

Práctica 1

Uso de MyApps URJC

Introducción a MATLAB:

Comandos básicos y creación de scripts

ÍNDICE

1. Instalación y uso de MyApps URJC	3
1.1 Instalación guiada de MyApps en un equipo informático propio	3
1.2 Acceso a MyApps desde Windows 10	4
1.3 Arranque de MATLAB en MyApps	7
2. Introducción a MATLAB	8
2.1 Interfaz de usuario de MATLAB	9
2.2 Comandos de ayuda y de sistema en MATLAB	10
2.1 Vectores y matrices	10
2.3 Definición de polinomios	12
2.4 Creación de gráficos	12
2.5 Creación de <i>scripts</i> .m (ficheros para ejecución de comandos por lotes)	15
2.6 Y entonces, ¿qué es son los “live scripts” de MATLAB?	18
2.7 Directorio de trabajo para MATLAB usando MyApps	19
2.8 Para saber más sobre MATLAB	19

1. Instalación y uso de MyApps URJC

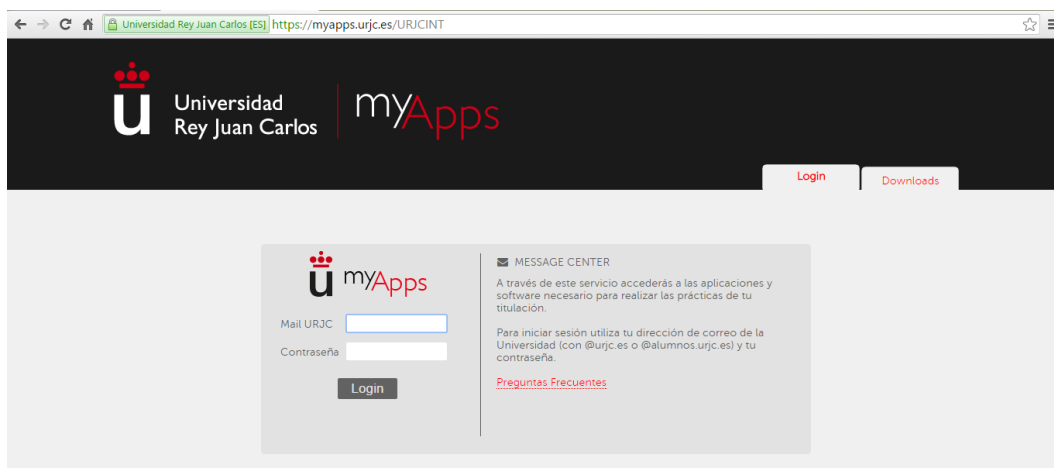
1.1 Instalación guiada de MyApps en un equipo informático propio

Para la realización de las prácticas de la asignatura de Fundamentos de Automática es necesario el software de cálculo numérico y programación denominado MATLAB.

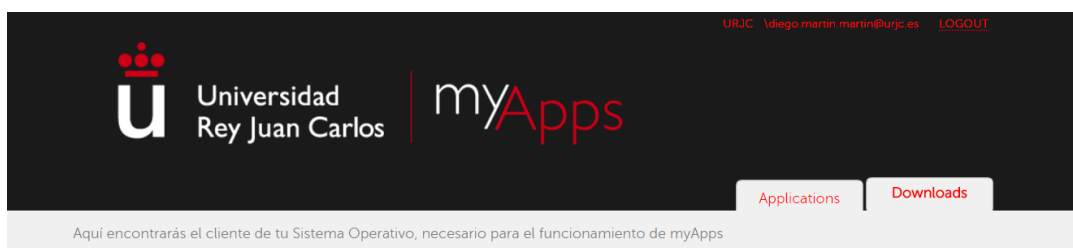
Desde hace algunos años la Universidad Rey Juan Carlos pone a disposición de los alumnos una herramienta de virtualización de aplicaciones informáticas denominada MyApps. Para acceder a la misma y usar sus aplicaciones necesitas tener una conexión a internet.




Abre el navegador e ingresa en la siguiente URL utilizando tu correo electrónico URJC y tu contraseña:

<https://myapps.urjc.es>

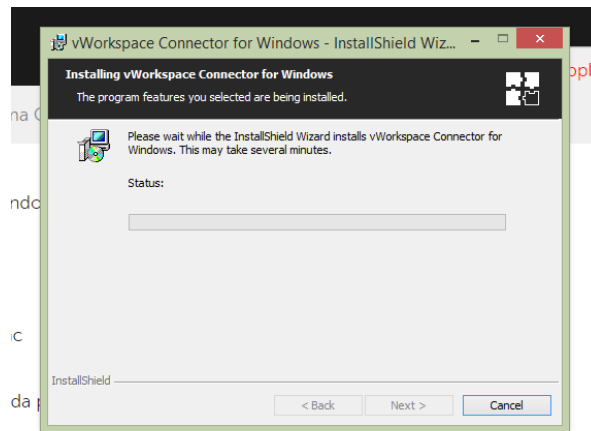


1. La primera vez que accedas a MyApps deberás instalar un cliente en tu equipo. Para ello, haz clic en la versión de tu Sistema Operativo y sigue las instrucciones de descarga e instalación.



-  Descarga cliente de Windows
-  Descarga cliente Linux
-  Descarga cliente de Mac

