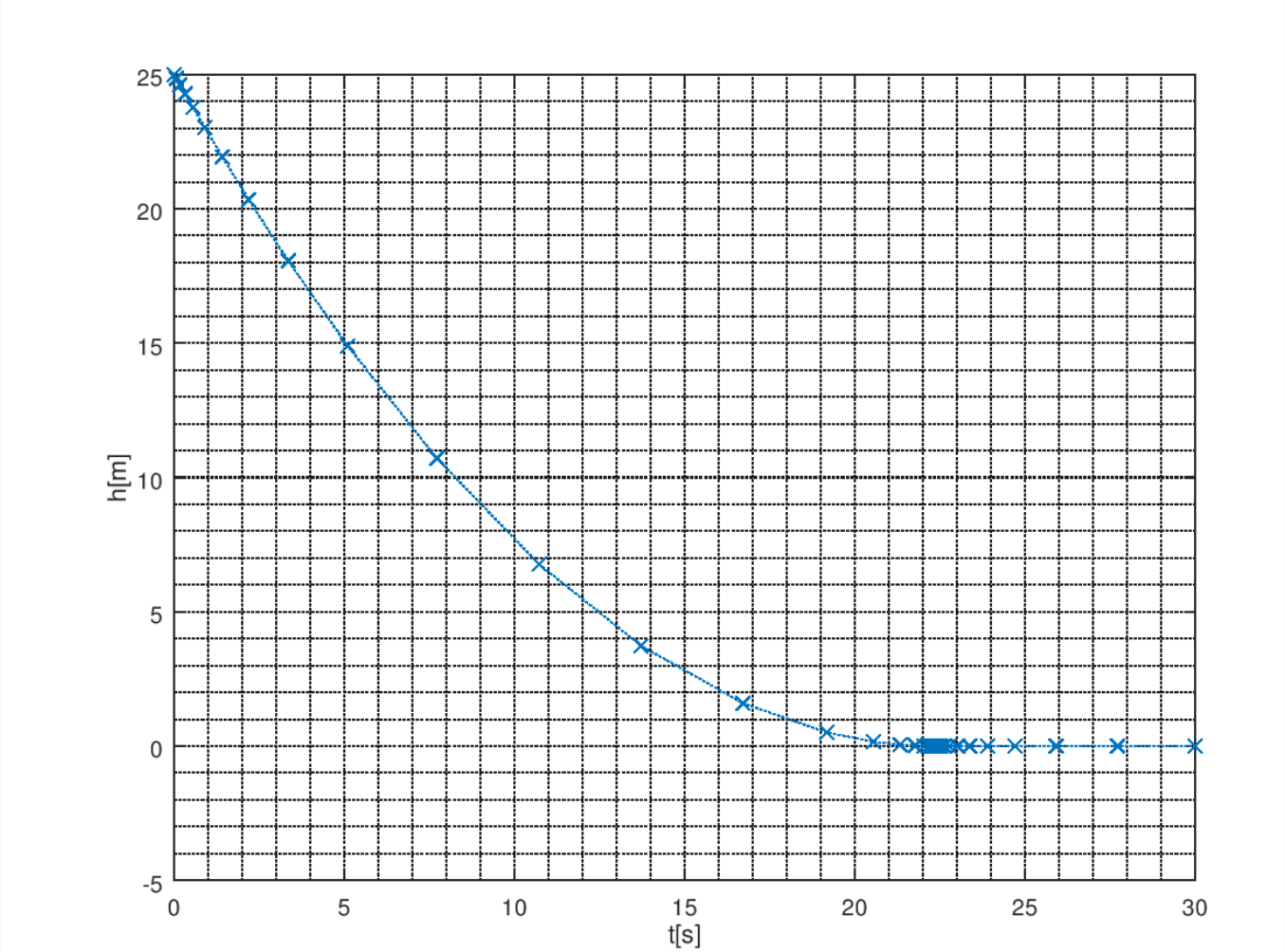
Por ejemplo, se puede redefinir f:

$$f(t,h) = \begin{cases} -k\sqrt{2gh} & , h > 0 \\ 0 & , h \leq 0 \end{cases}$$

Para ello, podemos utilizar la operación lógica `(h>0)`

```
g=10; k=0.1; h0=25; tfinal = 25;
f = @(t,h) -k*sqrt(2*g*h.*(h>0));

[t,h] = ode45(f, [0, tfinal], h0);
plot(t,h);
```

Problemas con ode45 (I)



¡Mucho mejor!

