A brief introduction to
MATLAB (and Octave)
for electrical communications
(Part 2)

Corso di Laboratorio Interdisciplinare II

if, then and elseif

The general form of the `if` statement is:

```
if <expression>
    <statement>, <statement>,…
elseif <expression>
     <statement>, <statement>,…
else
    <statement>, <statement>,…
end
```

Example

```
a=3;
b=floor(5*rand(1,1);
if a>b
    fprintf('a=%d larger than b=%d/n',a,b);
elseif a==b
    fprintf('a=%d equal to b=%d/n',a,b);
else
    fprintf('a=%d smaller than b=%d/n',a,b);
end
```

For and While

```
for <variable=expression>
<statement>, <statement>,…
end
```

```
while <expression>
<statement>, <statement>,…
end
```

```
N=5;
for i=1:N
    for j=1:N
        A(i,j)=1/(i+j-1);
    end
end
```

```
b=0;
a=10;
while (a>3)
    b=b+1;
    a=a-b;
end
```

Function “rand”

- `rand` uniformly distributed random numbers.
- `rand(N)` is an N-by-N matrix with random entries, chosen from a uniform distribution on the interval (0.0,1.0).
- `rand(M,N)` is a M-by-N matrix with random entries on the same interval.

```
octave-3.2.3:8> V=rand(3)
V =

   0.885006   0.985149   0.193368
   0.060968   0.912635   0.719775
   0.894609   0.040091   0.480421
```

```
octave-3.2.3:9> W=rand(1,3)
W =

   0.50589   0.70535   0.15719
```

Function “zeros”

- `zeros` zeros array.
- `zeros(N)` is an N-by-N matrix of zeros.
- `zeros(M,N)` is an M-by-N matrix of zeros.

```
octave-3.2.3:8> V=zeros(4)
V =

   0   0   0   0
   0   0   0   0
   0   0   0   0
   0   0   0   0
```

```
octave-3.2.3:9> W=zeros(1,3)
W =

   0   0   0
```

Function “ones”

- `ones` Ones array.
- `ones(N)` is an N-by-N matrix of ones
- `ones(M,N)` is an M-by-N matrix of ones.

```
octave-3.2.3:8> V=ones(4)
V =

   1   1   1   1
   1   1   1   1
   1   1   1   1
   1   1   1   1
```

```
octave-3.2.3:9> W=ones(1,3)
W =

   1   1   1
```

Function “find”

- `find` Finds indices of nonzero elements.
- `I = find(X)` returns the indices of the vector `X` that are non-zero.
- Note that `X` can be the result of the evaluation of an expression
- Example:

```
octave-3.2.3:17> A=floor(200*rand(1,10))
A =
    83   142    81    69   119     3    36    87    10    88

octave-3.2.3:18> X=A>100
X =
   0   1   0   0   1   0   0   0   0   0

octave-3.2.3:19> I=find(X)
I =

   2   5

octave-3.2.3:20>
```

Function “length”, “max” e “min”

`length`: Length of vector.

- For a vector, `length(X)` returns the number of elements in `X`.
- For a matrix NxM, `length(X)` returns the largest dimension between N and M.

`max`: Largest component.

- For a vector, `max(X)` returns the largest element in `X`.
- For a matrix, `max(X)` returns a row vector containing the largest element of each column in `X`.

`min`: Smallest component.

- For a vector, `min(X)` returns the smallest element in `X`.
- For a matrix, `min(X)` returns a row vector containing the smallest element of each column in `X`.

