

Selection programming:

- A **selection statement** allows a question to be asked or a condition to be tested to determine which steps are to be performed next.
- The question or condition is defined using **relational** and **logical operators**, which will be described prior to introducing the selection statement.

Operatori relazionali e logici (help ops)

For relational and logical expressions:

Inputs: True is any nonzero number
False is 0 (zero)

Outputs: True is 1 (one)
False is 0 (zero)

An output array variable assigned to a relational or logical expression is identified as logical. That is, the result contains numerical values 1 and 0, that can be used in mathematical statements, but also allow logical array addressing.

Operatori relazionali

Relational Operator	Description
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal
~=	not equal

Per confrontare:

- **due arrays della stessa dimensione**
- **un elemento di array con uno scalare**
- **un array con uno scalare (confronto elemento-per-elemento)**

esempi

```
>> A=1:9, B=8-A
```

```
A =
```

```
    1    2    3    4    5    6    7    8    9
```

```
B =
```

```
    7    6    5    4    3    2    1    0   -1
```

```
>> tf1 = A <=4
```

```
tf1 =
```

```
    1    1    1    1    0    0    0    0    0
```

```
>> tf2 = A > B
```

```
tf2 =
```

```
    0    0    0    0    1    1    1    1    1
```

```
>> tf3 = (A==B)
```

```
tf3 =
```

```
    0    0    0    1    0    0    0    0    0
```

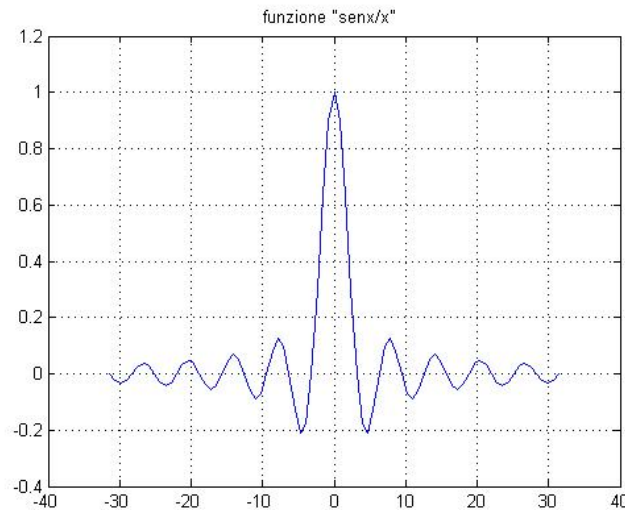
```
>> tf4 = B-(A>2)
```

```
tf4 =
```

```
    7    6    4    3    2    1    0   -1   -2
```

esempio

```
%% calcolo di  $\sin x/x$   
%% con eliminazione del valore indefinito, per  $x=0$ ,  
%% e grafico del risultato  
x = (-10:.25:10)*pi;  
x = x + (x==0)*eps; % per  $x=0$ , il valore di x diventa:  $x=eps$   
y = sin(x)./x;  
figure, plot(x,y), axis([-40 40 -0.4 1.2]), grid on,...  
title("funzione " $\sin x/x$ " ");
```



Operatori logici

Logical operators provide a way to combine or negate relational expressions.

Logical Operator	Description
&	and
	or
~	not

A fourth logical operator is implemented as a function:

`xor(A,B)` Exclusive or: Returns ones where either A or B is True (nonzero); returns False (zero) where both A and B are False (zero) or both are True (nonzero).

Operatori logici

Definitions of the logical operators, with 0 representing False and 1 representing True:

A	B	$\sim A$	$A B$	$A \& B$	$\text{xor}(A, B)$
0	0	1	0	0	0
0	1	1	1	0	1
1	0	0	1	0	1
1	1	0	1	1	0

The precedence from highest to lowest is relational operators, followed by logical operators \sim , $\&$, and $|$. Parentheses can be used to change the precedence and should be used liberally to clarify the operations.

esempi

```
>> A=1:9
```

```
A =
```

```
     1     2     3     4     5     6     7     8     9
```

```
>> tf1 = A>4
```

```
tf1 =
```

```
     0     0     0     0     1     1     1     1     1
```

```
>> tf2 = ~(A>4)
```

```
tf2 =
```

```
     1     1     1     1     0     0     0     0     0
```

```
>> tf3 = (A>2)&(A<6)
```

```
tf3 =
```

```
     0     0     1     1     1     0     0     0     0
```

```
>> tf4 = xor((A>2),(A<6))
```

```
tf4 =
```

```
     1     1     0     0     0     1     1     1     1
```


