

Margaret Hamilton (1936)



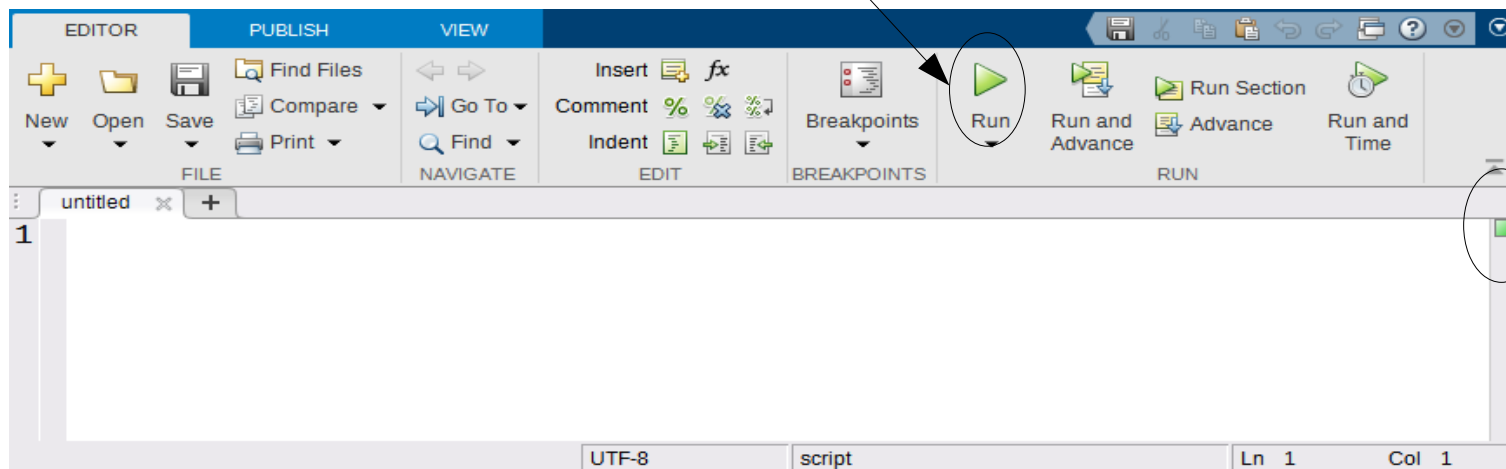
- Primeira enxeñeira do software
- Desenvolveu o **software de navegación** para o programa espacial EEUU (años 60)
- Xefa da equipa de programación da NASA para a viaxe á lúa
- Medalla Presidencial de Liberdade (2016)

Programas

- Ficheiros coa extensión `.m`: conteñen comandos que se executan secuencialmente
- Execución: escribe o seu nome (sen `.m`) na ventá de comandos
- O arquivo debe estar no directorio actual (ou nun directorio incluído na variábel *path*, que se pode consultar co comando *path*)
- Pódese engadir directorios a *path* no menú *File* submenú *Set Path* ou co comando *addpath(dir)*
- Execución alternativa: na ventá de directorio, seleccionar arquivo e *Run* (F5) no menú contextual
- Ou dende o editor de Matlab, menú *Debug -> Run*

Edición do programa

- Botón *New* en barra botóns: abre o editor de Matlab (dende o cal tamén se pode executa-lo programa)
- Execución: botón *Run* (F5) no editor ou escribi-lo nome do programa + Intro na ventá de comandos
- Permite depura-lo programa durante a execución
- Comentarios con %



Execución do programa (I)

- Matlab: linguaxe interpretado (o programa necesita ao Matlab para executarse)
- Non hai erros de compilación (non hai compilación): só erros de execución e lóxicos
- Matlab só atopa un erro de sintaxe cando executa o programa e chega ao erro
- O editor de Matlab indica con cor vermella, laranxa ou verde se o programa ten erros (vermello), advertencias (laranja) ou é correcto (verde) antes de executalo
- Pode haber erros aínda que o indicador sexa verde

Execución do programa (II)

- Co Matlab dende a terminal de comandos de Linux (ponlle *exit* ao final do programa para que o Matlab retorne á terminal):

matlab -nosplash -nodesktop -r programa

- Dentro do Octave: escribe o nome do programa sen a extensión *.m*
- Co Octave dende a terminal de comandos de Linux:

octave programa.m

- Outra forma co octave dende a terminal de Linux:

- 1) Engade a liña *#!/usr/bin/octave* ao comezo do programa para indicarlle ao *bash* que interprete o programa co Octave.
- 2) Na terminal (*bash*), executa *chmod u+x programa.m* para darlle permiso de execución.
- 3) Executa *programa.m* dende a terminal.