Function "sort" (1/2)

`sort`: sorts the elements of a vector in ascending or descending order

- For a vector, `sort(X)` sorts the elements of `X` in ascending order.
- For a matrix, `sort(X)` sorts the elements of each column of `X` in ascending order.

```
>> V=[7 5 9 2 4];
>> sort(V)
ans =
     2     4     5     7     9

>>
```

```
>> A=[7 5 2; 4 3 5; 9 8 3]
A =
     7     5     2
     4     3     5
     9     8     3

>> sort(A)
ans =
     4     3     2
     7     5     3
     9     8     5
>>
```

Function "sort" (2/2)

The default behavior of `sort` can be modified with additional inputs

- `sort(A,dim)` sorts the elements of a matrix `A` in ascending order by dimension dim (dim=1: columns *(default)*, dim=2: rows).
- `sort(A,'descend')` sorts each column of a matrix `A` in descending order
- `sort(A,2,'descend')` sorts each row of a matrix `A` in descending order

```
>> A=[0 23 12; 5 3 6]
A =
    0   23   12
    5    3    6

>> sort(A,2)
ans =
    0   12   23
    3    5    6

>>
```

```
>> A=[0 23 12; 5 3 6]
A =
    0   23   12
    5    3    6

>> sort(A,2,'descend')
ans =
    23   12   0
    6    5    3

>>
```

Function “round”, “ceil” e “floor”

- `round`: Round towards nearest integer.
- `floor`: Round towards the integer immediately lower.
- `ceil`: Round towards plus the integer immediately higher.

```
octave-3.2.3:30> test= [0.4 0.7]
test =

   0.40000   0.70000

octave-3.2.3:31> round(test)
ans =

   0   1

octave-3.2.3:32> floor(test)
ans =

   0   0

octave-3.2.3:33> ceil(test)
ans =

   1   1

octave-3.2.3:34>
```

Function "size"

`size`: Size of each dimension of a vector/matrix

- For vectors, same as `length`
- For matrices, different behavior

```
>> A=[7 5 2; 4 3 5;]
A =

     7     5     2
     4     3     5

>> size(A)
ans =

     2     3

>> length(A)
ans =

     3

>> size(A,1)
ans =

     2
```

Function "sum"

`sum`: Sum of elements of a vector/matrix

- For vectors, it returns the sum of all elements
- For matrices, it returns a row vector containing the sums of the elements of each row
- Behavior for matrices can be changed as seen for the `sort` command

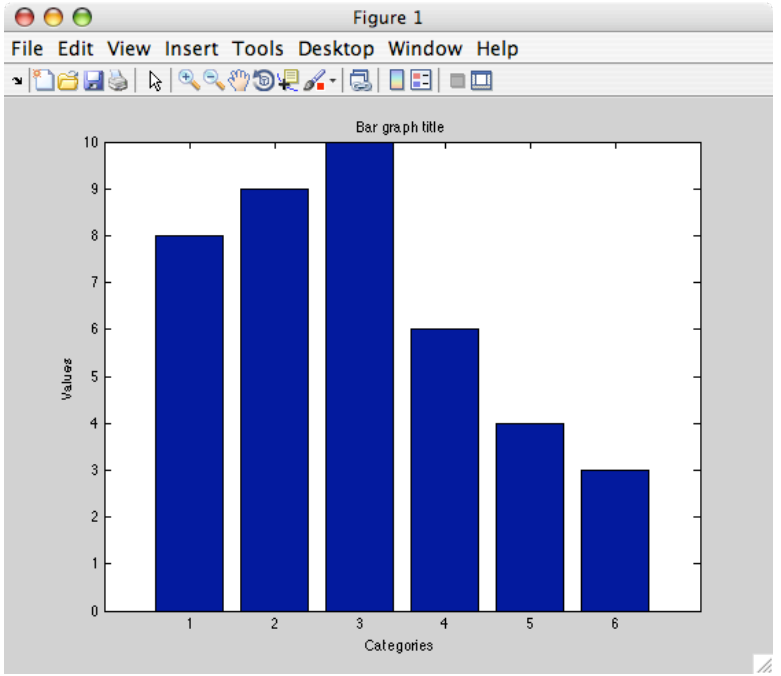
```
>> V=[1 2 3]
V =
     1     2     3

>> sum(V)
ans =
     6

>> A=[1 2 3; 2 4 6]
A =
     1     2     3
     2     4     6

>> sum(A)
ans =
     3     6     9

>> sum(A,1)
ans =
     3     6     9

>> sum(A,2)
ans =
     6
    12

>>
```

