Notas de la clase de 16 marzo, 2016.l

- Unos ejemplos de "vectorización"(eliminar ciclos)
  - Graficar $y = x^2$ en el rango $-1 \leq x \leq 1$.

Con ciclos

```
x=-1:.1:1;
y=x;
for i=1:length(x)
   y(i)=x(i)^2;
end
plot(x,y);
```

Sin Ciclos

```
x=-1:.1:1;
plot(x,x.^2);

% Nota como usamos aqui x.^2 en
% lugar de x^2
```

  - Definir el vector `[1,1/2,1/3,...,1/10]`

Con ciclos

```
x=1:10;
for i=1:10
   x(i)=1/x(i);
end
```

Sin Ciclos

```
x=1./(1:10);

% usamos ./ en lugar de / para tomar el
% reciproco de cada uno de los
% elementos de (1:10)
```

  - Definir el vector `[1,2,4,...,2^10]`

Con ciclos

```
x=0:10;
for i=1:length(x)
   x(i)=2^x(i);
end
```

Sin Ciclos

```
x=2.^(0:10);

% en lugar de ^ usamos .^
```

  - Definir el polinomio $p_n(x) = 1 + x + x^2/2 + ... + x^n/n!$

Con ciclos

```
function y=p(n,x)
y=0;
for i=0:n
 y=y+x.^i/factorial(i);
end
```

Sin Ciclos

```
function y=p(n,x)
i=(0:n);
y=sum(x.^i./factorial(i));
end
```

OJO: Si definimos a `p(n,x)` de la 2nda manera (sin ciclos) entonces `p(n,x)` NO va aceptar un vector `x`, sino solo un escalar `x` (un número). El problema es la expresión `x.^i`, que no tiene sentido cuando `x,i` son vectores de longitud distinta y > 1. Así que no sirve en un script tipo

```
x=-2:.1:2;
plot(x,p(3,x));
```

(como se requiere en el problema 4b de la guia del primer parcial). Se puede superar este problema con el comando `bsxfun`, pero me parece complicado (puedes mirar en el help) y no lo recomiendo ahora. En este caso recomiendo simplemente usar un ciclo para la definición de `p(n,x)`, nimodo.

- Se puede definir funciones dentro de un script. Esto funciona para una definición breve, de una sola linea. Por ejemplo, en lugar de poner en un archivo separado la funcion

```
function y=f(x)
y=x.^2;
end
```

se puede escribir dentro de un script que va usar la funcion $f(x)$ la linea

```
f=@(x)x.^2;
```

- Una herramienta básica de Matlab es poder seleccionar y procesar rápidamente un subconjunto de los elementos de un vector (o matriz), sin usar ciclos. Hay varias maneras, dependiendo del subconjunto de elementos que queremos escoger. Van unos ejemplos.

  Suponomes que hemos definido

```
>> x = [1 3 -2 5 -7];
```

  - Escoger los primeros 3 elementos de `x` :

```
>> x(1:3)
ans=
    [1 3 -2]
```

  - Escoger los términos positivos de `x` :

```
>> x_pos=x(x>0)
ans=
    [1 3 5]
```

  - Cambiar los términos positivos de `x` por sus cuadrados:

```
>> x(x>0)=x(x>0).^2; x
ans=
    [1 9 -2 25 -7]
```

  - Prob. 21 de la Tarea 6: tenemos definidas 3 funciones $y_1(x), y_2(x), y_3(x)$ (vectorizadas) y queremos graficar la función $y(x)$ dada por

$$y(x) = \begin{cases} y_1(x) & x < -1 \\ y_2(x) & -1 \le x \le 5 \\ y_3(x) & x > 5 \end{cases}$$

  en el rango $-5 \le x \le 6$. Vimos varias maneras (la 2nda es nueva):

```
x=-5:.1:6;
y=y2(x);
dom1=(x<-1);
dom3=(x>5);
y(dom1)=y1(dom1);
y(dom3)=y3(x=dom3);
plot(x,y);
```

  Otra opción bonita:

```
x=-5:.1:6;
dom1=(x<-1);
dom2=(x>=-1)&(x<=5);
dom3=(x>5);
```

```
y=dom1*y1+dom2*y2+dom3*y3;
plot(x,y);
```

¿Entiendes qué está pasando?

- Problema 17 de la tarea 6: una pelota está lanzada con velocidad inicial v y ángulo A (en radianes). Las coordenadas (x, h) (altura y distancia horizontal) como función de tiempo t están dadas por

$$x(t) = v\cos(A)t$$
$$h(t) = v\sin(A)t - \frac{g}{2}t^2$$

donde $g = 9.81$ (la aceleración de la gravedad).

Para $v = 10$, $A = 35$ grados, encontrar la altura máxima h_{max} que alcanza, la distancia horizonal x_{max} que alcanza al caer al suelo, y el tiempo t_{max} que le toma caer al suelo. Graficar la trayectoria de la pelota para $h > 0$.

```
t=0:.1:10;   % un rango de tiempo dentro del cual seguro regresa al suelo
v=10;
A=35*pi/180;  % conversion de grados a radianes
g=9.81;
x=v*cos(A)*t;
h=v*sin(A)*t-(g/2)*t.^2;
h_max=max(h);
t_max=max(t(h>0));
x_max=x(t==t_max);
plot(x(h>0), h(h>0));
```

Otra opción: detectar primero el `t_max` mediante un ciclo `while`

```
g=9.81; v=10; A=35*pi/180;
h_fun=@(t)v*sin(A)*t-(g/2)*t.^2; % la altura
t_max=.1;
while h_fun(t_max)>=0
   t_max=t_max+.01;
end
t=0:.1:t_max;
h=h_fun(t);
h_max=max(h);
x=v*cos(A)*t;
x_max=x(end);  %  el ultimo elemento de x
plot(x, h);
```