2. Usa las opciones de instalación que aparecen por defecto:

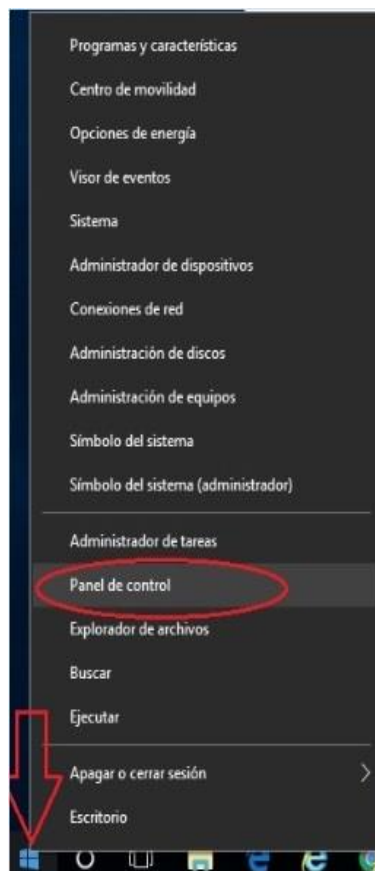


1.2 Acceso a MyApps desde Windows 10

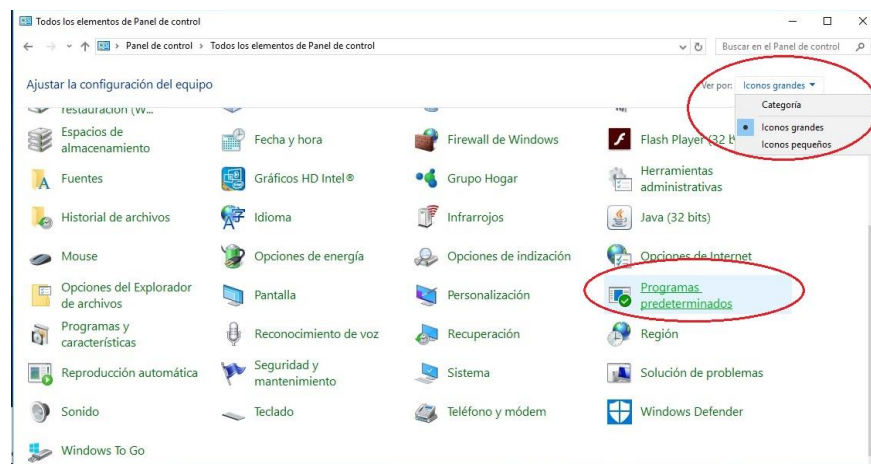
(Extraído de las “Preguntas frecuentes sobre MyApps” de <https://cau.urjc.es/myapps/>)

Si al acceder a myApps, y ejecutar una de las aplicaciones, se te descarga un fichero ".pit" y no consigues asociarlo al programa vWorkspace, debes realizar los siguientes pasos:

1. Iremos al Panel de Control (pulsando con el botón derecho del ratón sobre el menú de inicio),



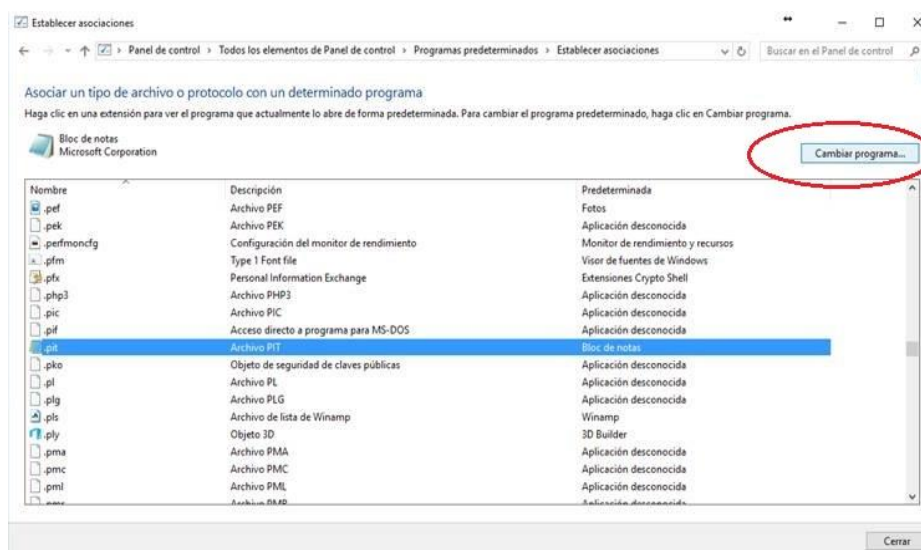
En la siguiente ventana que se nos muestra, lo configuraremos como "iconos grandes" y seleccionaremos "Programas Predeterminados"



2. Seleccionaremos "Asociar un tipo de archivo o protocolo con un programa"



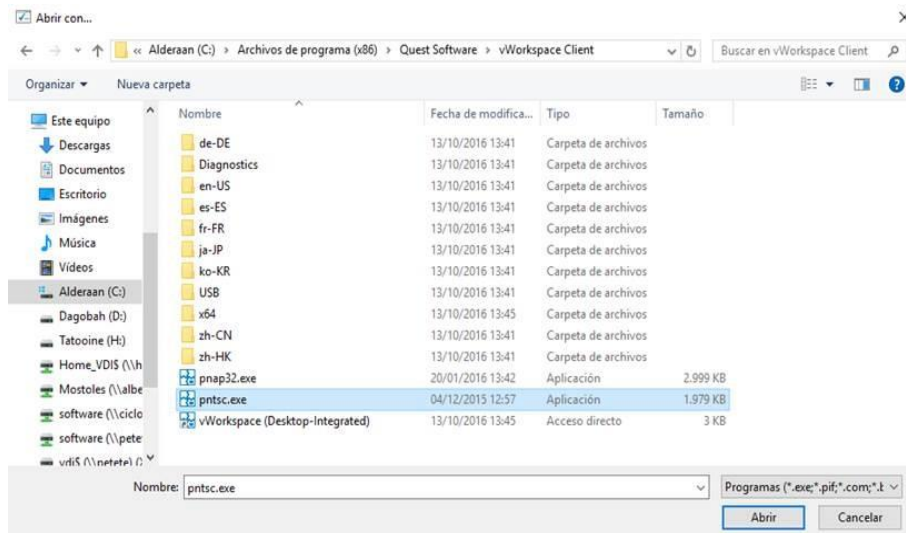
3. Tenemos que seleccionar la extensión **.pit** y darle a "Cambiar Programa"



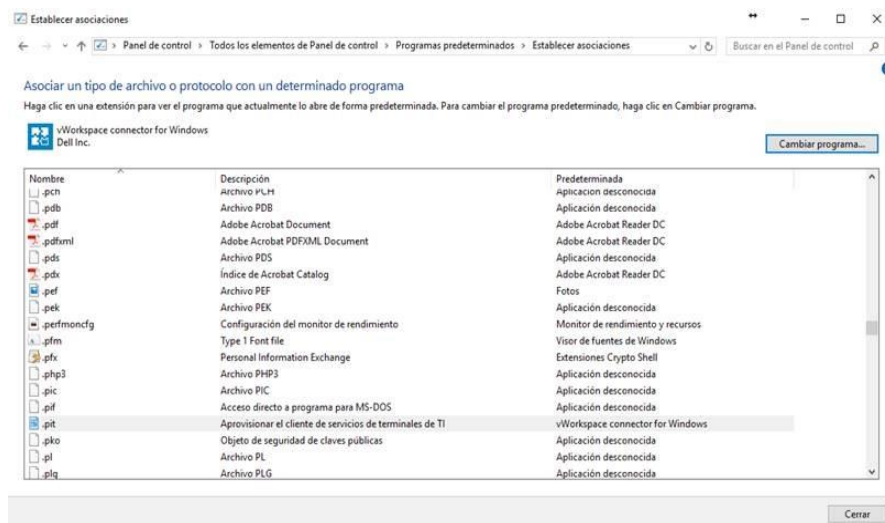
4. Hay que darle a "Buscar otra aplicación"



5. Le tenemos que indicar la siguiente ruta C:\Program Files (x86)\Quest Software\vWorkspace Client y seleccionar **pntsc.exe**