Function “bar”

`bar`: Bar graph

- Typically used when the values are not samples of a function

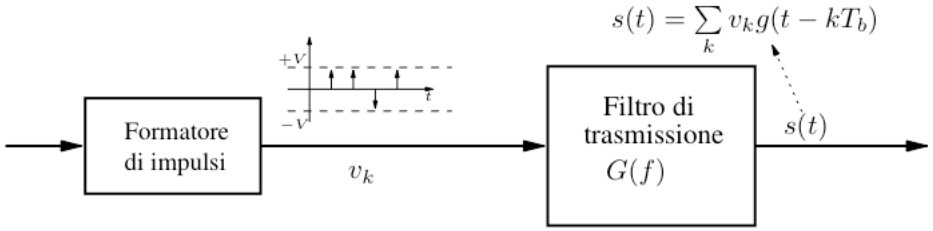
```
>> V=[8 9 10 6 4 3];
>> bar(V)
>> title('Bar graph title')
>> xlabel('Categories')
>> ylabel('Values')
>>
```



Example of application of Matlab to electrical communications

Eye diagram

- The eye diagram is useful in analyzing the performance of a numerical link using a baseband impulsive modulation:



- The samples at the receiver (neglecting noise) can be written as follows:

$$s(nT_b) = \sum_k v_k g(nT_b - kT_b) = v_n g(0) + \sum_{\substack{k \\ k \neq n}} v_k g(nT_b - kT_b)$$

- Leading to the Nyquist conditions in the time domain:

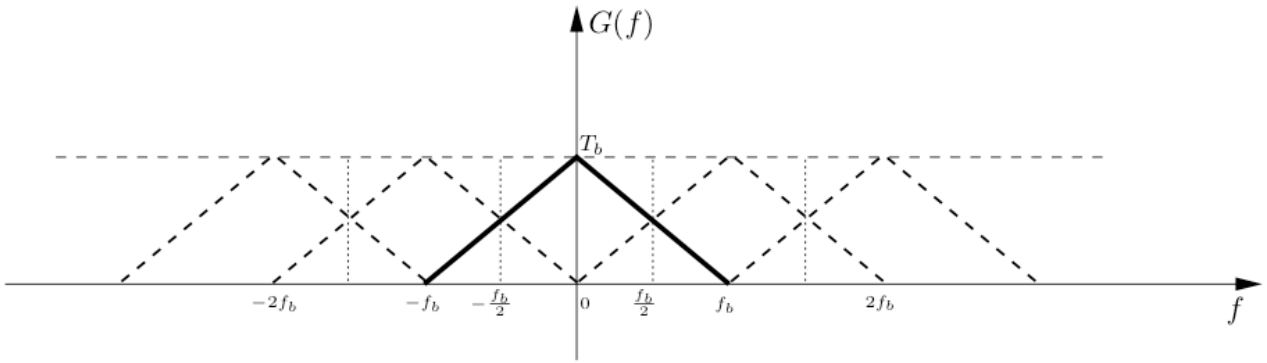
$$g(jT_b) = \begin{cases} 1 & j = 0 \\ 0 & j \neq 0 \end{cases}$$

- Which translate in the following condition in the frequency domain:

