Politecnico di Milano

Segnali per le Telecomunicazioni
2014

Matlab

Analisi dei codici matlab presentati nel corso del semestre

Autori
Badini Federico
mat. 816820 - federico.badini@mail.polimi.it
Bodini Stefano
mat. 817615 - stefano2.bodini@mail.polimi.it

Docente
Claudio Prati

Contents

1 | Serie di Fourier

Il seguente codice Matlab richiede all'utente di indicare il numero di armoniche con le quali si vuole andare a ricostruire un segnale di tipo onda quadra. Il programma procede step by step all'aggiunta delle armoniche mostrando la via via maggior precisione raggiunta nella ricostruzione.

Codeblock 1: Serie di Fourier

```
% PROGRAMMA SerieFourier

clear all
clf
t=[-10:.01:10];
T0=5;
N=length(t);

NA=input('numero armoniche per ricostruire il segnale? ');

c0=1/2;
c = zeros(1,NA)

for n=1:NA;
c(n)=sin(pi*n/2)/(pi*n);
end

x0=ones(1,N)*1/2;
x=zeros(1,N);
figure(1)
clf
for n=1:NA;
    x=x+2*c(n)*cos(2*pi*n/T0*t);
    subplot(211)
    plot(t,x0+x)
    xlabel('tempo in secondi')
    subplot(212)
    stem(n,2*c(n),'filled')
    xlabel('armoniche')
    hold on
    pause
end
```

Per una migliore comprensione del codice sopra descritto si fornisce ora un breve riassunto delle principali funzioni che compaiono nel codice. I comandi "clear all" e "clf" servono per resettare il workspace e le finestre. Alla riga 5 compare la sintassi per la definizione di un array con lo scopo di discretizzare l'asse dei tempi (sintassi t = [estremo inferiore:step:estremo superiore]). Con "T0" facciamo riferimento al periodo del segnale

onda quadra. “NA” rappresenta il numero di armoniche: come si può notare alla riga 9 il comando input permette l’inserimento del valore desiderato da tastiera. L’istruzione alla riga 12 e’ stata inserita per evitare un errore (non bloccante) in fase di esecuzione. “c(n)” rappresenta l’array dei coefficienti, ricalcolati di volta in volta all’interno del ciclo for alla riga 14, mentre “x” rappresenta il vettore somma che accumula passo passo i contributi delle varie armoniche. L’operazione di somma delle singole armoniche viene svolta nel codice all’istruzione 23. Le istruzioni “figure” aprono un nuovo pannello entro cui poter disegnare il grafico, operazione effettuata tramite le istruzioni plot alla riga 25 o stem, alla riga 28. La differenza tra queste due istruzioni è che mentre la prima collega i punti nel piano fra loro, la seconda collega i punti nel piano all’asse delle ascisse. Le istruzioni di subplot dividono la figure in due parti cosi da permettere di inserire due grafici in un solo pannello.

2 | Convoluzione

Il seguente programma calcola la convoluzione tra una cosinusoide di frequenza normalizzata f, indicata dall’utente, e una risposta impulsiva i cui campioni sono inseriti nel vettore h

Codeblock 2: Convoluzione

```
% PROGRAMMA Convoluzione

clear all
figure(1)
clf
n=(-50:50);
f=input('frequenza normalizzata del coseno? ');
x=cos(2*pi*f*n);
stem(n,x)
hold on
h=[-1/4 1/2 -1/4];
y=conv(x,h);
pause
m=(-51:51);
stem(m,y,'r')
hold off
```

Nelle prime 10 righe il programma svolge operazioni di cui si è già data spiegazione nella sezione precedente. Si noti tuttavia la sintassi compatta per la definizione di un vettore con step = 1 alla riga 6. Il programma in sostanza riceve in input la frequenza normalizzata del segnale cosinusoidale x, inizializzato alla riga 8. Procede poi al calcolo del valore y dell’uscita tramite convoluzione con il vettore a tre elementi h. Tale vettore rappresenta un esempio di filtro passa alto. Per valori di frequenza normalizzata prossimi a 0 il segnale viene abbattuto (non si faccia caso al rumore alle estremita’). Al contrario

un segnale con frequenza normalizzata prossima a 0,5 sarebbe rilevabile in uscita senza variazioni. Se la risposta in frequenza all'impulso fosse stata [1/4 1/2 1/4] ci saremmo al contrario trovati di fronte ad un filtro passa basso. Dal punto di vista del codice si noti la versione a tre parametri della funzione stem alla riga 15, dove l'ultimo parametro indica il colore con cui disegnare il grafo.

3 | Chirp

Il seguente programma Esegue la convoluzione tra un chirp (coseno linearmente modulato in frequenza) ed una replica del chirp ribaltata nel tempo.Il risultato è un segnale di tipo impulsivo. Il programma permette anche di ascoltare il suono del "chirp".

Codeblock 3: Chirp

```
% PROGRAMMA Chirp

clear all
figure(1)
clf
figure(2)
clf
n=(1:10000);
x=cos(2*pi*n.^2/40000);
w=fir1(40,[.1 .9],'bandpass');
c=conv(x,w);
chirp=c(41:10000-41);
h=fliplr(chirp);

subplot(211)
plot(chirp)
title('chirp')
display('press for zooming')
pause
axis([500 1500 -1 1])
display('press for playing')
pause
soundsc(chirp,22050)
subplot(212)
plot(h,'r')
title('time reversed chirp')
display('press for zooming')
pause
axis([8000 9500 -1 1])
display('press for chirp compression')
pause
figure(2)
y=conv(chirp,h);
```

Codeblock 4: Chirp

```
plot(y)
display('press for zooming')
pause
axis([9900 9940 -1200 4000])
```

Le prime 7 righe del programma contengono le operazioni preliminari di reset e creazione dei nuovi pannelli. Alla riga 9, la notazione ".^" indica che si vuole elevare ad esponente 2 ogni membro del vettore n. La funzione fir1 crea un filtro passa banda definito con una precisione di 40 coefficienti. 0.1-0.9 rappresentano le frequenze di cut-off del filtro passa banda. Il segnale cosinusoidale viene convoluto con il filtro passa banda per produrre il chirp. Il chirp ribaltato viene ottenuto alla riga 13 tramite la funzione fliplr. Dopodichè il programma procede a mostrare i segnali ottenuti e il risultato della convoluzione del chirp con il chirp ribaltato. Tale convoluzione ha la particolarità di produrre un segnale di tipo impulsivo. Si adotta questo stratagemma per la trasmissione di impulsi al fine di aggirare l'ostacolo rappresentato dall'enorme potenza che sarebbe necessaria per produrre un impulso in grado di viaggiare a grande distanza.

Si ricorda che per visionare la documentazione relativa alle funzioni presenti nei programmi è sufficiente digitare "help «nome-funzione»" nella linea di comando di Matlab

4 | Risposta in frequenza

Il seguente programma visualizza N valori di modulo e fase della risposta in frequenza di h (segnale rettangolare da 0 a T secondi con ampiezza unitaria) in corrispondenza delle frequenze f. La risposta in frequenza e' calcolata come modulo e fase della risposta ad esponenziali complessi con differenti frequenze.

N.B. L'integrale di convoluzione è approssimato con una somma di convoluzione moltiplicata per l'intervallo di campionamento (piccolo) dt.