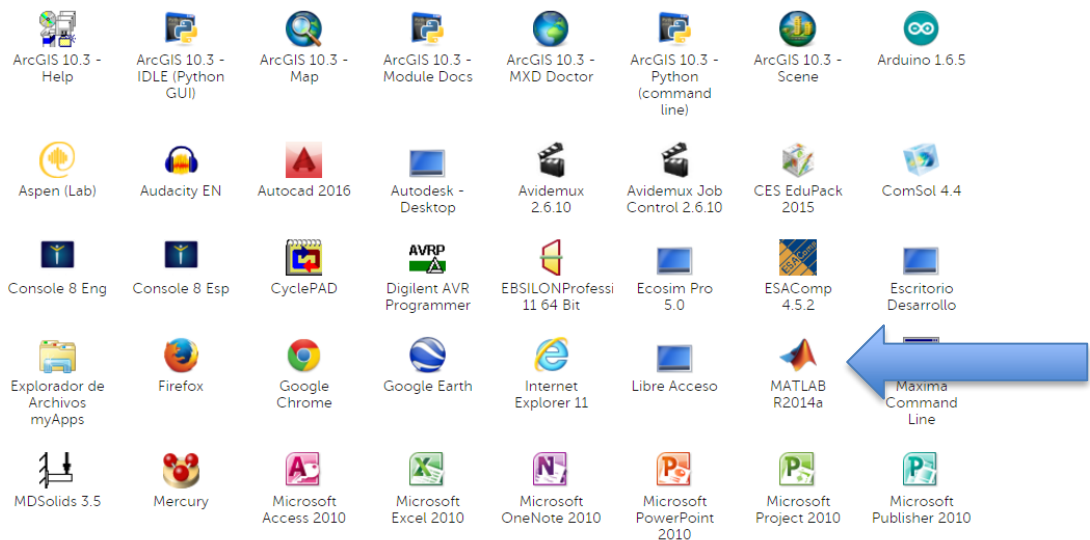


6. Y al ejecutarlo a partir de ahora, ya se asocia correctamente, y nos lo va a abrir automáticamente.



1.3 Arranque de MATLAB en MyApps

Una vez terminada la instalación ya podrás acceder al conjunto de aplicaciones disponibles para tu usuario a través del navegador. Comprueba que todo funcione correctamente y que MATLAB se encuentre entre las aplicaciones instaladas para tu usuario. Anota el número de versión que te aparece e intenta arrancar el programa.

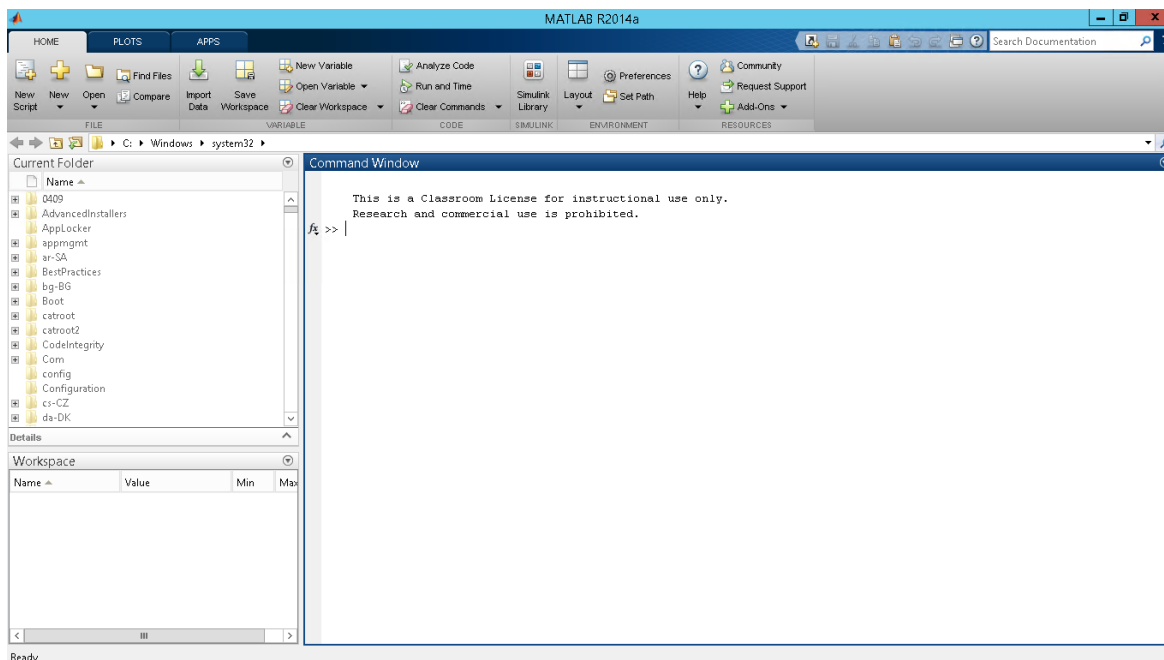


Recientemente se ha incluido en MyApps la opción de visualizar las aplicaciones **directamente en el navegador web**. Esto permite acceder a las herramientas sin necesidad de tener el cliente de MyApps instalado, algo útil para acceder desde dispositivos móviles, por ejemplo.

Para activar esta opción debes hacer clic el marcador “HTML5” que aparece en la esquina superior derecha de la página de inicio de MyApps:



Arranca MATLAB y comprueba que puedes acceder a la aplicación. Si no dispones de una conexión rápida a internet, a veces puede tardar varios segundos. El resultado de este primer apartado de la práctica debería ser una ventana parecida a la siguiente:



2. Introducción a MATLAB

MATLAB, acrónimo de **MATrix LABoratory**, es hoy en día una de las principales herramientas *software* existentes en el mercado para el cálculo matemático, análisis de datos, simulación y visualización de resultados. Todas las operaciones que realiza MATLAB se basan en una estructura de datos matricial. Dentro del entorno de trabajo de MATLAB, se pueden definir nuevos comandos o funciones, programadas por el propio usuario, a través de ficheros **.m**. Este tipo de ficheros se encuentran en las llamadas *toolbox* de MATLAB, que son una colección de funciones ya programadas y disponibles para el usuario.

Dentro del campo del control, MATLAB ha desarrollado un gran número de funciones para el análisis de los sistemas de control automático. Todas ellas se encuentran dentro del **Control System Toolbox**, que permite el análisis de sistemas de control en el dominio del tiempo y de la frecuencia, tanto de sistemas continuos como discretos.