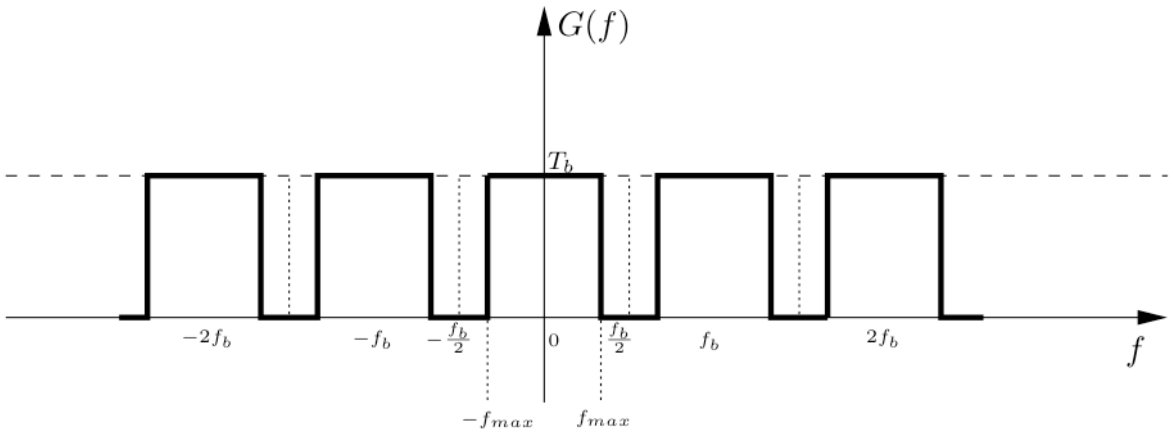
$$\sum_k G(f - kf_b) = T_b$$

Eye diagram

- Example of G(f) meeting the Nyquist condition:



- Example of G(f) <u>NOT</u> meeting the Nyquist condition:

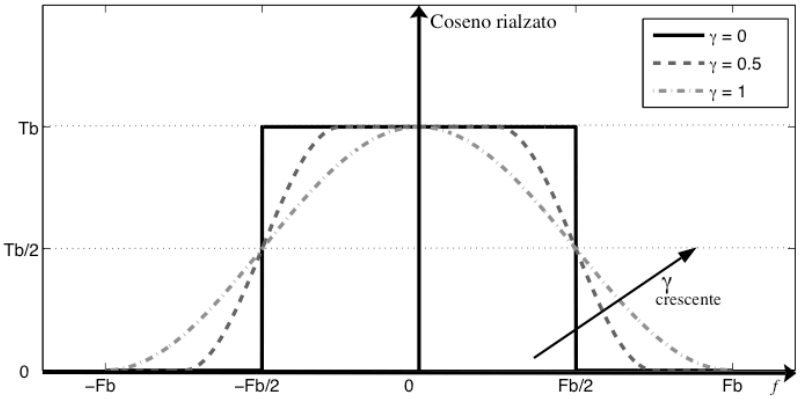


Eye diagram

- A typical family of impulse responses that meet the Nyquist criterion is the raised cosine one:

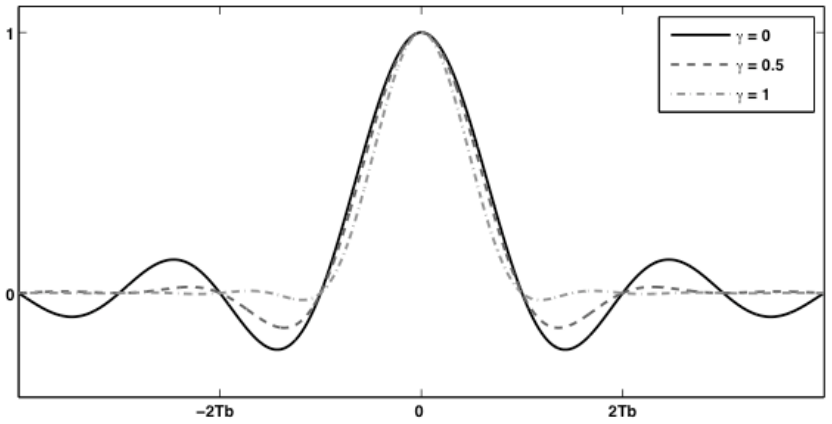
$$G(f) = \begin{cases} T_b & 0 \le |f| \le \frac{f_b}{2}(1-\gamma) \\ \frac{T_b}{2}\left(1+\cos\left(\frac{\pi T_b}{\gamma}\left(|f| - \frac{f_b}{2}(1-\gamma)\right)\right)\right) & \frac{f_b}{2}(1-\gamma) < |f| \le \frac{f_b}{2}(1+\gamma) \\ 0 & |f| > \frac{f_b}{2}(1+\gamma) \end{cases}$$