Codeblock 5: Risposta in frequenza

```
% PROGRAMMA Risposta in frequenza

clear all
figure(1)
clf
Tmax=10;
dt=0.01;
T=.5;
t=[0:dt:Tmax];
h=ones(1,T/dt);
N=500;
```

Codeblock 6: Risposta in frequenza

```
for k=1:N+1;
    f=(k-N/2)*1/100;
    x=exp(j*2*pi*f*t);
    y=conv(x,h)*dt;
    fr(k)=f;
    modulo(k)=abs(y(Tmax/2/dt));
    fase(k)=angle(y(Tmax/2/dt).*conj(x(Tmax/2/dt)));
end
figure(1)
subplot(211)
plot(fr,modulo)
title('Modulo')
xlabel('Frequenza')
grid
subplot(212)
plot(fr,fase)
title('Fase')
xlabel('Frequenza')
grid
```

Il programma contiene un preambolo che realizza il reset e prepara il contesto dell'esecuzione allocando le variabili Tmax, dt, T, il vettore t ed N. Il cuore del programma è il ciclo alla linea 12, che realizza ad ogni iterazione la convoluzione tra il segnale rettangolare h e l'esponenziale x, la cui frequenza viene ricalcolata ed aggiornata ad ogni iterazione in modo da spaziare nel nostro caso nell'intervallo (-2.5, 2.5).

5 | Media mobile

Il seguente programma esegue una media mobile di 1000 campioni su una sequenza di N campioni costituita da una cosinusoide a bassa frequenza e da una retta a pendenza leggermente negativa. La media mobile e' eseguita tramite convoluzione con una finestra rettangolare di 1000 campioni di valore 1/1000.

Codeblock 7: Media mobile

```
% PROGRAMMA Media mobile

clf
clear all
N=11000;
 n=1:N;
x=0.15*cos(2*pi*n/7000)-n/20000+randn(1,N);
plot((1:N-1000),x(501:N-500))
h=ones(1,1000)/1000;
y=conv(x,h);
```

Codeblock 8: Media mobile

```
pause
hold on
plot((1:N-1000),y(1001:N),'r')
pause
axis([1 N-1000 -.8 .2])
```

Le istruzioni iniziali svolgono il set up; alla linea 7 il codice presenta la creazione del vettore di campioni x, costituito da un coseno con una tendenza verso il basso data dal termine -n/20000. Il segnale x presenta anche per ogni campione una componente randomica di valore incluso tra 1 ed N. Il programma procede poi alla stampa del segnale x e al calcolo della convoluzione tra il segnale x e il rettangolo di ampiezza unitaria h, costituito da 1000 campioni. Il risultato della convoluzione è il segnale y, il quale come detto rappresenta la media mobile del segnale. E questo poichè il rettangolo h ha ampiezza 1/1000 e pertanto la convoluzione produce in ogni punto dell'asse dei tempi un valore mediato sui vicini 1000 campioni.

6 | Modulazione IQ

Il seguente programma esegue la modulazione e demodulazione in fase e quadratura di 2 segnali gaussiani x1(t) e x2(t) con ampiezze il primo il doppio del secondo. I segnali modulati coseno e seno (y1(t) e y2(t)) sono sommati tra loro a formare il segnale modulato IQ z(t). Il segnale z(t) viene demodulato coseno (z1(t)) e poi seno (z2(t)). La componente a bassa frequenza di z1(t) e z2(t) coincide con i 2 segnali iniziali gaussiani x1(t) e x2(t). ATTENZIONE i segnali nel dominio delle frequenze sono ottenuti moltiplicando i risultati della DFT per l'intervallo di campionamento dt. In pratica si è approssimato l'integrale dela trasformata di Fourier con una scalinata ottenuta utilizzando un intervallo di campionamento molto piccolo.

Codeblock 9: Modulazione IQ

```
% PROGRAMMA Modulazione IQ

clear all
close all
dt=.0002;
N=2000;
fo=200;
t=(-N:N-1)'*dt;
df=1/(dt*(2*N));
f=(-N:N-1)'*df;
display('SEGNALE x1(t) A BASSA FREQUENZA DA MODULARE COSENO')
figure(1)
clf
xx1=(gausswin(2*N+1,20));
```

Codeblock 10: Modulazione IQ

```
x1=xx1(1:2*N);
plot(t,x1)
axis([-dt*2*N/10 dt*2*N/10 -0.5 1.5])
grid on
xlabel('time (seconds)')
title('x1(t)')

display('PREMERE PER VISUALIZZARE LA TRASFORMATA DI FOURIER (reale) X1(f)')
pause

figure(2)
clf
X1=fft(fftshift(x1));
plot(f,fftshift(real(X1))*dt)
axis([-3*fo 3*fo -100*dt 300*dt])
grid on
xlabel('frequency (Hz)')
title('Real part of X1(f)')

display('PREMERE PER VISUALIZZARE SEGNALE x2(t) A BASSA FREQUENZA DA MODULARE SENO')
pause

figure(1)
hold on
xx2=.5*(gausswin(2*N+1,20));
x2=xx2(1:2*N);
plot(t,x2,'r')
axis([-dt*2*N/10 dt*2*N/10 -0.5 1.5])
xlabel('time (seconds)')
title('x1(t) and x2(t)')
hold off

display('PREMERE PER VISUALIZZARE LA TRASFORMATA DI FOURIER (reale) X2(f)')
pause

figure(2)
hold on
X2=fft(fftshift(x2));
plot(f,fftshift(real(X2))*dt,'r')
axis([-3*fo 3*fo -100*dt 300*dt])
xlabel('frequency (Hz)')
title('Real part of X1(f) and X2(f)')
hold off
```

Codeblock 11: Modulazione IQ

```
display('PREMERE PER VISUALIZZARE y1(t)=x1(t)*cos(2*pi*t*fo)')
pause
y1=x1.*cos(2*pi*t*fo);
figure(3)
plot(t,y1)
axis([-dt*2*N/10 dt*2*N/10 -1.5 1.5])
grid on
xlabel('time (seconds)')
title('y1(t)=x1(t)*cos(2*pi*t*fo)')

display('PREMERE PER VISUALIZZARE y2(t)=x2(t)*sin(2*pi*t*fo)')
pause
y2=x2.*sin(2*pi*t*fo);
figure(3)
hold on
plot(t,y2,'r')
xlabel('time (seconds)')
title('y1(t)=x1(t)*cos(2*pi*t*fo) and y2(t)=x2(t)*sin(2*pi*t*fo)')
axis([-dt*2*N/10 dt*2*N/10 -1.5 1.5])

display('PREMERE PER VISUALIZZARE LA TRASFORMATA DI FOURIER (reale)
    Y1(f)')
pause

figure(4)
Y1=fft(fftshift(y1));
Y2=fft(fftshift(y2));
plot(f,fftshift(real(Y1))*dt)
axis([-3*fo 3*fo -100*dt 200*dt])
grid on
xlabel('frequency (Hz)')
title('Real part of Y1(f)')
hold on

display('PREMERE PER VISUALIZZARE LA TRASFORMATA DI FOURIER
    (immaginaria) Y2(f)')
pause

plot(f,fftshift(imag(Y2))*dt,'r')
axis([-3*fo 3*fo -100*dt 200*dt])
xlabel('frequency (Hz)')
title('Real part of Y1(f) and Imaginary part of Y2(f)')

display('PREMERE PER VISUALIZZARE z(t)=y1(t)+y2(t)')
pause

z=y1+y2;
```

