

Ann Zeilinger Karakristi (1921-2016)



- Especialista en criptografía con ordenadores da NSA (axencia de seguridade nacional dos EEUU)
- 2ª guerra mundial (contra Japón) e a guerra fría
- Directora adxunta da NSA (1980)
- Premio de servizos civís distinguidos dos EEUU

Funcións

- Pode estar nun arquivo *.m* co mesmo nome que a función:

```
function [ret1,...,retN] = nome(arg1,...,argM)
%nome: liña de axuda (liña H1)
%texto da axuda
sentenzas da función;
ret1=...;...;retN=...;
end
```

Se so hai un valor devolto,
os corchetes sobran

- Pode chamarse dende calquera programa se o directorio onde se atopa está no path
- *arg1, ..., argM*: argumentos de entrada
- *ret1, ..., retN*: valores devoltos pola función
- A función pode estar tamén logo do programa principal. Esta función é **local**, é dicir, non se pode chamar dende outros arquivos *.m*

```
clear
v=randi(20,1,10);
w=f(v);disp(w)

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

Estructura dunha función

- Cando chamamos a unha función, Matlab búscala no arquivo actual, nos arquivos da carpeta actual e nos arquivos das carpetas do *path*
- Debe haber sentenzas que asignen valores a *ret1*, ..., *retN*
- En **octave**, a función tamén pode estar no programa principal, pero antes da chamada á función
- Se a función comeza na primeira liña do arquivo, éste será considerado un arquivo de función, e non un programa

```
clear

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

v=randi(20,1,10);
w=f(v);disp(w)
```

Exemplo de función

Función que calcule $\frac{1}{n} \sum_{k=1}^n \cos(kx)$ dados x, n

```
function y = f(x, n)
```

```
y = 0;
```

```
for k=1:n
```

```
    y = y + cos(k*x);
```

```
end
```

```
y=y/n;
```

```
%y = mean(cos((1:n)*x));
```

```
end
```

Título da función, variábel devolta e argumentos

Dáselle o valor á variábel devolta

Versión vectorizada

Exemplo de función (varios valores devoltos)

```
function [b x np ni] = funcionf(a)
[nf nc]=size(a);
b = zeros(nf,nc); x = zeros(1,nf);
for i=1:nf
    for j=1:nc
        b(i,j)=a(i,j)*a(j,i); % bij=aijaji
    end
    x(i) = a(i,:)*a(:,i); %prod. escalar fila i por columna i
end
i=mod(a(:),2);
np = sum(i==0); %nº elementos pares
ni = sum(i==1); %nº elementos impares
end
```

Función con varios valores devoltos

- Se non imos empregar algún dos valores devoltos pola función, na chamada á función podemos non almacenalos en ningunha variábel. Ex:
 - Corpo da función:
function [z t] = *calcula*(x,y)
 z=x+y;t=x*y;
end
 - Chamada a función: [~, b]=*calcula*(3,4);
- Neste exemplo, o primeiro valor devolto non se almacena en ningunha variábel.

Chamada á función

- A función pode ser chamada dende a ventá de comandos, dende outra función ou dende un programa (*script*)
- Chamada á función:

$[var1 \dots varN] = f(arg1, \dots, argM);$

Ex: $a = randi(10,4,4);$

$[b \times np \ ni] = f(a);$

- Tamén se pode chamar á función dende unha expresión ou dende a chamada a outra función:

$z = y + f(arg1, \dots, argM);$

$fprintf('y = %f\n', sin(f(arg1, \dots, argM)));$

Función con varios puntos de retorno

- Sentenza *return*: provoca o retorno inmediato da función:

```
function y=le_archivo(fich)
f=fopen(fich,'r');
if -1==f
    y=NaN;return
end
n=fscanf(f,'%g',1);y=zeros(n);
fclose(f);
end
```

- Pode haber varias sentenzas *return* (que poden estar asociadas a diferentes valores retornados) na mesma función.

Función con argumentos opcionais

- Matlab almacena o número de argumentos na variábel *nargin* (non tes que definila).