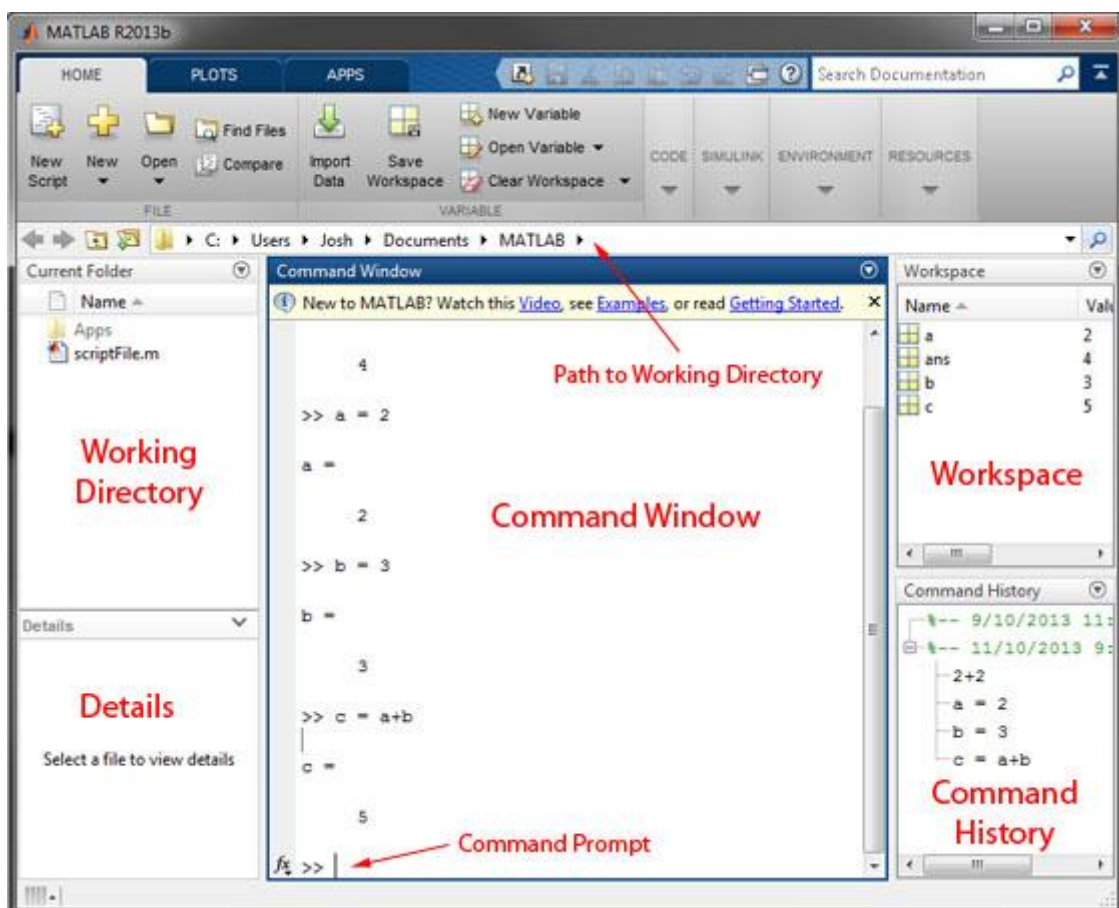
MATLAB incorpora interfaces gráficas como la *Root Locus Tool* para el análisis y diseño de sistemas de control mediante el método del lugar de las raíces o la *System Identification Tool* para identificación de sistemas.

La interfaz gráfica por excelencia es *Simulink*, que es una aplicación integrada dentro de MATLAB, y que permite simular mediante la construcción gráfica de diagrama de bloques.

2.1 Interfaz de usuario de MATLAB

Aunque depende de la versión del programa que estemos usando, al iniciar MATLAB aparece una interfaz gráfica de usuario que contiene tres elementos básicos que debemos conocer bien. Estos elementos básicos son los siguientes:

- **Consola de comandos:** Situada en la parte central de la ventana de la aplicación, nos permite introducir y ejecutar comandos en el prompt (`>>`) y visualizar los últimos comandos ejecutados.
- **Directorio de trabajo:** Situado a la izquierda, muestra los archivos contenidos en el directorio de trabajo. Dicho directorio se puede seleccionar desde la barra superior.
- **Workspace o espacio de trabajo:** conjunto de variables introducidas en MATLAB en la sesión actual, junto con sus valores actuales (se puede guardar y volver a cargar más adelante):



Cada una de estas áreas de la ventana principal se puede maximizar, minimizar o extraer a una ventana independiente ("undock"). Para recuperar la apariencia original, se puede utilizar la opción "Default" dentro del menú "Home ->Layout"

2.2 Comandos de ayuda y de sistema en MATLAB

Con el comando **help**, seguido del nombre de una instrucción o comando, se obtiene una ayuda **en línea** con información sobre la utilidad dicho comando. Además, muestra un resumen de las diferentes sintaxis y argumentos de los mismos.

Para obtener una información más detallada de una instrucción o comando, denominada “página de referencia” (ventana independiente, que incluye la sintaxis, descripción ejemplos de usos, comandos relacionados, etc.) se debe utilizar el comando **doc**, seguido del nombre de dicho comando.

Si se ejecuta simplemente **doc** se accede a la página principal de la documentación de MATLAB, categorizada por bloques de herramientas o Toolboxes.

¿Y qué ocurre si no sabemos o no recordamos el nombre del comando? En ese caso, podemos buscarlo con el comando **lookfor**, seguido de alguna palabra clave relacionada con la funcionalidad u objetivo del comando que estamos buscando.

Ejercicio práctico 1. Uso de comandos de ayuda

1. Averigua los Toolboxes que están instalados en tu versión de MATLAB, y si alguno de ellos tiene que ver con *Sistemas de Control*.
2. Busca los comandos de MATLAB que tienen que ver con funciones de transferencia (*transfer function*).

Mediante **memory** podrás conocer la memoria principal (RAM) instalada en tu equipo (sea virtual o no), además de la disponible para MATLAB

```
>> memory
Maximum possible array:      6877 MB (7.211e+09 bytes) *
Memory available for all arrays: 6877 MB (7.211e+09 bytes) *
Memory used by MATLAB:      635 MB (6.663e+08 bytes)
Physical Memory (RAM):      8192 MB (8.589e+09 bytes)

* Limited by System Memory (physical + swap file) available.
fx >> |
```

2.1 Vectores y matrices

En MATLAB los **vectores** y **matrices** se definen entre corchetes, con sus elementos separados por espacios o por comas. En el caso de las matrices, cada fila se delimita con un punto y coma.

```
>> vector = [1 2 3]
```

vector =

```
1 2 3
```

```
>> matriz= [1 3 5;2 4 6; 7 9 0]
```

matriz =

1 3 5

2 4 6

7 9 0

Los vectores también se pueden definir automáticamente si sus componentes están equiespaciados, indicando el valor inicial, incremento y valor final:

>> vector2 = 1:2:19 % del 1 al 19, con un incremento de 2 unidades entre componentes

vector2 =

1 3 5 7 9 11 13 15 17 19

Un vector columna se construye separando elementos con punto y coma o transponiendo un vector fila con el operador `'`.

>> vector3 = [4;3;2]

vector3 =

4

3

2

Ejercicio práctico 2. Creación de vectores y matrices

1. Averigua para qué sirven las instrucciones **vector_1 = linspace(0,10)** y **vector_2 = logspace(1,1000)**.
2. Utilizando una única instrucción, crea una matriz A de dimensiones 5 x 10 donde sus elementos sean los 50 primeros números enteros, en orden de menor a mayor.