Codeblock 12: Modulazione IQ

```
figure(5)
plot(t,z)
grid on
xlabel('time (seconds)')
title('y1(t)+y2(t)')
axis([-dt*2*N/10 dt*2*N/10 -1.5 1.5])

display('PREMERE PER VISUALIZZARE LA TRASFORMATA DI FOURIER
    (complessa) Z(f)')

pause
Z=fft(fftshift(z));
figure(6)
plot(f,fftshift(real(Z))*dt)
axis([-3*fo 3*fo -100*dt 200*dt])
grid on
hold on
plot(f,fftshift(imag(Z))*dt,'r')
axis([-3*fo 3*fo -100*dt 200*dt])
xlabel('frequency (Hz)')
title('Real and Imaginary part of Z(f)')

display('PREMERE PER VISUALIZZARE z1(t)=z(t)*cos(2*pi*t*fo)')
pause

z1=z.*2.*cos(2*pi*t*fo);
z2=z.*2.*sin(2*pi*t*fo);
figure(7)
plot(t,z1);
grid on
xlabel('time (seconds)')
title('z(t)*cos(2*pi*t*fo)')
axis([-dt*2*N/10 dt*2*N/10 -1.5 2.5])
hold on

display('PREMERE PER VISUALIZZARE z2(t)=z(t)*sin(2*pi*t*fo)')
pause
plot(t,z2,'r');
grid on
xlabel('time (seconds)')
title('z(t)*2*cos(2*pi*t*fo) and z(t)*2*sin(2*pi*t*fo) ')
axis([-dt*2*N/10 dt*2*N/10 -1.5 2.5])

display('PREMERE PER VISUALIZZARE LA TRASFORMATA DI FOURIER
    (complessa) Z1(f)')
pause
```

Codeblock 13: Modulazione IQ

```
figure(8)
Z1=fft(fftshift(z1));
Z2=fft(fftshift(z2));
subplot(211)
plot(f,fftshift(real(Z1))*dt)
axis([-3*fo 3*fo -100*dt 300*dt])
grid on
xlabel('frequency (Hz)')
title('Real part of Z1(f)')
hold on
subplot(212)
plot(f,fftshift(imag(Z1))*dt)
axis([-3*fo 3*fo -150*dt 300*dt])
grid on
xlabel('frequency (Hz)')
title('Imaginary part of Z1(f)')
hold on

display('PREMERE PER VISUALIZZARE LA TRASFORMATA DI FOURIER
    (complessa) Z2(f)')
pause

subplot(211)
plot(f,fftshift(real(Z2))*dt,'r')
axis([-3*fo 3*fo -100*dt 300*dt])
xlabel('frequency (Hz)')
title('Real part of Z1(f) and real part of Z2(f)')
subplot(212)
plot(f,fftshift(imag(Z2)*dt),'r')
title('Imaginary part of Z1(f) and imaginary part of Z2(f)')
axis([-3*fo 3*fo -150*dt 300*dt])
xlabel('frequency (Hz)')

pause
display('PREMERE PER VISUALIZZARE IL SEGNALE DEMODULATO COSENO')

figure(9)
plot(t,z1,'y');
grid on
xlabel('time (seconds)')
title('I+Q Demodulated signals')
axis([-dt*2*N/10 dt*2*N/10 -1.5 2.5])
hold on
plot(t,x1)

display('PREMERE PER VISUALIZZARE IL SEGNALE DEMODULATO SENO')
pause
```

Codeblock 14: Modulazione IQ

```
plot(t,z2,'g');
plot(t,x2,'r')
plot(t,x1)
```

Il programma non è poi complesso quanto la sua lunghezza potrebbe lasciare immaginare. Al contrario la lunghezza del sorgente dipende dalla lunga sequenza di operazioni che sono richieste per effetturare correttamente una modulazione in fase e quadratura. Innanzitutto il programma genera i due segnali di partenza in bassa frequenza (linee 14-15 e 39-40). Dal punto di vista del codice la funzione gausswin restituisce una curva gaussiana, e i due parametri passati alla funzione servono a specificare i punti che comporranno la curva e la deviazione standard. Il programma prosegue poi nel mostrare i valori delle rispettive trasformate dei due segnali e effettua la modulazione in fase e in quadratura moltiplicando il segnale x1 per un coseno (riga 61) ed il segnale x2 per un seno (riga 72). Vengono mostrate le trasformate dei due segnali prodotto Y1 ed Y2 ottenuti al punto precedente (righe 86 e 96) e viene definito il segnale z come somma dei due segnali y1 ed y2 (riga 104). Il passaggio finale di demodulazione viene mostrato alle righe 129 e 130: z1 viene ottenuto tramite moltiplicazione per un seno e z2 tramite moltiplicazione per coseno. I grafi temporali mostrati alla fine svelano la capacita' di ricostruire i segnali x1 ed x2 di partenza. Per correggere le imperfezioni si ricorda che è necessario un filtraggio passa basso.

Per un ripasso teorico si rimanda anche alle utili slide trovate a:
`http://www-dsp.elet.polimi.it/fdt/prati/LEZIONI/06_modulazione_IQ/Modulazione_IQ.PDF`.

7 | Sync periodico

Il programma rappresenta il seno cardinale periodico, trasformata della sequenza unitaria di N campioni, a partire da n=0 fino a n=N-1

Codeblock 15: Sync periodico

```
% PROGRAMMA Sync periodico

N=4;
fi=(-2:.001:2)+.0005;
x=exp(-j*pi*fi*(N-1)).*sin(pi*fi*N)./sin(pi*fi);
subplot(211)
plot(fi,abs(x))
title('modulo del sinc periodico')
xlabel('frequenza normalizzata')
axis([-2 2 0 N])
grid on
subplot(212)
```

Codeblock 16: Sync periodico

```
plot(fi,angle(x))
title('fase del sinc periodico')
xlabel('frequenza normalizzata')
axis([-2 2 -pi pi])
grid on
```

Il programma esegue operazioni preliminari nelle prime righe andando a definire il dominio delle frequenze ed il segnale trasformato della sequenza unitaria. Dopodiche viene semplicemente effettuata una stampa del modulo (vedi funzione abs) ed una stampa della fase (vedi funzione angle).

8 | Campionamento

Il programma campiona il segnale x(t)con intervallo di campionamento DT ottenendo il segnale campionato xn. Esegue la ricostruzione del segnale tempo-continuo x(t) convolvendo i campioni di xn con il filtro di ricostruzione ideale h(t) (sinc troncato). Mostra poi la sovrapposizione del filtro di ricostruzione applicato ai singoli campioni mettendo in evidenza che nessun campione interpolato interagisce con gli altri. Inoltre mostra la somma progressiva dei campioni interpolati con il filtro di ricostruzione fino ad ottenere il segnale ricostruito x(t).