- The shape of G(f) and g(t) as a function of the roll-off is the following:



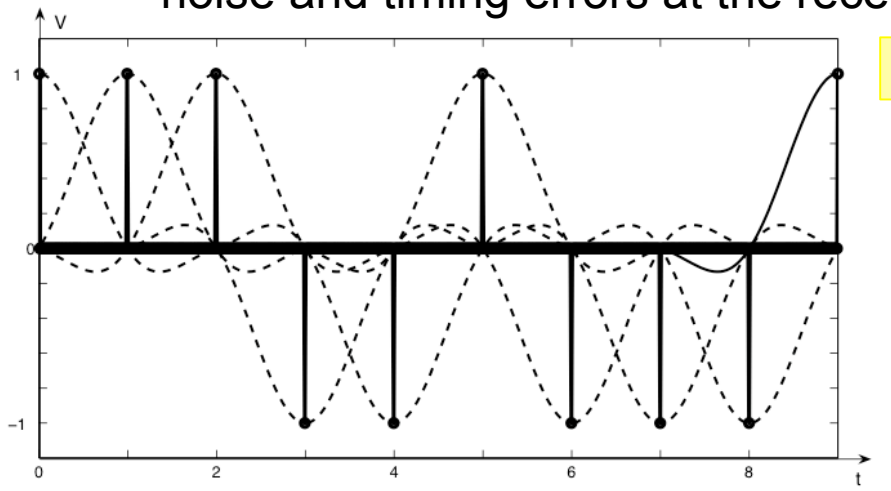
G(f)

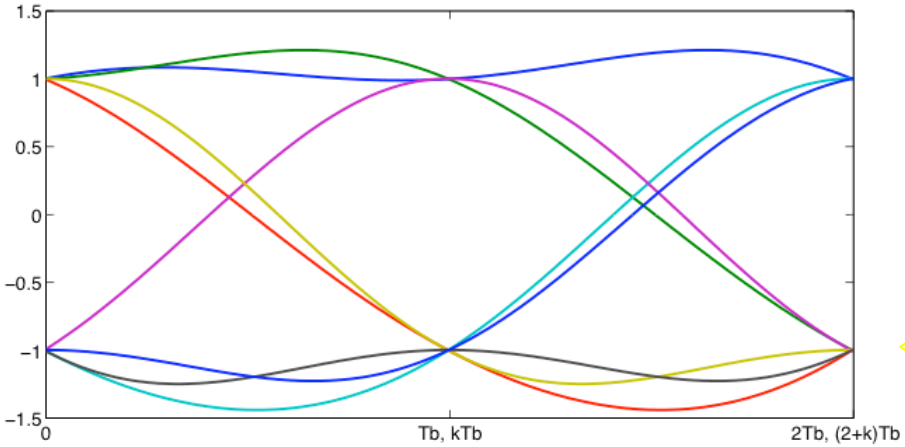
g(t)