En MATLAB, cuando se necesita realizar operaciones **elemento a elemento** con una matriz se deben utilizar los operadores precedidos por un punto:

- Multiplicación elemento a elemento de dos matrices A y B: **A.*B**
- División elemento a elemento de dos matrices A y B: **A./B**
- Potenciación de una matriz A a la potencia n elemento a elemento: **A.^n**

Ejercicio práctico 3. Operaciones elemento a elemento en matrices

1. Crea una nueva matriz B donde cada elemento B_{ij} sea el cuadrado de cada elemento A_{ij} de la matriz A del ejercicio práctico 2, punto 2.
2. Multiplica elemento a elemento las matrices A y B

2.3 Definición de polinomios

En MATLAB los polinomios se definen como vectores filas, siempre entre corchetes, los coeficientes de sus elementos en orden de potencia **descendente**. Se debe añadir un cero en la posición de aquellos elementos que no existen dentro del polinomio.

```
>> p1 = [1 6 5 -3]           %definición del polinomio  $x^3 + 6x^2 + 5x - 3$ 
>> p2 = [2 0 1 -1 1]        %definición del polinomio  $2x^4 + x^2 - x + 1$ 
```

Para obtener las **raíces de un polinomio** se utiliza el comando **roots**:

```
>> r1 = roots(p1)
```

r1 =

-4.8385

-1.5592

0.3977

Para definir un polinomio a través de sus raíces se utiliza el comando **poly**:

```
>> r3 = [-1;0.5+i;0.5-i];
```

```
>> p3= poly(r3)
```

p3 =

1.0000 0 0.250 1.250 % el resultado es el polinomio $x^3 + 0.25x + 1.25$

Ejercicio práctico 4. Manejo de polinomios

1. Halla los tres polinomios que tienen una raíz quántuple igual a -1, a 2 y a -3, respectivamente.
2. ¿Qué característica tienen los polinomios con todas sus raíces reales y negativas?
2. Halla los polinomios que tiene las raíces complejas iguales a -i, i, tanto simples como dobles y triples. ¿Encuentras algo en común entre ellos?
3. Calcula las raíces del polinomio $x^{10} + x^5 - x + 1$

2.4 Creación de gráficos

Mediante la función “**plot**” de MATLAB se pueden representar funciones bidimensionales del tipo $y = f(t)$ de la siguiente forma: dados los vectores de igual longitud “t” y “y” la función se dibuja con el comando **plot(t,y)**.

Si t es tiempo, se puede generar un vector de la siguiente forma $t=[0:1:100]$. Este *array* genera un vector de 101 elementos de valores comprendidos entre el 0 y el 100.

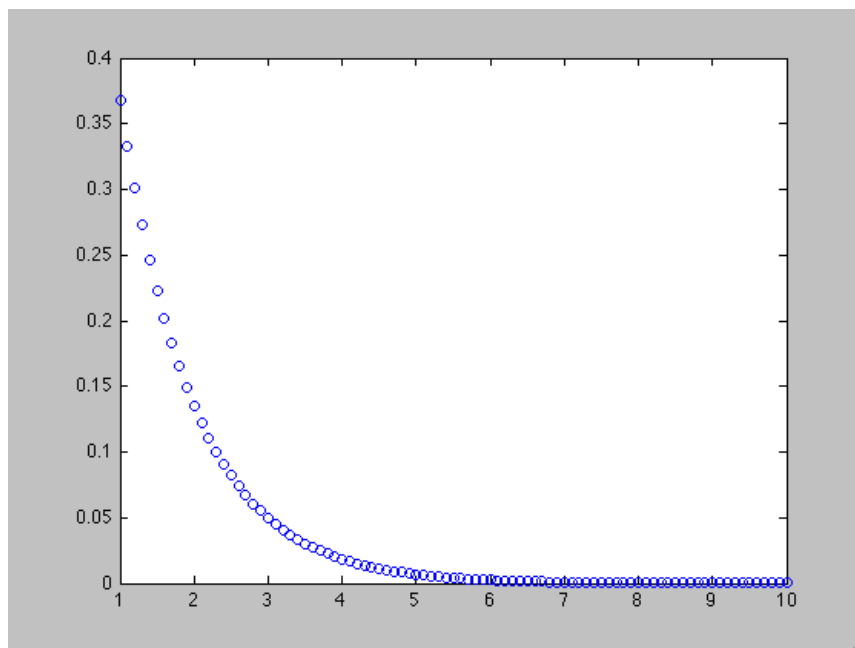
Si se quisiera representar $y=e^{-t}$, en un rango de t entre 0 y 10 a intervalos de 0.1 marcando los puntos de la gráfica con un símbolo circular 'o', entonces, las instrucciones serían:

```
>> t=[0:0.1:10]
```

```
>> y=exp(-1*t)
```

```
>> plot(t,y,'o')
```

que daría como resultado:



MATLAB genera una sola gráfica para la representación de resultados con el nombre *Figure*. Si se desea representar sucesivos resultados sobre la misma gráfica, es necesario llamar a la función `plot` con sucesivos argumentos x,y , por ejemplo `plot(x,y1,x,y2)`. También se puede activar el comando **hold**, mediante la instrucción **hold on**. La desactivación del comando `hold` se realiza mediante la instrucción **hold off**.

La creación de nuevas ventanas de gráficas se realiza con el comando **figure**.

Los resultados gráficos se pueden personalizar especificando tres tipos de atributos: **color**, **símbolo y estilo (tipo, ancho) de línea**. La opción elegida se implementa añadiendo al comando `plot` una cadena de tres caracteres, entre comillas simples, indicando la opción deseada para cada atributo. Para ver los distintos tipos de opciones consultar la ayuda mediante los comandos `doc plot` y `doc LineSpec`

Ejercicio práctico 5. Gráficas de funciones reales. Utiliza distintas opciones de color, símbolo y tipo de línea que aparecen explicadas en la ayuda (doc plot) para:

1. Realizar las gráficas de $f(t) = t^2$ y $f(t) = t^3$ en el intervalo $t = [-3, 3]$ en el mismo gráfico
2. Realizar la gráfica de $f(t) = e^{-a \cdot t} \cdot \cos(t)$ en el intervalo $t = [0, 5]$, para distintos valores de a
3. Realizar la gráfica de $f(t) = e^{-t} \cdot \cos(bt)$ en el intervalo $t = [0, 5]$, para distintos valores de b

Los comandos **grid**, **xlabel**, **ylabel** y **title** añaden a la gráfica una rejilla, un texto a los ejes y un título respectivamente.

La modificación del escalado de los ejes X-Y en una gráfica se realiza con el comando **axis**, indicando mediante un vector de cuatro elementos el valor mínimo y máximo del eje de abscisa y de ordenada.

```
>>axis([minX maxX minY maxY])
```

Ejercicio práctico 6. Prueba los comandos anteriores en las gráficas realizadas en el ejercicio práctico 5.