Codeblock 17: Campionamento

```
% PROGRAMMA Campionamento

clear all
close all

dt=.001;
t=(0:dt:8);
x=gausswin(length(t),2.5);
DT=0.5;
T=(0:DT:8);
xn=x(1:DT/dt:length(t));
figure(1)
clf
plot(t,x)
axis([0 8 -0.25 1.1])
xlabel('time')
title('original time-continuous signal')
grid on
hold on
pause
stem(T,xn,'r')
```

Codeblock 18: Campionamento

```
title('sampled signal')
xlabel('time')

tt=(-16:dt:16);
h=sinc(tt/DT);

pause
figure(1)
clf
axis([0 8 -0.25 1.1])
grid on
hold on
stem(T,xn,'r')
title('sampled signal')
xlabel('time')


pause

for k=1:length(T);
    figure(1)
    hold on
    axis([0 8 -0.25 1.1])
    grid on
    plot(tt+(k-1)*DT,h*xn(k))
    xlabel('time')
    title('convolution of the reconstruction filter with each sample')
    pause
end

plot(t,x,'g')

pause
r=zeros(1,length(tt));
figure(1)
clf
for k=1:length(T);
r=r+xn(k)*sinc((tt-(k-1)*DT)/DT);
plot(tt,r)
hold on
stem(T,xn,'r')
xlabel('time')
title('convolution of the reconstruction filter with the sampled
    signal')
grid
axis([0 8 -0.25 1.1])
hold off
```

Codeblock 19: Campionamento

```
pause
end
```

Analizzando il sorgente tra le righe 6 e 11 vengono definiti lo step temporale dt, il dominio temporale t, il segnale gaussiano da campionare x, il passo di campionamento DT, gli istanti di tempo in cui campionare T e il segnale campionato xn. Dopo una stampa del segnale originario a cui viene sovrapposto il set di campioni a passo DT, il programma procede a eseguire la convoluzione con il segnale h (sync troncato), definito alla linea 25-26. Tramite il ciclo for viene mostrato il contributo dei vari campioni alla ricostruzione del segnale (linee 41-50). L'ultima parte di programma è volta a mostrare come la somma dei vari campioni permetta di giungere alla ricostruzione del segnale. In termini pratici è stato dichiarato un vettore di zeri (linea 55) al quale è stato progressivamente sommato il contributo dovuto ai vari campioni (linea 59).

9 | Alias Coseno

Il seguente programma campiona il segnale x(t)=cos(t*2*pi*.2) con intervallo di campionamento DT=.5 ottenendo il segnale campionato xn. Mostra poi che gli stessi campioni si ottengono aumentando la frequenza del coseno di multipli interi di 1/DT. Un delta di 1/DT in frequenza induce come è noto dalla teoria, problematiche di aliasing.

Codeblock 20: AliasCoseno

```
% PROGRAMMA AliasCoseno

clear all
close all

dt=.001;
t=(0:dt:5);
x=cos(t*2*pi*.2);
DT=.5;
T=(0:DT:5);
xn=x(1:DT/dt:length(t));
figure(1)
clf
plot(t,x)
axis([0 5 -1.5 1.5])
xlabel('time')
title('Frequency of the cosine = 1/5 Hz')
grid on
hold on
```

Codeblock 21: AliasCoseno

```
pause
stem(T,xn,'filled')
pause

hold off
plot(t,x)
axis([0 5 -1.5 1.5])
xlabel('time')
title('Frequency of the cosine = 1/5 + 1/DT Hz')
grid on
hold on
x2=cos(t*2*pi*(.2+1/DT));
plot(t,x2,'r')
stem(T,xn,'filled')
pause


hold off
plot(t,x)
axis([0 5 -1.5 1.5])
xlabel('time')
title('Frequency of the cosine = 1/5 + 2/DT Hz')
grid on
hold on
x3=cos(t*2*pi*(.2+2/DT));
plot(t,x3,'r')
stem(T,xn,'filled')
pause


hold off
plot(t,x)
axis([0 5 -1.5 1.5])
xlabel('time')
title('Frequency of the cosine = 1/5 + 3/DT Hz')
grid on
hold on
x4=cos(t*2*pi*(.2+3/DT));
plot(t,x4,'r')
stem(T,xn,'filled')
```

Il codice procede ad una semplice sequenza di stampe di segnali campionati e non, evidenziando il fenomeno dell'aliasing. Tutte le funzioni presenti in questo codice sono già state analizzate nei precedenti esempi.

10 | Alias frequenza

Il programma seguente campiona il segnale x(t)=sin(t*2*pi*3) con frequenza di campionamento a scelta ottenendo il segnale campionato xn. Mostra l'effetto dell'alias in frequenza quando non si rispetta il teorema del campionamento che dice che fs>6. Si provi a vedere cosa succede se si utilizza fs=24, fs=12, fs=4, fs=3.

```
% PROGRAMMA AliasFrequenza

clear all
close all
t=(0:.001:1);
k=length(t);
x(1:k)=sin(2*pi*3*t);
figure(1)
plot(t,x)
grid on
xlabel('time')
title('sinusoid frequency = 3Hz')
hold on
fs=input('frequenza di campionamento fs? ');
T=1/fs;
nT=(0:T:1);
kk=length(nT);
xn(1:kk)=sin(2*pi*3*nT);
figure(1)
stem(nT,xn,'r','filled')
hold off
```

Il codice stampa un segnale sinusoidale (riga 9) e poi attende che l'utente indichi la frequenza di campionamento desiderata. Procede poi ad evidenziare sul grafico i campioni alla frequenza specificata dall'utente. Tutte le funzioni presenti in questo codice sono già state analizzate nei precedenti esempi.

11 | Oversampling

Il programma innanzitutto campiona il segnale x(t)=cos(2*pi*t) con intervallo di campionamento DT=1/4 ottenendo il segnale xm. Poi interpola 4:1 i campioni di xm utilizzando un seno cardinale di 31 campioni sinc((-15:15)/4). Il segnale interpolato 4:1 xn viene costruito come somma dei seni cardinali centrati sui campioni xm e moltiplicati per i valori dei campioni xm corrispondenti.

```
% PROGRAMMA Oversampling

clear all
close all
dt=1/1000;
t=(-2:.001:5);
x=cos(2*pi*1*t);
figure(1)
plot(t,x)
L=length(t);
axis([-2 4 -1.5 1.5])
grid on
xlabel('time')
title('time continuous signal x(t)')
pause

DT=1/4;
T=(-2:DT:5);
xm=x(1:DT/dt:L);
Lxm=length(xm);
figure(1)
hold on
stem(T,xm,'r','fill')
title('Signal sampled with DT=1/4')
pause

DT2=DT/4;
T2=(-2:DT2:5);
T2=(-2-(15*DT2):DT2:5+(15*DT2));
xn=zeros(1,length(T2));
figure(2)
for m=1:Lxm
    xn(4*m-3:4*(m-1)+31)=xn(4*m-3:4*(m-1)+31)+xm(m)*sinc((-15:15)/4);
    stem(T2,xn,'fill')
    axis([-2 4 -1.5 1.5])
    grid on
    xlabel('time')
    title('Progressive reconstruction of the oversampled 4:1 signal')
pause(.2)
end
```

Nella parte iniziale del programma viene resettato il workspace, vengono inizializzati l'intervallo temporale minimo dt e l'asse dei tempi, viene poi definito e stampato il segnale x (righe 7 e 9). Il segnale viene poi campionato a sequenza 1/4 alla riga 19 e il set di campionamenti viene sovrapposto al precedente grafico. Alla riga 27 viene definito il nuovo passo di campionamento DT2, il vettore xn è inizializzato a 0 e accumula i risultati parziali delle interpolazioni (linea 33).