Sistema de Ecuaciones Diferenciales Ordinarias

Un SEDO Lineal es básicamente lo mismo que una variable sola, pero ahora vamos a considerar simplemente que $\mathbf{x}$ es un vector de $\mathbb{C}^n$,

Vamos a describir el sistema algebráicamente como[7]:

$$\frac{d\mathbf{x}}{dt} = f(t, \mathbf{x})$$

Y dado que para Octave un escalar, un vector, o una matriz es lo mismo, podemos hacer exactamente lo que hacíamos antes.

[7] Recordar que los vectores y matrices se derivan igual que los reales, solo que componente a componente.

Ejemplo de SEDO

El modelo SIR ha sido propuesto para estudiar la evolución de una enfermedad en una población cerrada. Este modelo permite calcular, a través de simulaciones numéricas, el número de personas infectadas por una enfermedad contagiosa a lo largo del tiempo. El nombre del modelo deriva del hecho que se utilizan las siguientes variables temporales:

- $S(t)$: Número de personas susceptibles en el instante t
- $I(t)$: Número de personas infectadas en el instante t
- $R(t)$: Número de personas recuperadas en el instante t

Ejemplo de SEDO

El modelo SIR más popular es el de Kermack-McKendrick, el cual fué propuesto para explicar el rápido crecimiento y caída en el número de personas infectadas observadas durante las epidemias de la peste bubónica en Londres, 1665–1666, o la epidemia de cólera en Londres de 1865.

Este modelo asume que la población es fija, es decir que no hay nacimientos ni muertes durante el período simulado. Ademas asume que el período de incubación es instantáneo.

Ejemplo de SEDO

Este modelo simplificado, pero efectivo, consta del siguiente sistema de tres ecuaciones diferenciales, donde β es la velocidad de infección y γ es la velocidad de recuperación:

Llamando N=S+I+R por brevedad:

$$\begin{cases} S'(t) &= -\beta S(t) I(t)/N \\ I'(t) &= \beta S(t) I(t)/N - \gamma I(t) \\ R'(t) &= \gamma I(t) \end{cases}$$

Lo cual vectorialmente queda:

$$\mathbf{x} = \begin{bmatrix} S(t) \\ I(t) \\ R(t) \end{bmatrix}, \qquad \frac{d\mathbf{x}}{dt} = f(t, \mathbf{x}) = \begin{bmatrix} -\beta S(t) I(t)/N \\ \beta S(t) I(t)/N - \gamma I(t) \\ \gamma I(t) \end{bmatrix}$$