Una gráfica se puede subdividir en zonas para sucesivas representaciones con el comando **subplot**. La gráfica, a modo de matriz, se divide en celdas a través del número de filas y columnas que se indique en el comando; a su vez se debe especificar qué celda activar para la representación de resultados.

```
>>subplot(1,2,1)    % Dividir en dos zonas una figura (1 fila, 2 columnas,).
                    % Activando la zona izquierda (zona 1, tercer argumento) para el plot.
```

Para obtener más información sobre el commando subplot, consulta la ayuda de MATLAB mediante **doc subplot**

































































Ejercicio práctico 7. Utilizando el comando subplot:

1. Introduce las gráficas b y c del ejercicio práctico 5 en una única gráfica dividida en una fila y dos columnas.
2. Crea una gráfica con subplot que contenga cuatro subgráficas de las funciones trigonométricas $\sin(t)$, $\cos(t)$, $\tan(t)$, $\sin(t) \cdot \cos(t)$.

Aunque en esta asignatura serán menos importantes, MATLAB también permite realizar gráficas de funciones de varias variables, histogramas, gráficos de barras, etc. Para obtener más información se puede consultar “Types of MATLAB Plots” en la ayuda de MATLAB, o consultar la tabla de comandos de gráficos de la siguiente página.

Types of MATLAB Plots

There are various functions that you can use to plot data in MATLAB®. This table classifies and illustrates the common graphics functions.

Line Plots	Pie Charts, Bar Plots, and Histograms	Discrete Data Plots	Polar Plots	Contour Plots	Vector Fields	Surface and Mesh Plots	Volume Visualization	Animation	Images	
plot 	area 	stairs 	polarplot 	contour 	quiver 	surf 	mesh 	streamline 	animatedline 	image 
plot3 	pie 	stem 	polarhistogram 	contourf 	quiver3 	surfc 	meshc 	streamslice 	comet 	imagesc 
semilogx 	pie3 	stem3 	polarscatter 	contour3 	feather 	surf1 	meshz 	streamparticles 	comet3 	
semilogy 	bar 	scatter 	compass 	contourslice 		ribbon 	waterfall 	streamribbon 		
loglog 	barh 	scatter3 	ezpolar 	fcontour 		pcolor 	fmesh 	streamtube 		
errorbar 	bar3 	spy 				fsurf 		coneplot 		
fplot 	bar3h 	plotmatrix 				fimplicit3 		slice 		
fplot3 	histogram 	heatmap 								
fimplicit 	histogram2 									
	pareto 									

Ejercicio práctico 8. Gráficas 3D de funciones de varias variables usando “mesh()”:

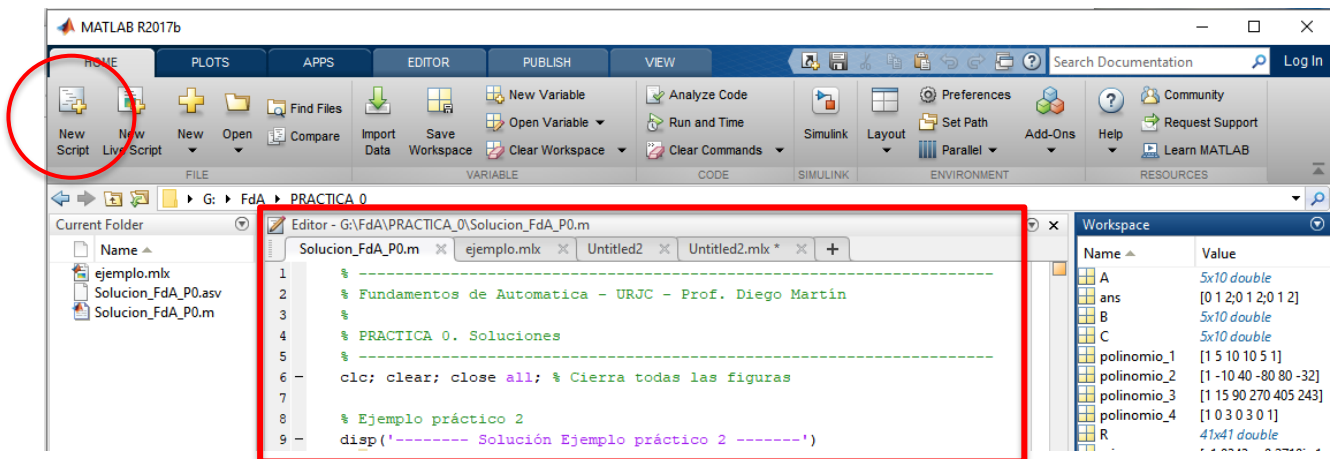
1. Realiza una gráfica de la función de dos variables $f(x,y) = x^2 + y^2$ en el intervalo $[-2,2]$ para x e y .
2. Realiza una gráfica de la función de dos variables $f(x,y) = \sin(x^2 + y^2)/(x^2 + y^2)$ en el intervalo $[-10,10]$ para x e y .
3. Realiza una gráfica de la función de dos variables $f(x,y) = \sin(x)*\cos(y)$ en el intervalo $[0,10]$ para x e y . Cambia el mapa de color del gráfico a la opción “jet” mediante **colormap(jet)** y observa el resultado.

Nota: Estudia el comando mesh(), y encontrarás que para usarlo las variables x e y deben ser una malla (matriz), no un vector. Dicha malla puede crearse con el comando “meshgrid”.

2.5 Creación de scripts .m (ficheros para ejecución de comandos por lotes)

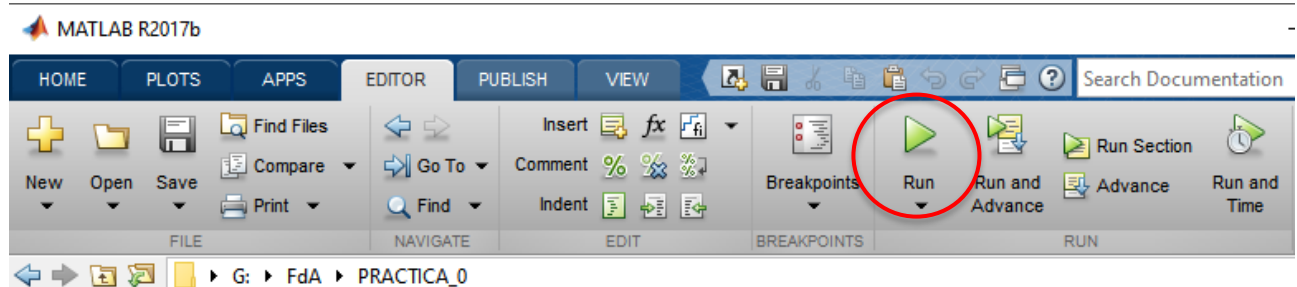
En MATLAB tienen especial importancia los **ficheros de extensión .m**. Dichos ficheros o scripts contienen conjuntos de comandos a ejecutar (o definiciones de funciones)

Un fichero .m es un fichero de texto plano (sin formato) y que pueden crearse a partir de un editor de textos cualquiera. No obstante, lo mejor es utilizar el editor del propio MATLAB al que se accede por defecto al abrir un nuevo fichero (Pestaña HOME → New Script):



Un fichero .m se ejecuta al teclear su nombre en la línea de comandos y pulsar intro. Es esencial que el fichero se encuentre almacenado y guardado en el directorio de trabajo seleccionado (*Current Folder*).