12 | Trasformata discreta

Il seguente programma esegue il calcolo della DFT di alcune sequenze x di N campioni e la confronta con la Trasformata di Fourier continua in frequenza della stessa squenza. Prima sequenza 1/4 1/2 1/4 simmetrica rispotto a n=0. Seconda sequenza -1/4 1/2 -1/4 simmetrica rispotto a n=0. Terza sequenza cos(2*pi*n/8) con n tra 0 e N-1 (numero intero di cicli). Quarta sequenza cos(2*pi*n/7) con n tra 0 e N-1 (numero non intero di cicli)

Codeblock 22: TFdiscreta

```
% PROGRAMMA TFdiscreta

clf
close all
clear all
DT=.1;
N=32;
n=(0:N-1);
% First time series (LP)
x=zeros(N,1);
x(1)=1/2;
x(2)=1/4;
x(N)=1/4;
stem(n,x,'filled');
xlabel('TIME SAMPLES')
title('first time series to be tranformed')
display('PLEASE NOTICE THE CIRCULAR FORMAT OF THE TIME SERIES:
    ACTUALLY THE 3 SAMPLES ARE SYMMETRIC WRT 0')

pause
To=N*DT;
DF=1/To;
X=fft(x);
stem(n,real(X))
title('DFT of the time series x')
xlabel('FREQUENCY SAMPLES')
pause
stem(DF*n,real(X))
title('DFT of the time series x')
xlabel('FREQUENCY HZ')

pause
M=1000;
m=(0:M-1);
xx=zeros(M,1);
xx(1)=1/2;
xx(2)=1/4;
```

Codeblock 23: TFdiscreta

```
xx(M)=1/4;
TTo=M*DT;
df=1/TTo;
XX=fft(xx);
plot(df*m,real(XX))
xlabel('FREQUENCY HZ')
title('TF continuous in frequency of the time series x')
hold on
pause
stem(DF*n,real(X),'r','filled')
title('TF (blue) and DFT (red) of the time series x')

pause
hold off

% Second time series (HP)
clf
close all
clear all
DT=.1;
N=32;
n=(0:N-1);
x=zeros(N,1);
x(1)=1/2;
x(2)=-1/4;
x(N)=-1/4;
stem(n,x,'filled');
xlabel('TIME SAMPLES')
title('second time series to be tranformed')
pause
To=N*DT;
DF=1/To;
X=fft(x);
stem(n,real(X))
xlabel('FREQUENCY SAMPLES')
title('DFT of the time series x')
pause
stem(DF*n,real(X))
xlabel('FREQUENCY HZ')
title('DFT of the time series x')

pause
M=1000;
m=(0:M-1);
xx=zeros(M,1);
xx(1)=1/2;
```

Codeblock 24: TFdiscreta

```
xx(2)=-1/4;
xx(M)=-1/4;
TTo=M*DT;
df=1/TTo;
XX=fft(xx);
plot(df*m,real(XX))
xlabel('FREQUENCY HZ')
title('TF continuous in frequency of the time series x')
hold on
pause
stem(DF*n,real(X),'r','filled')
title('TF (blue) and DFT (red) of the time series x')

display('PRESS TO COMPUTE THE INVERSE DFT FROM THE SAMPLES OF X')
pause
hold off
y=ifft(X);
stem(n,y,'r','filled');
xlabel('TIME SAMPLES')
title('IDFT of the frequency series X')

pause

% Third time series (cos: integer number of cycles)

clf
close all
clear all
DT=.1;
N=32;
n=(0:N-1);
x=cos(2*pi*n/8);
stem(n,x,'filled');
xlabel('TIME SAMPLES')
title('third time series to be tranformed')
pause
To=N*DT;
DF=1/To;
X=fft(x);
stem(n,real(X))
hold on
stem(n,imag(X),'r')
xlabel('FREQUENCY SAMPLES')
title('DFT of the time series x (real=blue, imag=red)')
hold off
pause
```

Codeblock 25: TFdiscreta

```
stem(DF*n,real(X))
hold on
stem(DF*n,imag(X),'r')
xlabel('FREQUENCY HZ')
title('DFT of the time series x (real=blue, imag=red)')
hold off

pause
M=1000;
m=(0:M-1);
xx=zeros(M,1);
xx(1:N)=cos(2*pi*n/8);
TTo=M*DT;
df=1/TTo;
XX=fft(xx);
plot(df*m,real(XX))
hold on
plot(df*m,imag(XX),'r')
xlabel('FREQUENCY HZ')
title('TF continuous in frequency of the time series x (real=blue,
    imag=red)')
grid on
pause
stem(DF*n,real(X),'b','filled')
stem(DF*n,imag(X),'r','filled')
title('TF and DFT of the time series x (real=blue, imag=red)')

pause

% Fourth time series (cos: non-integer number of cycles)

clf
close all
clear all
DT=.1;
N=32;
n=(0:N-1);
x=cos(2*pi*n/7);
stem(n,x,'filled');
xlabel('TIME SAMPLES')
title('fourth time series to be tranformed')
pause
To=N*DT;
DF=1/To;
X=fft(x);
stem(n,real(X))
```

Codeblock 26: TFdiscreta

```
hold on
stem(n,imag(X),'r')
xlabel('FREQUENCY SAMPLES')
title('DFT of the time series x (real=blue, imag=red)')
hold off
pause
stem(DF*n,real(X))
hold on
stem(DF*n,imag(X),'r')
xlabel('FREQUENCY HZ')
title('DFT of the time series x (real=blue, imag=red)')
hold off

pause
M=1000;
m=(0:M-1);
xx=zeros(M,1);
xx(1:N)=cos(2*pi*n/7);
TTo=M*DT;
df=1/TTo;
XX=fft(xx);
plot(df*m,real(XX))
hold on
plot(df*m,imag(XX),'r')
xlabel('FREQUENCY HZ')
title('TF continuous in frequency of the time series x (real=blue,
    imag=red)')
grid on
pause
stem(DF*n,real(X),'b','filled')
stem(DF*n,imag(X),'r','filled')
title('TF and DFT of the time series x (real=blue, imag=red)')
```

Il codice nella prima sezione contiene la definizione del passo e del numero dei campioni alle righe 6-7, e del segnale discreto x alle righe 10-13. Poichè x è costituito da solo 3 campioni si sceglie di partire da un vettore di soli zeri e di modificare i soli valori diversi da zero alle righe 11-13. Il codice prosegue poi al calcolo del segnale trasformato tramite la funzione fft (riga 22) mostrando anche il grafo in frequenza (riga 27). La sezione di codice che inizia alla riga 31 realizza un segnale campionato a frequenza più alta in modo da realizzare il segnale "continuo". e la relativa trasformata (riga 40). Lo schema delineato al passo precedente si ripete uguale anche per il secondo segnale, con l'unica variante che per il secondo segnale viene calcolata anche la DFT inversa (linee 99 e 100). Per i due segnali coseno, a parte la procedura di analisi che come detto ricalca quella appena esposta, si noti come l'analisi del segnale numero 3 produca una DFT con la coppia di impulsi tipici della trasformata del coseno, mentre il segnale 4, che non viene campionato per un numero intero di periodi, genera una trasformata anomala.

13 | Interpolazione in frequenza

Il seguente programma esegue un'interpolazione in frequenza con zero-padding nei tempi.