Ejemplo de SEDO

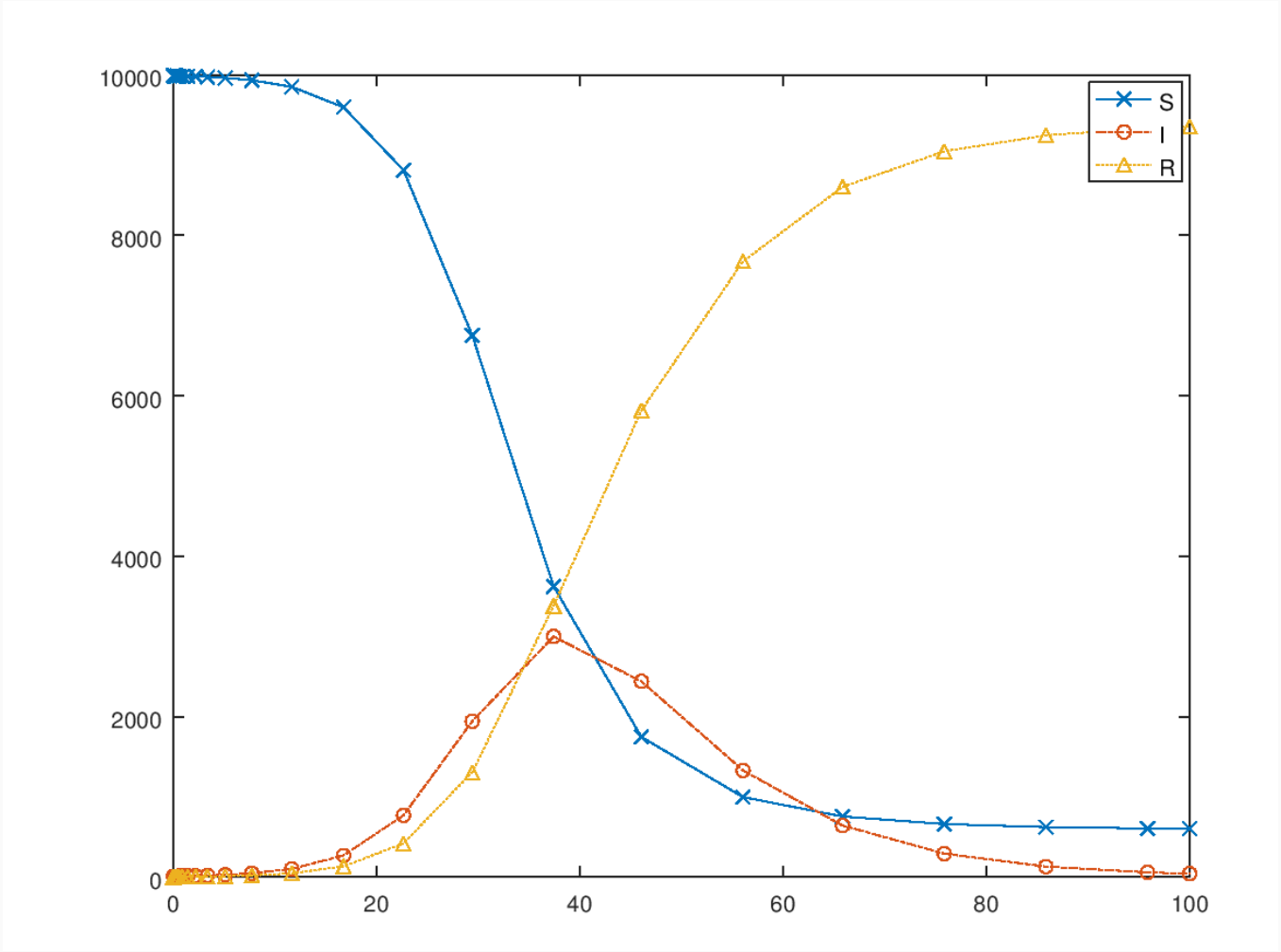
Dado:

$$\mathbf{x} = \begin{bmatrix} S(t) \\ I(t) \\ R(t) \end{bmatrix}, \qquad \frac{d\mathbf{x}}{dt} = f(t, \mathbf{x}) = \begin{bmatrix} -\beta S(t) I(t)/N \\ \beta S(t) I(t)/N - \gamma I(t) \\ \gamma I(t) \end{bmatrix}$$

Vamos a analizar la evolución de S,I,R para ciertos parámetros β, γ y una población dada. Ayuda de código:

```
b = .3; g=.1; tfinal=100; xo=[9990 10 0];
% x=[S,I,R] <- los uso a lo largo asi queda como tabla
f = @(t,x) [ -b*x(1).*x(2)/(x(1)+x(2)+x(3)),...
              b*x(1)*x(2)/(x(1)+x(2)+x(3))-g*x(2),...
              g*x(2)];
[t,x] = ode45(f, [0, tfinal], xo);
```

Modelo SIR



Está bien, aunque se ve medio “quebrado” ¿no?

Configuración de ode45

La mayoría de las funciones iterativas en Octave, tienen alguna configuración de la condición de corte. Por ejemplo, al llegar a una cota del error.

Para configurar **ode45** se utiliza la función **odeset**.