Otra opción para su ejecución es hacer clic en el icono RUN desde la pestaña EDITOR. Para ello el fichero debe estar guardado. En caso contrario MATLAB lo guardará automáticamente. Las salidas de los diferentes comandos se mostrarán en la consola.



Debes saber que en MATLAB la ejecución de un script .m es secuencial (de manera equivalente a los lenguajes de programación interpretados, como Python), es decir, se ejecutarán los comandos uno por uno, en orden de aparición, hasta el final del fichero o hasta que aparezca una línea que contenga un error de sintaxis.

En caso de error, la ejecución se parará y en la consola aparecerá, en color rojo, una descripción del error y de la línea del comando que lo contiene.

En los ficheros .m resulta muy recomendable introducir una cabecera con el nombre del autor, la fecha y una breve descripción de su contenido. Se pueden realizar mediante el uso de comentarios (utilizando % al principio de la línea de comentario). Su uso resulta fundamental para la organización y reutilización del trabajo que realicemos

```
% Esto es una línea de comentario en un fichero .m de MATLAB
```

Además, es recomendable incluir tras la cabecera los siguientes comandos, que nos permiten borrar el espacio de trabajo, la consola de comandos y las figuras previamente existentes al ejecutar el archivo .m:

```
clc; clear; close all;
```

Puedes usar el siguiente comando para mostrar un texto por la línea de comandos:

```
disp('----- Solución Ejemplo práctico 2 -----')
```

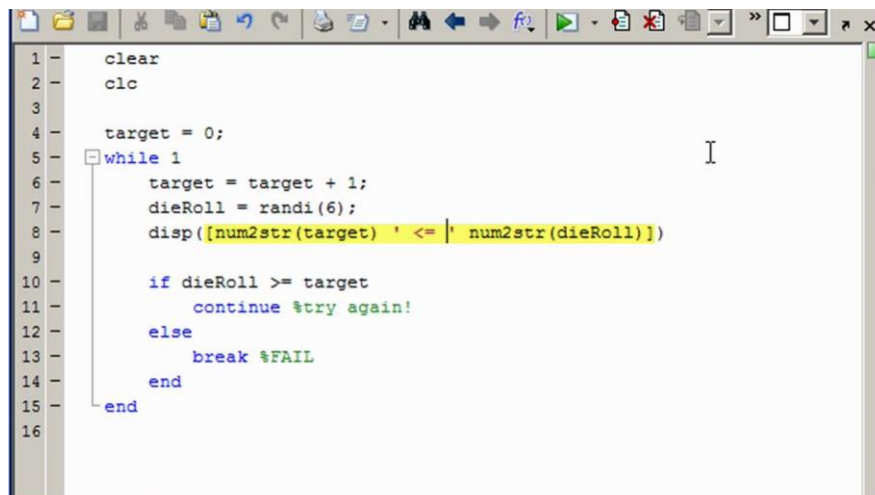
Por último, se puede dividir el archivo en secciones de código, para trabajar y ejecutar el código en diferentes fragmentos, controlando el flujo de ejecución, utilizando %%:

```
%% Un doble % indica el inicio de una sección de código
```

Ejercicio práctico 9

Crea un archivo .m para cada uno de los ejercicios prácticos anteriores, que contenga una cabecera, la sentencia de borrado descrita, comentarios y todos los comandos que has utilizado previamente.

Utilizando los archivos .m, en MATLAB se puede programar, ya que también están disponibles siguientes instrucciones de recursividad (control de flujo) **for** y **while** (**continue**, **break**) y sentencias condicionales **if** (**elseif**, **else**, **end**) y **switch** (**case**, **otherwise**), haciendo de MATLAB un **lenguaje de programación estructurada**:



```

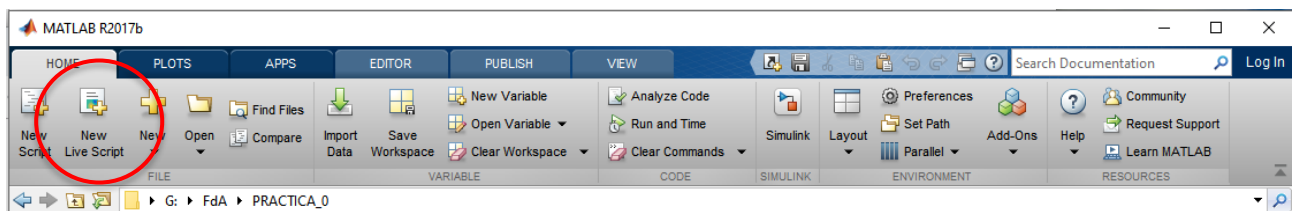
1 - clear
2 - clc
3
4 - target = 0;
5 - while 1
6 -     target = target + 1;
7 -     dieRoll = randi(6);
8 -     disp([num2str(target) ' <= ' num2str(dieRoll)])
9
10 -    if dieRoll >= target
11 -        continue %try again!
12 -    else
13 -        break %FAIL
14 -    end
15 - end
16

```

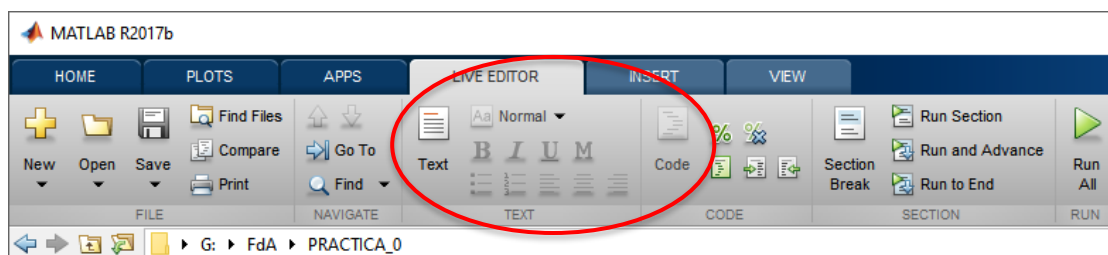
2.6 Y entonces, ¿qué es son los “live scripts” de MATLAB?