```
function z=exemplo(x,y)
    if nargin==2
        z=x+y;
    else
        z=2*x;
    end
end
```

```
function z=exemplo(varargin)
    if numel(varargin)==2
        z=varargin(1)+varargin(2);
    else
        z=2*varargin(1);
    end
end
```

- Chamada á función:
z=exemplo(2) ou *z=exemplo(2,3)*
- Alternativa: argumento (vector) chamado *varargin*: tes que comprobar a súa lonxitude dentro da función.

Variáveis globais

- As variáveis dunha función son **locais** (non se coñecen fóra da función)
- Na función non se pode acceder ás variáveis do *workspace* nin doutras funcións
- Para acceder a unha variábel do *workspace* na función, hai que declarala **global** na función:
- Para acceder a unha variábel da función dende fóra (*workspace* ou outra función), hai que poñela como **global** onde se quera usar

```
x=5;y=6;  
z=f(y)
```

```
function z=f(y)  
global x  
z=f+x  
end
```

```
global z  
x=5;  
y=f(x)
```

```
function y=f(x)  
global z  
z=5;y=z+x  
end
```

Paso dunha función como parámetro a *feval*

- Dende a mesma sentenza (que debería estar dentro dun bucle), *feval* permite chamar a unha función distinta en cada iteración do bucle
- O nome da función chamada por *feval* pode ser un carácter ou unha referencia a función (ver máis adiante)

[ret1 ... retM]=feval('nome función', arg1, ..., argN);

```
function y=nome(f,x)
...
y=eval(f, x);
end
```

nesta función o argumento *f* é unha función que en cada chamada a *nome(...)* pode ser distinta

Permite facer o mesmo que as funcións *external* en Fortran

Exemplo de *feval*: integral definida

Cálculo da integral indefinida dun vector de funcións (varias funcións á vez) usando *feval*:

```
clear all
f = {@sin, @cos};
for i = 1:2
    a=0;b=pi;x=a;h=0.001;integral=0;
    for x=a:h:b
        integral=integral+h*feval(f{i}, x);
    end
    fprintf('integral de %s en [%g, %g]= %g\n', char(f{i}),
        a, b, integral);
end
```

Vector de celdas de Matlab

Referencia a función

Transforma a referencia a función nunha cadea de caracteres

Función anónimas

- Funcións simples (dunha única liña, análogas a **funcións de sentenza** en Fortran). Pódense crear en calquer parte (dentro dunha función, arquivo ou liña de comando).
- Definición: $f=@(argumentos) \text{ expresión};$
- Chamada: $f(argumentos)$
- Aínda que non se lles pasen argumentos teñen que levar os parénteses tanto na creación como na chamada. Exemplos:
 - Definición: $f=@(x) x^2+1$, $f=@(x,y) \sin(x+y)$
 - Chamada: $f(4)$, $f(pi/2,pi/3)$

Referencias a función

1) Co **operador** `@` (“at” ou “arroba”), precedendo o operador `@` ó nome dunha función predefinida de Matlab: `f=@sin`

2) Coa función **`str2func('funcion')`** que toma como argumento o nome de función e devolve a referencia á función: `f=str2func('sin'); f(pi)`

Esta función permite transformar unha **expresión** (cadea de caracteres) en referencia a función:

```
expr='x^2';f=str2func(sprintf('@(x) %s',expr))
```

Funcións *inline* (I): obsoletas

- Funcións simples (dunha única liña, análogas a **funcións de sentenza** en Fortran) para cálculos matemáticos que requiren computación extensiva e deben ser eficientes, xa que se executan moitas veces
- Defínense dentro do programa (non nun arquivo separado)

nome = inline('expresión matemática')
nome = inline('expresión', 'var1', ..., 'varN')

- Pode incluír funcións de Matlab ou propias
- As *var1...varN* son as variábeis independentes
- Debe respecta-la dimensión do argumento (escalar / vector/matriz). Se é vector/matriz, hai que facer as operacións compoñente a compoñente (*.**, *.^*, *./*)

Funcións *inline* (II)

- A expresión pode ter varias variábeis independentes (non i ou j , unidade imaxinaria)
- Ex: $f = \text{inline}('exp(x^2)/sqrt(x^2+5)');$
 $f(2)$
 $ans=18.1994$
- Ex: $f=\text{inline}('x^2+y^2+z^2','x','y','z');$
 $f(1,2,1) \Rightarrow \text{resultado: } 6$
- Ex: $f=\text{inline}('exp(-x.^2)./(x.^2+5)')$ *% con vectores*
 $x=-1:0.1:1; f(x)$
- Se as variábeis independentes non se indican, Matlab asume que son as letras da expresión por orde alfabética
- Son equivalentes ás funcións anónimas

Resumen de referencias a función

- Función anónima: $f=@(x) x^2$
- Operador @: $f=@\sin$
- Inline: $f=\textit{inline}('x^2');$
- Con *str2func*: $f=\textit{str2func}('@(x) \sin(x)^2')$
 $s='x*\sin(1+x)'; f=\textit{str2func}(\textit{sprintf}('@(x) \%s',s));$
- Todas poden ser chamadas: $f(5)$
- Agás *inline*, podes obter a súa expresión como cadea de caracteres coa función *func2str*(f):

$\textit{printf}('f=\%s\n',\textit{func2str}(f))$