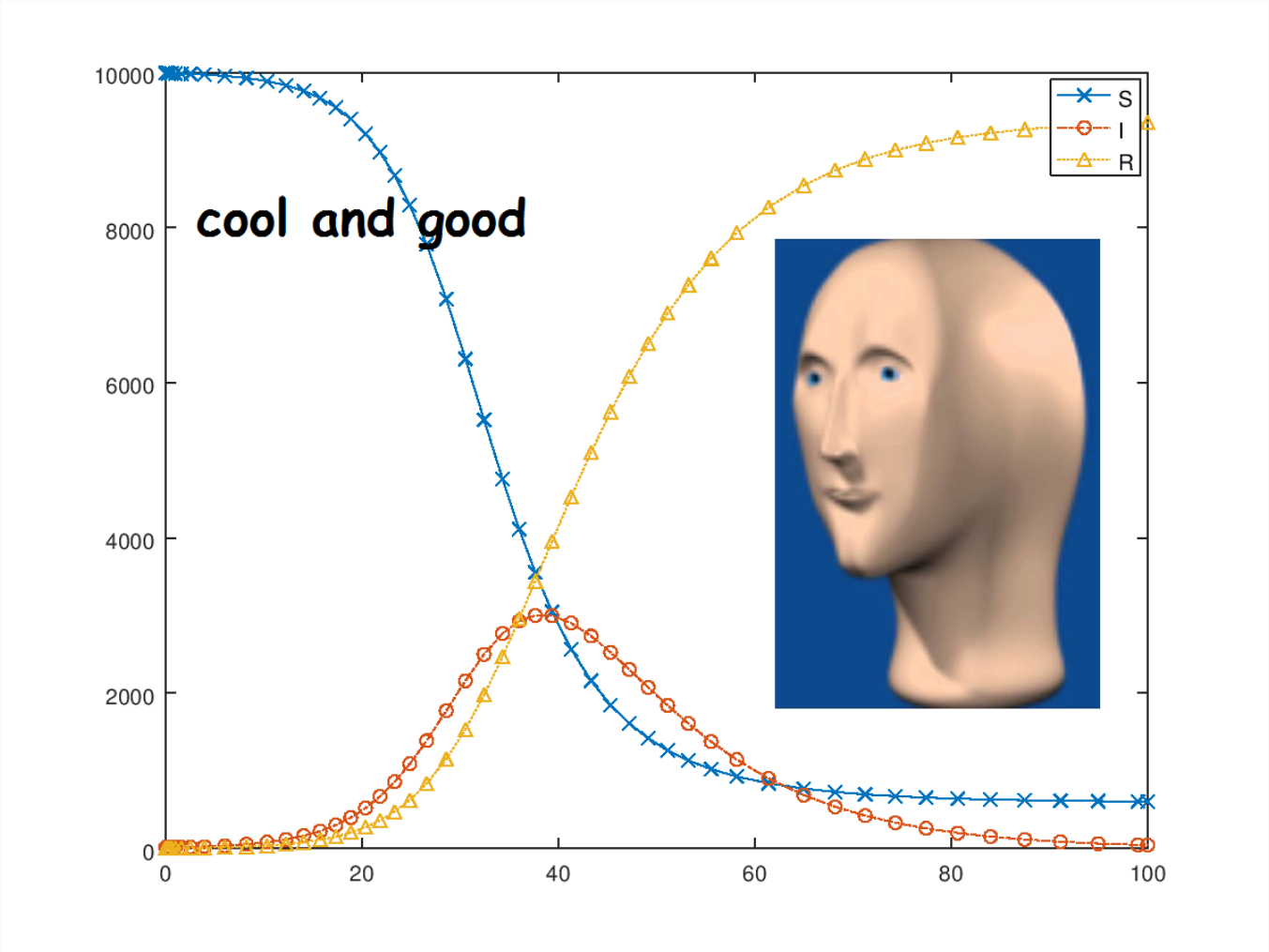
En el codigo anterior, modificar esta parte:

```
odeopt = odeset ("AbsTol", 1e-3, "RelTol", 1e-9);
[t,x] = ode45(f, [0, tfinal], xo, odeopt);
```

Básicamente, se está configurando la cota para el error absoluto y para el error relativo[8]. Si bien Octave suele elegir estos valores por nosotros, no siempre sabe lo que está haciendo.

[8] Ejecutar doc odeset para más configuraciones.