Codeblock 27: InterpFrequenza

```
% PROGRAMMA InterpFrequenza

clf
close all
clear all
O=input('oversampling factor? ');
figure(1)
DT=.1;
N=16;
n=(0:N-1);
x=zeros(N,1);
x(1)=1/4;
x(2)=1/2;
x(3)=1/4;
stem(n*DT,x,'filled');
xlabel('TIME SAMPLES')
title('time series x N=16 samples at DT=1/10 sec.')
pause

To=N*DT;
DF=1/To;
X=fft(x);
stem(DF*n,abs(X),'r','filled')
title('Absolute value of the DFT of the time series x')
xlabel('FREQUENCY HZ')

pause

figure(2)
DT=.1;
N2=N*O;
n2=(0:N2-1);
x=zeros(N2,1);
x(1)=1/4;
x(2)=1/2;
x(3)=1/4;
stem(n2*DT,x,'filled');
xlabel('TIME SAMPLES')
title('time series x N=32 samples at DT=1/10 sec.')
pause

To2=N2*DT;
DF2=1/To2;
```

Codeblock 28: InterpFrequenza

```
X2=fft(x);
stem(DF2*n2,abs(X2))
title('Absolute value of the DFT of the time series x after
    zero-padding')
xlabel('FREQUENCY HZ')
hold on
pause
stem(DF*n,abs(X),'r','filled')
hold off
```

Il precedente codice realizza un interpolazione con zero-padding del segnale definito alle righe 11-14. Alla riga 14 viene calcolata la trasformata di Fourier del segnale senza oversampling. Nella seconda parte del programma la sequenza di zeri viene aggiunta ad una copia del segnale originale (righe 33-36). Viene poi calcolata la DFT del secondo segnale (riga 44).

14 | Interpolazione nei tempi

Il seguente programma esegue un'interpolazione nei tempi con zero-padding in frequenza.

Codeblock 29: InterpTempi

```
% PROGRAMMA InterpTempi

clf
close all
clear all
O=input('oversampling factor? ');
figure(1)
DT=.1;
N=16;
n=(0:N-1);
x=zeros(N,1);
x(4)=1/4;
x(5)=1/2;
x(6)=1/4;
stem(n*DT,x,'filled');
xlabel('TIME SAMPLES')
title('time series x N=16 samples at DT=1/10 sec.')
pause

To=N*DT;
DF=1/To;
X=fft(x);
stem(DF*n,abs(X),'r','filled')
```

Codeblock 30: InterpTempi

```
title('Absolute value of the DFT of the time series x')
xlabel('FREQUENCY HZ')

pause

figure(2)
N2=N*O;
n2=(0:N2-1);
X2=zeros(N2,1);
X2(1:N/2+1)=X(1:N/2+1);
X2(N2:-1:N2-N/2+2)=X(N:-1:N/2+2);
stem(DF*n2,abs(X2),'r','filled')
title('Absolute value of the DFT after frequency zero-padding')
xlabel('FREQUENCY HZ')
pause
x2=ifft(X2)*O;
DT2=To/N2;
stem(DT2*n2,real(x2))
title('time series x after interpolation')
xlabel('TIME')
hold on
pause
stem(n*DT,x,'filled');
xlabel('TIME')
```

Il precedente programma, come nel caso dell'interpolazione in frequenza, definisce un segnale di partenza (righe 11-14) e ne calcola la trasformata di Fourier alla linea 22. Alle righe 30-34 viene definito un nuovo segnale X2, che è la copia con oversampling del segnale in frequenza X. La sintassi è un po' diversa dalle precedenti poichè gli zeri vengono inseriti non alla fine ma nella parte centrale del segnale. Tramite la trasformata inversa di Fourier alla riga 39 ci si riconduce al segnale nel tempo che viene poi stampato alla linea 41. Infine la stampa alla linea 46 sovrappone il segnale originale alla versione ottenuta tramite oversampling.

15 | Densita' di probabilita'

Il seguente programma stima la densita' di probabilità di una variabile casuale dai valori assunti dalla variabile casuale in N realizzazioni dell'esperimento. La d.d.p. della variabile casule si ottiene dividendo l'istogramma per il numero N delle realizzazioni e per l'intervallo di campionamento dell'istogramma.

Codeblock 31: DDP

```
% PROGRAMMA Densita' di probabilita'

clear all
close all

N=100000;
x=rand(1,N);
dado1=floor(x*6)+1;
y=rand(1,N);
dado2=floor(y*6)+1;
da=1;
ddpx=hist(dado1,[1:6])/N/da;
ddpxpy=hist(dado1+dado2,[1:12])/N/da;
figure(1)
subplot(211)
bar(ddpx)
title('D.d.p del risultato del lancio di un dado')
subplot(212)
bar(ddpxpy)
title('D.d.p della somma dei risultati del lancio di 2 dadi')
pause

% Esempio 2

clear all
close all

N=100000;
x=randn(1,N);
da=0.25;
ddpx=hist(x,[-4:da:4])/N/da;
figure(1)
bar((-4:da:4),ddpx)
title('D.d.p gaussiana a varianza unitaria e m=0')

pause

% Esempio 3

clear all
close all

N=100000;
x=(randn(1,N)).^2;
da=0.25;
ddpx=hist(x,[0:da:8])/N/da;
figure(1)
```

Codeblock 32: DDP

```
bar((0:da:8),ddpx)
title('D.d.p gaussiana a varianza unitaria e m=0 elevata al quadrato')

pause

% Esempio 4

clear all
close all

N=100000;
x=(randn(1,N)).^2;
y=(randn(1,N)).^2;
z=sqrt(x+y);
da=0.25;
ddpz=hist(z,[0:da:8])/N/da;
figure(1)
bar((0:da:8),ddpz)
title('D.d.p della radice della somma dei quadrati di 2 gaussiane
    indipendenti')
pause

% Esempio 5

clear all
close all

N=10000000;
x=cos(2*pi*rand(1,N));
da=0.001;
ddpx=hist(x,[-1:da:1])/N/da;
figure(1)
bar((-1:da:1),ddpx)
title('D.d.p del coseno di uniforme da 0 a 2pi')
pause

% Esempio 6

clear all
close all

N=10000000;
x=rand(1,N);
y=rand(1,N);
z=x.*y;
da=0.001;
ddpz=hist(z,[0:da:1])/N/da;
```

Codeblock 33: DDP

```
figure(1)
bar((0:da:1),ddpz)
title('D.d.p del prodotto di 2 uniformi indipendenti')
pause

% Esempio 7

clear all
close all

N=10000000;
x=rand(1,N)+1;
y=rand(1,N)+1;
z=x./y;
da=0.01;
ddpz=hist(z,[0:da:2])/N/da;
figure(1)
bar((0:da:2),ddpz)
title('D.d.p del quoziente di 2 uniformi indipendenti')
```

Il codice è suddiviso in 7 differenti porzioni. La prima mostra la densita' di probabilita' relativa al risultato dei lanci di un dado ed alla somma degli esiti dei lanci di due dadi. Come si può vedere dal codice, viene dichiarato un intero N alla linea 6. Tale intero rappresenta quante volte vengono lanciati i dadi per produrre le distribuzioni mostrate in seguito. Il comando rand presente ad esempio alle linee 7 e 9 genera un vettore di N elementi a valori tra 0 e 1. Gli istogrammi vengono costruiti con il comando hist presente alle linee 12-13. "da" rappresenta il passo che definisce la larghezza degli intervalli di base delle barre dell'istogramma e rappresenta quindi l'unita' di discretizzazione dell'asse delle ascisse. La seconda parte del codice sfrutta la funzione randn alla linea 29 (che restituisce un vettore di valori estratti da una distribuzione normale standard) e visualizza la distribuzione gaussiana. La terza parte del codice mostra la densita' gaussiana elevata al quadrato: si perviene a questo risultato elevando al quadrato tutti gli elementi del vettore x (linea 44). La quarta parte mostra la densita' relativa alla radice della somma dei quadrati di due distribuzioni gaussiane. Tale distribuzione è ottenuta a partire da due distribuzioni gaussiane elevate al quadrato (linee 59-60), sommate e poste sotto radice quadrata (linea 61). La quinta parte realizza la densita' di probabilita' del coseno in 0-2π, la cui analisi teorica è svolta anche sul libro al capitolo 6. Il codice ricalca gli esempi precedenti, unica differenza la definizione del vettore x che segue un coseno definito da una variabile casuale (linea 75). Le ultime due parti del codice trattano rispettivamente la distribuzione del prodotto e del quoziente di due uniformi.