Depurador de Matlab

- No editor de Matlab, abre o programa a depurar
- Establece un punto de ruptura (*breakpoint*) pulsando no marxe esquerdo na liña desexada
- Executa o programa (F5 ou menú *Debug->Run* no editor): detense a execución no punto de ruptura
- Podes inspecciona-las variábeis poñendo o rato sobre o variábel no programa (ou no workspace)
- Podes executa-lo programa sentenza a sentenza con F10 (ou menú *Debug->Step*)
- Entra nunha función: F11 ou *Debug->Step into*
- Continua a execución: F5 ou *Debug->Continue*

Variáveis e entrada de datos

- O programa usa as variáveis globais (as do workspace)
- Recomendábel executar *clear* ao comezo do programa (borra as variáveis existentes)
- Entrada de datos por teclado:

var = input('introduce un valor: ');

cadea = input('introduce unha cadea: ', 's');

- Exemplo:

x = input('introduce x: ');

- Podes introducir un vector entre corchetes: [1 2 3]
- Matriz: punto e coma separando filas: [1 2 3;4 5 6]

Saída de información por pantalla

- Comando *disp*:

disp(var); disp('mensaxe de texto');

- Comando *fprintf*:

fprintf('formato', var1, ..., varN);

- 'formato': cadea con caracteres, e tamén ...
- códigos de formato (ver páx. seguinte)
- secuencias de control: **\n** para nova liña, **\t** para tabulador, **\r** para retorno de carro, **\b** para borrar un carácter impreso, **** (carácter '\'), **%%** (carácter '%'), **\"** (carácter "');

Códigos de formato

- **%c**: carácter simple
- **%s**: cadea de caracteres
- **%i** ou **%d**: enteiro; **%5i** para enteiro de ancho 5.
- **%f**: nº real sen expoñente; **%.6f**: con 6 decimais; **%10.3f**: con ancho 10 (incluindo o punto decimal) e 3 cifras decimais
- **%e**: nº real en formato exponencial; **%10.2e**: real exponencial con ancho 10 e 2 decimais (tamén con **%n.dE**, neste caso E no expoñente)
- **%g**: nº real na forma máis compacta entre **%e** ou **%f**
- Exemplo: `x=3.5;t=5; s='ola';
fprintf('x= %.3f x=%.2e t= %4i s=%s\n', x, x, t, s);`

Función *fprintf*

- *fprintf* está vectorizada: se unha variábel é vector ou matriz, repítese a función até que se imprimen tódolos elementos (por columnas) na mesma liña, agás que se poña `\n`.
- Ex: `x = [1 2; 3 4]; fprintf('x=%i\n', x);`
x=1 % en cada liña por ter \n
x=3
x=2
x=4
- Ex: `fprintf('%i ', x);`
1 3 2 4 % na mesma liña por non ter \n

Función *sprintf*

- *fprintf* también permite almacenar nun arquivo
- A función *sprintf* opera igual que *fprintf* pero non mostra a cadea por pantalla, senón que retorna a cadea formateada, para logo facer cousas con ela:

s=sprintf('cadea formato', var1, ..., varn)

- Útil cando se quere manipular cadeas (concatenar con outras, etc.)
- Ex: *s = sprintf('x= %i y= %f\n', x, y);
mensaxe=[s ' ' sprintf('a= %c\n', a)];
disp(mensaxe)*

Estructura de selección básica

- Similar a IF de Fortran
- Avalía unha condición definida polos operadores $>$, $>=$, $<$, $<=$, $==$, $\sim =$

```
if x<=0  
    disp('baixo');  
elseif x<=1  
    disp('medio');  
else  
    disp('alto');  
end
```

- Sentenza IF-ELSE IF:

```
if condición1  
    sentenzas1;  
elseif condición2  
    sentenzas2;  
  
...  
else  
    sentenzasN;  
end
```

Estructura iterativa básica

- Definida:

```
for var = ini:paso:fin  
    sentenzas;  
end
```

```
for k = 1:10  
    fprintf('k= %i\n', k);  
end
```

Similares a *do*
definido e
indefinido e
do-while de
Fortran

- Indefinida:

```
while condición  
    sentenzas;  
end
```

```
n=0; suma=0;  
while n ~= -1  
    n=input('introduce n (-1 para rematar)');  
    suma = suma + n;  
end
```

Remate dun programa

- *return*: remata a execución do programa (ou retorna dende unha función)
- *error('mensaxe')*: remata a execución e mostra a *mensaxe* de erro en cor vermella (que pode estar formateada como con *fprintf*):
 - Ex: *error('erro: x=%i < 0!\n', x)*
- *break*: remata un bucle *for/while* (análogo a *exit* en Fortran) cando se cumpre unha condición
- *break* remata a execución se está fora dun bucle
- Se usas *exit*, remata o Matlab (isto é útil se executas o programa dende a terminal de comandos)