Modelo SIR



(Al hacer el plot el meme puede no aparecer)

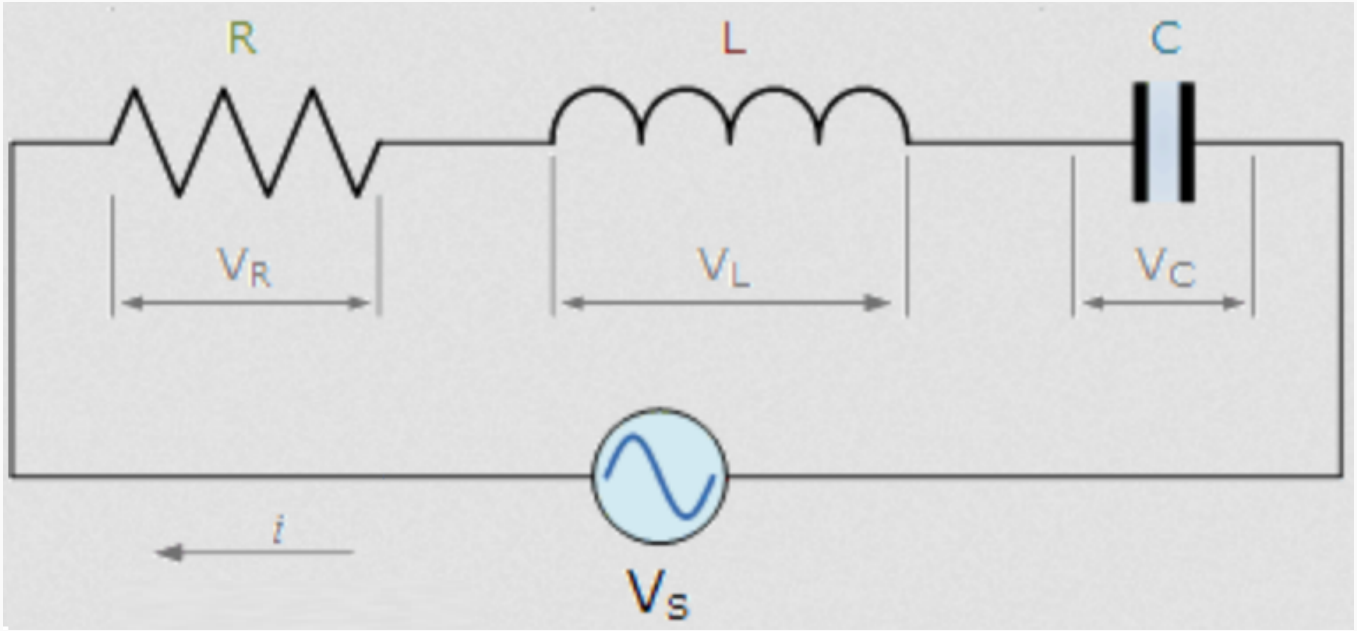
Ecuaciones de orden superior

Hay un teorema[9] que dice que cualquier ecuación diferencial ordinaria lineal de orden n, se puede escribir como n ecuaciones diferenciales ordinarias lineales de primer orden.

[9] Esto es mucho muy importante.

Ejemplo: RLC

Planteando las ecuaciones de Kirchoff a el circuito RLC tenemos:



Ejemplo: RLC

Malla:

$$V_R + V_L + V_C = V_S \tag{6}$$

Componentes:

$$\begin{cases} V_R &= i\,R \\ V_L &= L\,i' \\ i &= R\,V_C' \end{cases} \tag{7}$$

Reemplazando (2) en (1):

$$C\,V_C' + LCV_C'' + V_C = V_S$$

Llegamos a una típica ecuación difrencial de segundo orden. Una vez que podamos resolverla y hallar $V_C(t)$ se podrán despejar el resto de las ecuaciones según (2).

Ejemplo: RLC

Si pudiese armar un vector $\mathbf{x}$ que contenga las tensiones y/o corrientes del circuito, de tal forma que pudiese expresar el problema como $d\mathbf{x}/dt = f(t, \mathbf{x})$, podría resolverlo sin problemas.

Un criterio conveniente, es tomar las magnitudes de los elementos que acumulan energía, en este caso, la tensión del capacitor (V_R) y la corriente del inductor (i, porque es un circuito serie):

$$\mathbf{x} = \begin{bmatrix} i \\ V_C \end{bmatrix}$$

Y ahora, voy a tratar de definir todo en función de ese vector, y de su derivada $d\mathbf{x}/dt = \begin{bmatrix} i' \\ V_C' \end{bmatrix}$.