esempio

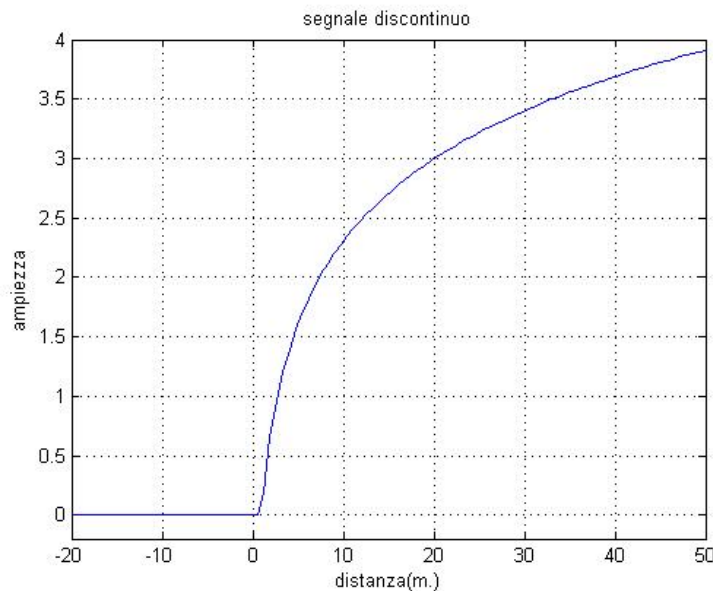
% implementazione di un segnale discontinuo

```
x = linspace(-20,50,100);
```

```
y_1 = zeros(1,100);
```

```
y = y_1 + log(x).*(x>1);
```

```
figure, plot(x,y), axis([-20 50 -0.2 4]), xlabel('distanza(m)'), ...  
    ylabel('ampiezza'),title('segnale discontinuo'), grid on;
```



Funzioni relazionali e logiche

Function	Description
<code>any(x)</code>	Returns a scalar that is 1 (true) if <i>any</i> element in the vector <code>x</code> is nonzero; otherwise, the scalar is 0 (false). Returns a row vector containing a 1 (true) in each element for which any element of the corresponding column of matrix <code>x</code> is nonzero, and a 0 (false) otherwise.
<code>all(x)</code>	Returns a scalar that is 1 (true) if <i>all</i> elements in the vector <code>x</code> are nonzero; otherwise, the scalar is 0 (false). Returns a row vector containing a 1 (true) in each element for which all elements of the corresponding column of matrix <code>x</code> are nonzero, and a 0 (false) otherwise.
<code>find(x)</code>	Returns a vector containing the indices of the nonzero elements of a vector <code>x</code> . Returns a vector containing the indices of the nonzero elements of <code>x(:)</code> , which is a single-column vector formed from the columns of matrix <code>x</code> .
<code>isnan(x)</code>	Returns an array with ones where the elements of <code>x</code> are NaN and zeros where they are not.
<code>isfinite(x)</code>	Returns an array with ones where the elements of <code>x</code> are finite and zeros where they are not. For example, <code>isfinite([pi NaN Inf -Inf])</code> is <code>[1 0 0 0]</code> .
<code>isinf(x)</code>	Returns an array with ones where the elements of <code>x</code> are <code>+Inf</code> or <code>-Inf</code> and zeros where they are not.
<code>isempty(x)</code>	Returns 1 if <code>x</code> is an empty array and 0 otherwise.

Comandi per controllo di flusso

(help lang)

- **Simple if Statement**
- **Nested if Statement**
- **else and elseif Clauses**
- **Switch Selection Structure**
- **Loops (prossima lezione)**

Simple if Statement

```
if    logical expression
      commands
end
```

Esempio:

```
if d < 50
    count = count + 1;
    disp(d);
end
```

Nested if Statement

```
if d < 50
    count = count + 1;
    disp(d);
    if b > d
        b = 0;
    end
end
end
```

***else* Clause**

else clause: allows one set of statements to be executed if a logical expression is true and a different set if the logical expression is false.

```
if interval < 1
    xinc = interval/10;
else
    xinc = 0.1;
end
```

***elseif* clause**

```
if temperature > 100
    disp('Too hot - equipment malfunctioning.')
```

elseif temperature > 90

```
    disp('Normal operating range.')
```

elseif temperature > 50

```
    disp('Below desired operating range.')
```

else

```
    disp('Too cold - turn off equipment.')
```

end

Switch Selection Structure

The syntax is

```
switch expression
  case test expression 1
    commands
  case {test expression 2, test expression 3}
    commands
  .
  .
  .
  otherwise
    commands
end
```


esempio

```
% esempio utilizzo del comando "Switch"  
% converte la misura in metri  
x = input('valore di distanza x: ');  
unit = input('unità di misura (ft,in,m,cm,mm): ','s');  
switch unit  
    case 'in'  
        y = x*0.0254;  
        disp(['x = ' num2str(y) ' metri']);  
    case 'ft'  
        y = x*0.3048;  
        disp(['x = ' num2str(y) ' metri']);  
    case 'm'  
        y = x;  
        disp(['x = ' num2str(y) ' metri']);  
    case 'cm'  
        y = x/100;  
        disp(['x = ' num2str(y) ' metri']);  
    case 'mm'  
        y = x/1000;  
        disp(['x = ' num2str(y) ' metri']);  
    otherwise  
        disp(['unità di misura sconosciuta: ' unit]);  
end %fine del comando "switch"
```