16 | Processi sinusoidali

Il seguente programma visualizza N realizzazioni di un processo casuale con ampiezza casuale uniforme, con fase iniziale uniforme e altre combinazioni.

Codeblock 34: Processi sinusoidali

```
% PROGRAMMA Processi_sinusoidali

clear all
close all


t=[-10:.01:10];

N=input('numero realizzazioni? ');


for n=1:N;
    A=rand(1,1);
    r=A*cos(2*pi*0.1*t);
    figure(1)
    plot(t,r)
    pause(.1)
    hold on
end
hold off

pause

for n=1:N;
    fi=rand(1,1)*2*pi;
    r=cos(2*pi*0.1*t+fi);
    figure(2)
    plot(t,r)
    pause(.1)
    hold on
end
hold off

pause

for n=1:N;
    A=1;
    fi=(rand(1,1)-.5)*pi;
    r=A*cos(2*pi*0.1*t+fi);
    figure(3)
    plot(t,r)
```

Codeblock 35: Processi sinusoidali

```
    pause(.1)
    hold on
end
hold off

pause

for n=1:N;
    A=rand(1,1);
    r=A+cos(2*pi*0.1*t);
    figure(4)
    plot(t,r)
    pause(.1)
    hold on
end
hold off
```

Il precedente codice è divisibile in 4 sezioni. Dopo aver richiesto all'utente il numero di realizzazioni desiderate, sono presenti 4 cicli che disegnano ciascuno un set di realizzazioni. Il primo ciclo crea un set di realizzazioni che differiscono per ampiezza come si puo' ben vedere dalle linee 13 e 14. Il secondo ciclo crea le diverse realizzazioni variando in maniera casuale tra 0 e 2π la fase del segnale coseno alla linea 26. Il terzo ciclo genera le realizzazioni variando la fase tra $-\pi/2$ e $\pi/2$ (notare il comando alla linea 38 che permette di definire fi come uniforme tra -0.5 e 0.5). Il quarto ciclo realizza un set di realizzazioni sulla base dell'aggiunta di una costante additiva distribuita in maniera uniforme tra 0 e 1 (linee 50-51)

17 | Linear prediction

Il seguente programma fornisce la stima del campione all'istante n+1 noto il valore del campione all'istante n di una serie temporale generata dal modello $x(n+1) = rho*x(n)+kk*w(n)$ dove rho è il coefficiente di correlazione a passo 1 del processo casuale, kk è un coefficiente che garantisce la stazionarietà del processo casuale e w(n) è un processo casuale indipendente da x(n) e con ddp unifome tra -1/2 e +1/2. Il codice stima la densità di probabilità di una variabile casuale dai valori assunti dalla variabile casuale in N realizzazioni dell'esperimento. La d.d.p. della variabile casuale si ottiene dividendo l'istogramma per il numero N delle realizzazioni e per l'intervallo di campionamento dell'istogramma. Viene inoltre stimata e visualizzata tramite istogramma 3D la densità di probabilità congiunta tra x(n+1) e x(n). Da questa è facile realizzare l'andamento della ddp di x(n+1) condizionata a x(n) e il suo valore medio che corrisponde alla predizione che minimizza l'errore quadratico medio tra dato previsto e dato vero.

Codeblock 36: Linear prediction

```
% PROGRAMMA Linear prediction

clear all
close all

ro=input('coefficiente di correlazione? ');
N=2000000;
M=2000;
% Calcolo di kk per garantire la stazionarieta' del processo
kk=sqrt(1-ro^2);
x=zeros(1,N+1);
w=rand(1,N)-.5;
x(1)=rand(1,1)-.5;

% Generazione del processo casuale
for n=1:N;
   x(n+1)=ro*x(n)+kk*w(n);
end

% Stima del coefficiente di correlazione del processo a passo 1
   utilizzando
% M campioni di una realizzazione del processo

rostim=mean(x(1:M).*x(2:M+1))/var(x(1:M))

% Predizione lineare a passo 1 utilizzando il coefficiente di
   correlazione stimato
% e calcolo dell'errore di predizione
for n=1:N;
x2stim(n)=rostim*x(n);
error(n+1)=x(n+1)-x2stim(n);
end

display('VALORI OTTENUTI UTILIZZANDO IL COEFF. DI CORRELAZIONE
   STIMATO')
varianza_x=var(x)
varianza_err=var(error)

% Visualizzazione di M campioni del processo (blu), dei valori
   predetti
% (rossi) e dell'errore di predizione (verde)

figure(1)
stem(x(1:M))
pause
hold on
stem(x2stim(2:M+1),'r')
```

Codeblock 37: Linear prediction

```
pause
stem(error(2:M+1),'g')
hold off
pause

% Predizione lineare a passo 1 utilizzando il coefficiente di
    correlazione
% vero e calcolo dell'errore di predizione
for n=1:N;
x2stim(n)=ro*x(n);
error(n+1)=x(n+1)-x2stim(n);
end

display('VALORI OTTENUTI UTILIZZANDO IL COEFF. DI CORRELAZIONE VERO')
varianza_x=var(x)
varianza_err=var(error)

pause

% visualizzazione della densita' di probabilita' del processo x(n)
figure(3)
da=0.05;
cnt=hist(x(1:N),[-1:da:1]);
ddpx=cnt/N/da;
C=(-1:da:1);
plot(C,ddpx)
title('Ddp del processo casuale x(n)')

pause
% visualizzazione della densita' di probabilita' congiunta tra x(n)
    e x(n+1)
figure(4)
yx(1:N,2)=x(2:N+1);
yx(1:N,1)=x(1:N);
da=.05;
db=.05;
cnt=hist3(yx,{-1:da:1 -1:db:1});
ddpyx=cnt/N/(da*db);
imagesc((-1:da:1),(-1:db:1),ddpyx)
title('Ddp congiunta di x(n+1) e x(n)')

xn=0;

% visualizzazione della densita' di probabilita' di x(n+1)
    condizionata a x(n)
for k=1:5;
xn=input(' Inserire un valore di x(n) compreso tra -1 e +1 ... >> ');
```

Codeblock 38: Linear prediction

```
nn=floor((xn+1)/da);
figure(5)
plot((-1:db:1),ddpyx(nn,:)/(ddpx(nn)))
grid on
pause
end
```