En las últimas versiones de MATLAB han aparecido otro tipo de scripts denominados “*Live Scripts*” o “*Guiones en vivo*”. Se trata de ficheros con extensión `.mlx` que funcionan como documentos interactivos, combinando texto formateado con código de MATLAB ejecutable, cuya ejecución aparece insertada debajo (o al lado) de cada celda del *live script*, lo que permite generar archivos HTML, PDF o LaTeX para publicar

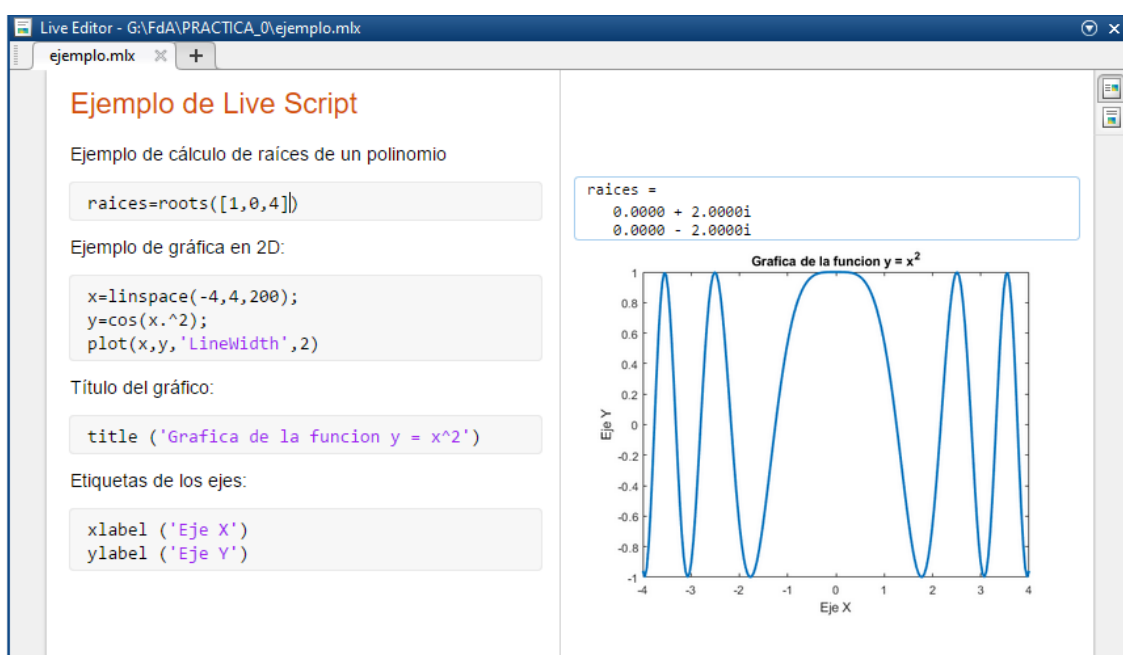
Para crear un Live Script puedes hacer clic en HOME → New Live Script:



Aparecerá un editor denominado “Live Editor” que lleva asociada una barra de herramientas propia, que te permitirá insertar código y celdas de texto intercaladas:



Este es el aspecto de un Live Script sencillo. Para más información, teclea **doc Live Script**.

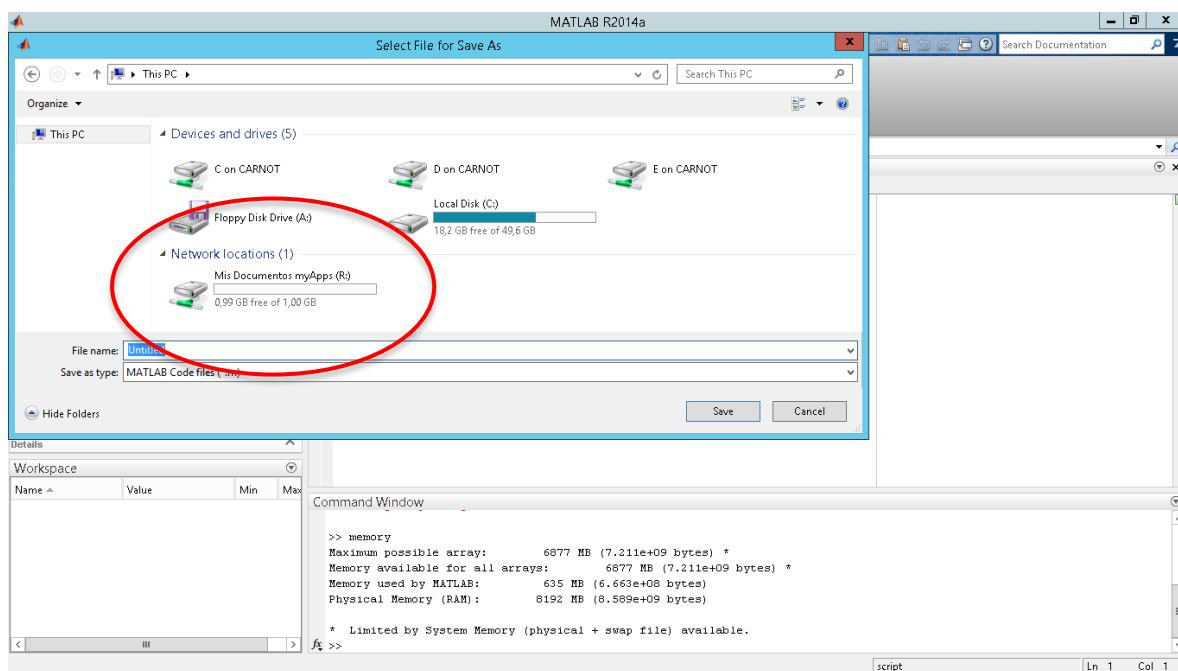


Ejercicio práctico 10.

Crea un Live Script como el de la imagen anterior o de alguno de los ejercicios prácticos anteriores. Prueba a exportarlo tanto a PDF (Save → Export to PDF) como a HTML (Save → Export to HTML), visualizando la salida desde el visor de archivos PDF y desde un navegador de internet, respectivamente.

2.7 Directorio de trabajo para MATLAB usando MyApps

Como ya seguramente sabes, en el servicio de MyApps de la URJC se dispone de un disco duro virtual (unidad R:) de 3 GB de espacio total, al que se puede acceder desde cualquier aplicación ejecutada en MyApps y desde cualquier equipo. Cuando trabajes con MATLAB desde MyApps, debes seleccionar siempre una carpeta de R: como directorio de trabajo:



Sin embargo, si usas MATLAB desde un ordenador propio resulta recomendable crear un directorio en un disco duro local, (C:, D: ...), establecerlo como directorio de trabajo y utilizarlo para almacenar todos los archivos .m que se vayan creando durante las prácticas de esta asignatura.

2.8 Para saber más sobre MATLAB

Si deseas profundizar en tu conocimiento de MATLAB, aprender más operaciones básicas con matrices, importar ficheros de datos o realizar programas más completos, te recomendamos que consultes tanto el documento PDF titulado “MANUAL DE MATLAB” que se adjunta a esta práctica como la ayuda y los cursos online de MATLAB disponibles en <https://matlabacademy.mathworks.com/es>.