Ejemplo: RLC

Quiero llegar a escribir $d\mathbf{x}/dt = f(t, \mathbf{x})$, es decir:

$$\begin{bmatrix} i' \\ V_C' \end{bmatrix} = f\left(t, \begin{bmatrix} i' \\ V_C' \end{bmatrix}\right)$$

Partiendo de la ecuación de la malla: $V_R + V_L + V_C = V_S$, la dejo en función de i, i', V_C, V_C':

$$i\,R + L\,i' + V_C = V_S$$

De donde despejo i':

$$i' = (-V_C - i\,R + V_S)/L$$

Y de la ecuación del capacitor $i = C\,V_C'$ despejo:

$$V_C' = i/C$$

Ejemplo: RLC

Finalmente, podemos decir:

$$\begin{bmatrix} i' \\ V_C' \end{bmatrix} = \begin{bmatrix} \frac{-R}{L} i - \frac{1}{L} V_C + \frac{1}{L} V_S \\ \frac{1}{C} i \end{bmatrix} \tag{8}$$

Que incluso, se puede escribir matricialmente a lo Álgebra II[10] como:

$$\begin{bmatrix} i' \\ V_C' \end{bmatrix} = \begin{bmatrix} -R/L & -1/L \\ 1/C & 0 \end{bmatrix} \begin{bmatrix} i' \\ V_C' \end{bmatrix} + \frac{1}{L} \begin{bmatrix} V_S \\ 0 \end{bmatrix}$$

No siempre se puede llevar a esta forma matricial, acá porque eran ecuaciones lineales. De todas formas la $d\mathbf{x}/dt = f(t, \mathbf{x})$ contempla también los casos no lineales.

[10] analizar autovalores

Ejemplo: RLC

En fin, la expresion:

$$\begin{bmatrix} i' \\ V_C' \end{bmatrix} = \begin{bmatrix} \frac{-R}{L} i - \frac{1}{L} V_C + \frac{1}{L} V_S \\ \frac{1}{C} i \end{bmatrix} \tag{9}$$

La podemos escribir como:

```
R=100; C=22e-6; L=.5;
VS = @(t) 5*(t>1); % Escalon de tension
f = @(t,x) [ -R/L*x(1)-1/L*x(2)+1/L*VS(t),...
                1/C*x(1) ]; % ojo con los espacios
```

Y meter a **ode45** u **ode23**.

Ejemplo: RLC

Nota: En el sistema

$$\begin{bmatrix} i' \\ V_C' \end{bmatrix} = \begin{bmatrix} -R/L & -1/L \\ 1/C & 0 \end{bmatrix} \begin{bmatrix} i' \\ V_C' \end{bmatrix} + \frac{1}{L} \begin{bmatrix} V_S \\ 0 \end{bmatrix}$$

Es importante recordar que las soluciones son de la forma $C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}$, donde λ_1, λ_2 son los autovalores de A.

Está bueno ver cómo se van moviendo los autovalores de A en el plano complejo al variar R y LC. Recordar que la parte imaginaria de los autovalores da la frecuencia de oscilación frente al escalón, y la parte real da el tiempo que dura el transitorio. Plotear la respuesta temporal y comparar con la posición de los aves.

Ajuste de Curvas

Ajuste lineal

Datos a ajustar: Tengo un monton de puntos (t_i, x_i).

Y tengo un par de funciones li: $f_a(t)$, $f_b(t)$, etc.

Quiero hallar los mejores coeficientes de la combinación lineal:

$$f(t) = \alpha f_a(t) + \beta f_b(t) ... etc$$

Tal que:

$$\sum_{i=1}^{n} [x_i - f(t_i)]^2$$

sea mínima. La clave acá, es que son los coeficientes de una combinación lineal.

Ajuste lineal

tl;dr, cosas de álgebra.

Evaluo la combinación lineal en los puntos (t_i, x_i):

$$\begin{aligned} x_1 &= \alpha f_a(t_1) + \beta f_b(t_1) \\ x_2 &= \alpha f_a(t_2) + \beta f_b(t_2) \\ &\vdots \\ x_n &= \alpha f_a(t_n) + \beta f_b(t_n) \end{aligned}$$

Y escribo el sistema matricialmente como $\mathbf{y} = A\mathbf{x}$:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} f_a(t_1) & f_b(t_1) \\ f_a(t_2) & f_b(t_2) \\ \vdots & \vdots \\ f_a(t_n) & f_b(t_n) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Ajuste lineal

Y escribo el sistema matricialmente como $\mathbf{y} = A\mathbf{x}$:

$$\mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad A = \begin{bmatrix} f_a(t_1) & f_b(t_1) \\ f_a(t_2) & f_b(t_2) \\ \vdots & \vdots \\ f_a(t_n) & f_b(t_n) \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Y como el sistema está sobredeterminado, esto es, es incompatible, no hay solución.