Alla riga 6 l'istruzione input richiede il valore del coefficiente di correlazione desiderato. Alla riga 10 ha luogo l'assegnamento di kk in modo tale da non compromettere la stazionarieta' del processo. Il vettore x viene definito inizialmente come un vettore di N zeri alla linea 11. La generazione del processo casuale ha luogo a partire dalla linea 16 tramite ciclo for, sfruttando anche il vettore w, creato in precedenza alla linea 12 sfruttando il comando $-.5$ in modo da generare campioni nell'intervallo $[-1/2, 1/2]$. Alla linea 24 viene stimato il coefficiente ρ sulla base di M campioni. Nel ciclo seguente si calcolano passo passo una stima del segnale e l'errore derivante da tale stima (linee 28-29). La stampa a video dei risultati ottenuti finora e' posticipata alle linee 40, 43 e 45. Vengono calcolate la varianza e la varianza dell'errore anche sfruttano il coefficiente di correlazione reale (linee 57-58). Le successive due figure sono una rappresentazione della densita' di probabilita' di x, calcolata e disegnata alle linee 65-68, e della densita' di probabilita' congiunta x(n), x(n+1), generato tramite la funzioni alle linee 78-80. Si faccia attenzione al fatto che la funzione hist3 appartiene allo Statistic toolbox, pertanto nel caso si riscontrino errori in fase di esecuzione, il motivo potrebbe essere la mancanza del citato toolbox, che dovra' pertanto essere installato dal sito di Mathworks. L'ultimo blocco di codice visualizza il valore della densita' di probabilita' di x(n+1) condizionato ad un valore di x(n) inserito dall'utente.

18 | Limite centrale

Il seguente programma mostra il formarsi di una d.d.p. gaussiana a partire da delle v.c. uniformi in $[0 - 1]$.

Codeblock 39: Limite centrale

```
% PROGRAMMA Limite_centrale
clear
clf
bin=0.05;
N=500000;
x=rand(1,N);
y=rand(1,N);
z=rand(1,N);
v=rand(1,N);
```

Codeblock 40: Limite centrale

```
w=rand(1,N);
frR1=hist(x,[0+bin/2:bin:1-bin/2])/N/bin;
frR2=hist(x+y,[0+bin/2:bin:2-bin/2])/N/bin;
frR3=hist(x+y+z,[0+bin/2:bin:3-bin/2])/N/bin;
frR4=hist(x+y+z+v,[0+bin/2:bin:4-bin/2])/N/bin;
frR5=hist(x+y+z+v+w,[0+bin/2:bin:5-bin/2])/N/bin;

figure(1)
bar([0+bin/2:bin:1-bin/2],frR1)
pause
a=[0+bin/2:bin:1-bin/2];
ga=1/sqrt(2*pi*1/12)*exp(-(a-0.5).^2/(2*1/12));
hold on
plot([0+bin/2:bin:1-bin/2],ga,'r')
pause

figure(1)
clf
bar([0+bin/2:bin:2-bin/2],frR2)
pause
a=[0+bin/2:bin:2-bin/2];
ga=1/sqrt(2*pi*2/12)*exp(-(a-2*0.5).^2/(2*2/12));
hold on
plot([0+bin/2:bin:2-bin/2],ga,'r')
pause

figure(1)
clf
bar([0+bin/2:bin:3-bin/2],frR3)
pause
a=[0+bin/2:bin:3-bin/2];
ga=1/sqrt(2*pi*3/12)*exp(-(a-3*0.5).^2/(2*3/12));
hold on
plot([0+bin/2:bin:3-bin/2],ga,'r')
pause

figure(1)
clf
bar([0+bin/2:bin:4-bin/2],frR4)
pause
a=[0+bin/2:bin:4-bin/2];
ga=1/sqrt(2*pi*4/12)*exp(-(a-4*0.5).^2/(2*4/12));
hold on
plot([0+bin/2:bin:4-bin/2],ga,'r')
pause

figure(1)
```

Codeblock 41: Limite centrale

```
clf
bar([0+bin/2:bin:5-bin/2],frR5)
pause
a=[0+bin/2:bin:5-bin/2];
ga=1/sqrt(2*pi*5/12)*exp(-(a-5*0.5).^2/(2*5/12));
hold on
plot([0+bin/2:bin:5-bin/2],ga,'r')
```

Il precedente programma effettua una visualizzazione progressiva dei risultati prodotti tramite somma di variabili uniformemente distribuite in $[0-1]$. Nella parte iniziale del programma vengono creati i vettori contenenti valori casuali nell'intervallo $[0-1]$,linee 6-10. I segnali effettivamente visualizzati saranno quelli ottenuti per somma a partire dai segnali appena definiti: si vedano le linee 11-15. I successivi blocchi di codice rappresentano le visualizzazioni dei segnali delle linee 11-15. Si noti infine che ad ogni visualizzazione corrisponde anche la successiva rappresentazione di una gaussiana approssimante teorico dell'andamento.

19 | ISI

Il seguente codice indaga la problematica dell'interferenza tra simboli nella trasmissione dei segnali.

Codeblock 42: ISI

```
% PROGRAMMA ISI
% Autore: Claudio Prati
% Politecnico di Milano, 20 Marzo 2013

G=zeros(1,10000);
G(1:501)=1;
G(9501:10000)=1;
figure(1)
plot((-5000:4999),fftshift(G))
xlabel('frequenze')
grid
pause
g=ifft(G);
figure(2)
plot((-5000:4999)/10000,fftshift(real(g)))
xlabel('tempi')
hold on
pause
plot((-5000:4999)/10000,-fftshift(real(g)))
hold off
```

Codeblock 43: ISI

```
pause
G(1:1001)=cos(2*pi*(0:1000)/2000)*1/2+1/2;
G(9001:10000)=cos(2*pi*(1001:2000)/2000)*1/2+1/2;
figure(1)
hold on
plot((-5000:4999),fftshift(G),'r')
hold off
pause
g=ifft(G);
figure(3)
plot((-5000:4999)/10000,fftshift(real(g)))
xlabel('tempi')
hold on
pause
plot((-5000:4999)/10000,-fftshift(real(g)))
hold off
pause
figure(2)
hold on
plot((-5000:4999)/10000,fftshift(real(g)),'r')
```

Il precedente programma indaga la problematica della trasmissione numerica in banda base. In particolare viene mostrato come il segnale seno cardinale (linea 13), goda della proprieta' di risultare uguale a zero in tutti i campionamenti diversi dal punto di picco massimo. Questo risultato ci permetterebbe di dire che il seno cardinale sarebbe idealmente un'ottima soluzione per la trasmissione di segnale in quanto impulsi adiacenti non si influenzano ne si disturbano vicendevolmente. Dato che un segnale seno cardinale è un concetto puramente ideale nella realtà sono utilizzati per la trasmissione di impulsi segnali ispirati al seno cardinale, che invece di avere una trasformata rettangolare, hanno una trasformata raccordata a zero, più o meno marcatamente. Si nota inoltre che se si adottano impulsi in banda ispirati dalla trasformata del seno cardinale raccordati a zero, l'ampiezza del segnale nel tempo decresce più rapidamente, facilitando la lettura del ricevitore. Questa convergenza più rapida viene mostrata tramite le istruzioni della seconda parte del codice Matlab. Alla linea 26 viene mostrato il segnale raccordato in banda base e nelle successive visualizzazioni viene mostrato il comportamento nel dominio del tempo.