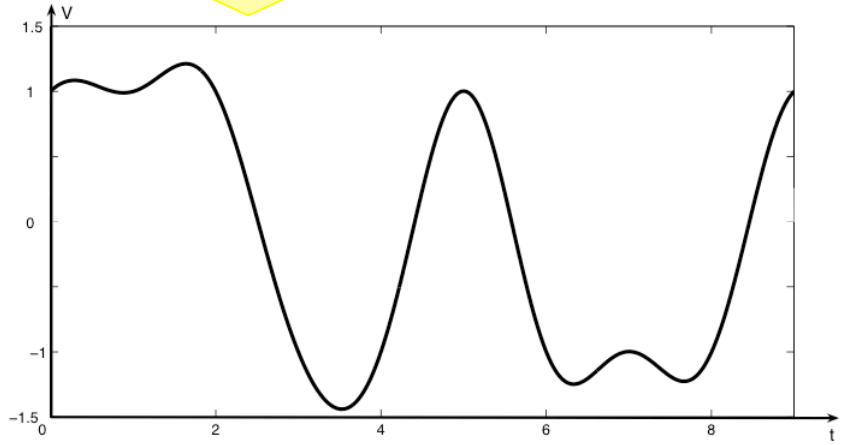


Eye diagram

- The performance of such a system can be analyzed by drawing the eye diagram, which provides a measure of the robustness of the system to noise and timing errors at the receiver:







Eye diagram

- Implementation in Matlab:
  - Main script **eyeDiagram.m**
  - Function **generateLevelSequence.m**
  - Function **pulseGenerator.m**
  - Function **noiseSet.m**

- Code available at:

http://newyork.ing.uniroma1.it/~lucadn/courses/labint/20110112_LI_matlab_code.zip

Esercizio (45 minuti circa)

Si chiede di produrre uno script Matlab/Octave in grado di manipolare un numero **num** di sequenze di numeri interi positivi (incluso lo 0) come indicato nei seguenti quesiti:

1) Si costruisca una matrice `A` contenente le **num** sequenze, una per riga, sapendo che ogni sequenza deve avere lunghezza pari a **lseq** e può contenere valori compresi tra **0** e **valmax**. Si definisce come energia di una sequenza la somma dei quadrati degli elementi che compongono la sequenza. Sulla base di tale definizione, si stampi su schermo il valore di energia massimo tra quelli delle sequenze presenti nella matrice **A** e la sequenza a cui risulta associato (ad esempio: "*il valore di energia massimo è **Emax** e caratterizza la riga **x** della matrice*").
   N.B.: Si faccia uso a questo proposito di una funzione specifica che prenda in ingresso un vettore e restituisca il valore della sua energia.
2) Si crei un vettore contenente le energie delle **lseq** sequenze rappresentate dalle *colonne* di `A` e lo si ordini in senso crescente: se ne stampino infine su schermo i valori.
3) Si crei un grafico usando la funzione `bar` in modo da rappresentare un istogramma che rappresenti in ordinata le energie relative alle sequenze corrispondenti alle colonne di A (le stesse ricavate e ordinate al punto 2)) con il vincolo di rappresentare unicamente le energie con valore superiore alla media delle energie delle colonne di A. In ascissa è sufficiente inserire l'indice del valore rappresentato all'interno del vettore creato al punto 2)

N.B.: si assumano i seguenti valori; **num**=3, **lseq**=10, **valmax**= 20

Script “threshscript” (Esercizio per casa)

- Si chiede di produrre uno script che generi **100** valori diversi tra **0** e **100** e calcoli le percentuali di superamento di due differenti soglie. Dati il valore **minimo** e quello **massimo** tra i 100 generati:
  - la prima soglia supera il valore minimo di una quantità pari a **3/5** della differenza tra massimo e minimo
  - la seconda soglia supera il valore minimo di una quantità pari a **4/5** della differenza suddetta.

  - Le due soglie devono essere generate da una funzione che prenda in ingresso i valori minimo e massimo e restituisca le soglie stesse (una funzione che genera più di un output ha una dichiarazione del tipo :

    function [out1, out2] = func (a, b)
  - Lo script deve utilizzare la funzione di cui al punto precedente e produrre un grafico che visualizzi i campioni in gioco e le due soglie generate.
  - Si chiede di produrre lo script 2 volte: la prima usando cicli e if statements, la seconda utilizzando la funzione **find**

Reference material

- Slides available at:

  http://newyork.ing.uniroma1.it/~lucadn/teaching.html