```
% PROGRAMMIAMO UNA FUNCTION CHE APPLICHI IL METODO DI
%  CAVALIERI-SIMPSON AD f(x) SU UN INTERVALLO [a,b]
%
% function [I,r]=cavsimp(f,a,b)
% Output: I = valore approx dell'integrale
%             usando Cavalieri-Simpson
%         r = ordine di esattezza polinomiale (3)
%
% E LA TESTIAMO SULL'ESEMPIO f(x)=exp(x), a=0, b=1

>> f=@(x) exp(x)
f =
    @(x)exp(x)
>> cavsimp(f,0,1)
ans =
    1.7189
>> EXACT = exp(1)-exp(0);
>> cavsimp(f,0,1) - EXACT
ans =
   5.7932e-04

% VERIFICHIAMO CHE L'ORDINE DI CONVERGENZA RISPETTO ALLA L
UNGHEZZA
%  DELL'INTERVALLO SIA 5
========= convergcavsimp.m ====================
HH=2.^(-(0:7)); E=[];
f=@(x) exp(x);
for k=1:length(HH)
    h=HH(k);
    I=cavsimp(f,0,h);
    E(k)=abs(exp(h)-1 - I);
end
loglog(HH,E,'.-');
r = minqua(log(HH),log(E)); %usando la funzione della volta scorsa
% oppure usando la funzione di MatLab:
% p = polyfit(log(HH),log(E),1); r=p(1);
disp(sprintf('Ordine di convergenza %f',r));
=============================================

>> converg_cavsimp
```

Ordine di convergenza 5.089163

```
%Osserviamo che usando
%   HH=2.^(-(0:15));
% si otterrebbero dei risultati falsati dal fatto che l'errore non
% può scendere al di sotto della precisione macchina. Infatti per
% h<10^(-3), l'errore è sempre 10^(-16). (Osservate il grafico!)

% PROGRAMMIAMO LA FORMULA COMPOSITA
%function I=cavsimpcomp(f,a,b,N)
% Input: funzione f(x) (function handle)
%        a,b: estremi dell'intervallo
%        N: numero sottointervalli
% Output: valore approx dell'integrale
%         usando Cavalieri-Simpson composita

% E TESTIAMOLA PER VERIFICARE CHE L'ORDINE DI CONVERGENZA
RISPETTO AL
%  NUMERO DI SOTTOINTERVALLI SIA 4

=============== converg_cavsimp_comp.m =============
NN=2.^((0:7)); E=[];
f=@(x) exp(x);
for k=1:length(NN)
   I=cavsimpcomp(f,0,1,NN(k));
   E(k)=abs(exp(1)-1 - I);
end
loglog(NN,E,'.-');
p = polyfit(log(NN),log(E),1);
disp(sprintf('Ordine di convergenza %f',p(1)));
==================================================

>> converg_cavsimp_comp
Ordine di convergenza -3.995770

% PROGRAMMIAMO ORA UN METODO DI QUADRATURA ADATTIVA BAS
ATO SU UNA
%  QUALUNQUE FORMULA DI QUADRATURA BASE.

% function [Q,n]=autoquad(f,a,b,toll,S)
```

```
% f(x), estremi dell'intervallo a e b, tolleranza
% S deve essere una funzione S(f,a,b) che applica una
%   formula di quadratura semplice ad f(x) su [a,b]
%   dichiarata come [I,r]=S(f,a,b)
%   con I=integrale approx
%       r=ordine esattezza polinomiale
% Output: Q=valore approx dell'integrale a meno di toll
%         n=numero totale di intervalli usati

% E TESTIAMOLA APPROSSIMANDO L'INTEGRALE DI f(x)=x^2*cos(10*x^2
)
% SULL'INTERVALLO [0,1]

>> f=@(x)x.^2.*cos(10*x.^2);
>> x=linspace(0,1);
>> plot(x,f(x))

% CI ASPETTIAMO CHE VENGANO USATI PIÙ INTERVALLI NELLE ZONE
"RIPIDE" E
%  MENO INTERVALLI IN QUELLE "PIATTE"

>> format long
>> [Q,N]=autoquad(f,0,1,1e-2, @cavsimp)
Q =
  -0.038788016869731
N =
    6
>> hold on
>> plot(x,f(x))

% E CHE IL NUMERO TOTALE DI INTERVALLI CRESCA AL DIMINUIRE D
ELLA
%  TOLLERANZA RICHIESTA

>> [Q,N]=autoquad(f,0,1,1e-3, @cavsimp)
Q =
  -0.038743994139892
N =
    10
>> [Q,N]=autoquad(f,0,1,1e-6, @cavsimp)
```

```
Q =
  -0.039258195242411
N =
    56
>> [Q,N]=autoquad(f,0,1,1e-8, @cavsimp)
Q =
  -0.039258216113008
N =
   166
```