Peeero, puedo hallar un $\widetilde{\mathbf{x}}$ tal que $|\mathbf{y} - A\widetilde{\mathbf{x}}|$ sea la mínima posible para todo $\mathbf{x} \in \mathbb{R}^2$.

$$\text{tl;dr:} \quad \widetilde{\mathbf{x}} = A^{\#}\mathbf{y}$$

Ajuste lineal

La pseudoinversa de Moore-Penrose (la $A^{\#}$) se puede hallar con el comando *pinv*: `xhat=pinv(A)*y`

Pero, como es una tarea muy común, hay un operador para hacer esa cuenta:

```
xhat = A\y;
```

Teniendo en vista que $\widetilde{\mathbf{x}} = A^{\#}\mathbf{y}$, dado que es necesario premultiplicar por la inversa, si ningún matemático no está mirando, podemos pensar que es como decir: $\widetilde{\mathbf{x}} = \frac{1}{A}\mathbf{y} = A \setminus \mathbf{y}$

Esto lo digo, porque si las dimensiones dan, al producto $\mathbf{y}A^{\#}$ se lo escribe como `y/A` (que es el lío que pasa al dividir sin `./` por error).

Ajuste lineal

Tarea: Inventar una tabla de datos de tension–deformación para un resorte ideal de $k = 40N/m$. Plotear los puntos con circulos en un gráfico de deformación en función de la tensión.

Ahora, armar un vector aparte, sumando a los valores de deformación un numero aleatorio generado con `randn`, con alguna constante conveniente.

Ahora, con los puntos movidos, ajustar por cuadrados mínimos para recuperar k. Plotear la recta para el k hallado.

Graficar el ECM en función de la varianza del ruido (la constante que multiplica a randn).

Ajuste NO lineal

Hay veces que uno no puede ajustar linealmente, porque por ejemplo, tiene algunas de las constantes a determinar (A, f, ϕ) adentro de una función:

$$v(t) = A \sin(2\pi f t + \phi)$$

En estos casos, uno tira toda el Álgebra Lineal que sabía a la basura[11], y entra a probar con todas las combinaciones posibles... y uno se queda con la combinación que le de un menor Error Cuadrático, ¿vió?:

$$ECM(A, f, \phi) = \sum_{i=1}^{k} [v_k - v(t_k; A, f, \phi)]^2$$

[11] :'(

Ajuste NO lineal

Pero no todo es tristeza. Por suete, uno no tiene que apilar 3500 fors para probar todas las combinaciones posibles... Octave ya tiene una función que intenta encontrar el mínimo de una función, comenzando a probar desde un punto x_0. Probar:

```
%pkg load optim %usar en octave viejos
f = @(x) cos(pi.*x).*exp(-x.^2/10);

xo = 0; %probar 0, 1, 2
xmin = fminsearch(f, xo);

t=linspace(-5,5,100);
plot(t,f(t), xmin, f(xmin), 'o');
legend('f(t)', 'minimo');
```

Ajuste NO lineal

Contra de fminsearch: es bastante impredecible, pero con cuidado la podemos usar para ajustar un seno:

```
% Invento datos
fase_secreta = pi/3;
tdata = linspace(0,3,10);
vdata = sin(2*pi*tdata+fase_secreta);

% Modelo a ajustar, depende solo de phi
v_ideal = @(phi) sin(2*pi*tdata+phi);

% error cuadratico medio para un dado phi.
err = @(phi) norm(vdata - v_ideal(x));

phi_0 = 0;
phi = fminsearch(err, phi_0)
```

Ajuste NO lineal: 2 parámetros

A veces es necesario ajustar más de un parámetro:

```
% Invento datos
fase_secreta = pi/3;
A_secreta = 5;
tdata = linspace(0,3,10);
vdata = A_secreta*sin(2*pi*tdata+fase_secreta);

% Modelo a ajustar
v_ideal = @(A,phi) A*sin(2*pi*tdata+phi);

% error cuadratico medio para un dado phi.
% x=[A, phi]
err = @(x) norm(vdata - v_ideal(x(1),x(2)));

xo = [1, 0];
x = fminsearch(err, xo)
```

Ajuste NO lineal

Tarea

Medir el desfasaje en régimen estacionario en el circuito RLC simulado ajustando las curvas de corriente y tensión en el capacitor (descartar el transitorio).

Ejercicio I: Ecuación no lineal

La ecuación $f(x) = 2(x+3)\cos(x^2) - 0{,}1x^3$ posee 5 raíces (reales).

Vemos que $f(0) = 6 > 0$, y que $f(2) = -7{,}33\ldots < 0$. Esto indica que, ya que la función es contínua, debe existir algúna raíz[12] en el intervalo $[0, 2]$.

Utilizando el método de la bisección (explicado a continuación), aproximar dicha raíz.

[12] Por el Teorema de Bolzano

Ejercicio I: Ecuación no lineal (Bisección)

Método de la Bisección:

Se inicia con dos valores, $a, b \in \mathbb{R}$ tales que $f(a) > 0$ y $f(b) < 0$.

Se toma $m = (a + b)/2$, el punto medio entre a y b, y se evalúa $f(m)$.

Si $f(m) > 0$, entonces se actualiza el valor de a por el punto medio: $a = m$. Caso contrario, si $f(m) \leq 0$, entonces se actualiza el valor de b por el punto medio: $b = m$.

Repetir hasta que $|f(m)| < \Delta$, donde Δ es la cota del error en la aproximación que se busca.

Ejercicio II: Taylor I

Cree una función `mi_coseno(x)` que aproxime el coseno de `x` utilizando el desarrollo en serie de Taylor de la función coseno en torno al punto $x = 0$. Pruebe con un Taylor de orden $N = 5$ y uno de orden $N = 50$.

$$\cos(x) \approx \sum_{k=0}^{N} \frac{(-1)^k x^{2k}}{(2k)!}, \quad -\pi/2 \leq x \leq \pi/2 \tag{10}$$

Puede calcular el factorial con la función `factorial(X)` (btw, yo creía que el factorial se hacia con `fact()`. Puse `fact(5)` para probar y recibí alta sorpresa).

Ejercicio II: Taylor I bis

Sabiendo que el error de la aproximación hasta el término N es menor o igual en módulo, al término de orden $N + 1$ de la serie, modifique la función anterior para que devuelva como un valor secundario, dicha cota para el error.

El módulo puede calcularlo con la función `abs()`.

Ejercicio II: Taylor II

En vista al resultado anterior, cree una función `mi_coseno2(x,err)` que aproxime el valor de $\cos(x)$ de forma tal que el error cometido sea en módulo, menor que err.

Ejercicio II: Taylor II bis

Modifique la función anterior para que:

1. No falle cuando se pida error cero (ejemplo: `mi_coseno2(0.2,0)`).
2. Verifique que $-\pi/2 \leq x \leq \pi/2$. Si no es así, abortar con el comando `error("x fuera de rango")`.
3. Se extienda la función al intervalo $-\pi \leq x \leq \pi$ sabiendo que:
   - Si $\pi/2 < x \leq \pi$: $\cos(x) = -\cos(\pi/2 - x)$
   - El coseno es una función par: $\cos(-x) = \cos(x)$
4. Si no se especifica `err`, asuma que vale 1×10^{-10}. Para esto, puede ver la cantidad de parámetros pasados a la función con `nargin`. Ejemplo:

```
if(nargin == 1) % Si hay un solo parámetrio (x)
  err = 1e-10; % Usar valor por defecto
endif
```

Ejercicio II: Taylor II bis bis

Luego de externer la funcion `mi_coseno` al dominio $-\pi \leq x \leq \pi$, modifique la función nuevamente para que pueda funcionar con cualquier número real, sabiendo que:

1. La función coseno es períodica con período 2π
2. Si $\pi < x \leq 2\pi$: $\cos(x) = cos(\pi - x)$

Para que la función acepte cualquier $x \in \mathbb{R}$, puede calcular el ángulo equivalente ente $-\pi$ y π. Para esto, puede tomar el resto de la division del ángulo x con 2π, esto es: `alfa = mod(x,2*pi)`. De esta forma $\alpha = x + 2k\pi, k \in \mathbb{Z}$ y por ende $\cos(x) = \cos(\alpha)$.