```
% Esercizio 1
%
a=3e-4;  b=14e-4; T=213;
f=@(x)( 2.39e-11./(  x.^5.*(exp(1.432./(T.*x))-1) ) );
Iex=0.020690855481654;
% a)
figure(1)
fplot(f,[a,b])

% b)
x0=(a+b)/2;
I0=(b-a)*f(x0);                     E0=abs(I0-Iex)/abs(Iex);
I1=(b-a)/2*( f(a)+f(b) );           E1=abs(I1-Iex)/abs(Iex);
I2=(b-a)/6*( f(a)+4*f(x0)+f(b));   E2=abs(I2-Iex)/abs(Iex);

fprintf('Formula rettangoli:  %d  Formula trapezi:  %d   Formula Simpson:  %d\n',I0,I1,I2);
fprintf(' Errore rettangoli:  %d   Errore trapezi:  %d    Errore Simpson:  %d\n',E0,E1,E2);


% Esercizio 2
%
a=0;  b=2*pi;
f=@(x)( x.*exp(-x).*cos(2.*x) );
Iex=1/25*( 3*(exp(-2*pi) - 1) - 10*pi*exp(-2*pi) );

fprintf('Formula dei Trapezi composita\n')
fprintf('    n.int.      Integrale       Errore      p   \n')
for j=1:10,
   m = 2^j;
   I1m=TrapeziComp(a,b,m,f);
   Er(j)=abs(Iex-I1m)/abs(Iex);
   if (j>1),
     p = 1/log(1/2)*log(Er(j)/Er(j-1));
     disp([m,I1m,Er(j),p])
   end
end

fprintf('\n')
fprintf('Formula di Cavalieri-Simpson composita\n')
fprintf('    n.int.      Integrale       Errore      p   \n')
for j=1:10,
   m = 2^j;
   I2m=Cavalieri_Simpson(a,b,m,f);
   Er(j)=abs(Iex-I2m)/abs(Iex);
   if (j>1),
     p = 1/log(1/2)*log(Er(j)/Er(j-1));
```

```
      disp([m,I1m,Er(j),p])
    end
 end

 % Esercizio 3
 %
 a=0; b=1;
 f=@(x)(sqrt(x));
 Iex= 2/3;  %    primitiva 2/3 x^(3/2)

 fprintf('\n')
 fprintf('Confronto di convergenza formule composite\n')
 for j=1:10,
    m = 2^j;
    I1m=TrapeziComp(a,b,m,f);
    I2m=Cavalieri_Simpson(a,b,m,f);
    Er1(j)=abs(Iex-I1m)/abs(Iex);
    Er2(j)=abs(Iex-I2m)/abs(Iex);
    if (j>1),
      p1 = 1/log(1/2)*log(Er1(j)/Er1(j-1));
      p2 = 1/log(1/2)*log(Er2(j)/Er2(j-1));
      disp([m,I1m,Er1(j),p1,I2m,Er2(j),p2])
    end
 end
 figure(2)
 semilogy(2.^(1:10),Er1)
 hold on
 semilogy(2.^(1:10),Er2)
 hold off
 legend('trapezi','Simpson')
 xlabel('numero di sottointervalli m')
 ylabel('errore relativo')



 % Esercizio 4
 clear all
 m=10;
 a=-pi/2; b=pi/2;
 % funzione integranda
 g=@(t)(t.^2.*cos(t).*sin(t));

 % formula di quadratura per matrice di intervalli
 II = @(x)( (x(:,end)-0)/6/m.*( g(x(:,1))+2*sum(g(x(:,3:2:2*m)),2)+4*sum(g(x(:,2:2:2*m)),2)+g(x(:,2*m+1))) );

 % generazione griglia di nodi per plot
 k=0;
```

```
t=linspace(a,b,100);
for tt=t
   k=k+1; x(k,:)=linspace(0,tt,2*m+1);
end

% Plot della funzione f
figure(3)
plot(t,II(x),'k','LineWidth',4)
hold on

fprintf('\n')
disp(['   num nodi n   ',' ||s-f||_inf'])
for n=4:2:12

    xx=linspace(a,b,n+1);
% valutazione funzione integrale nei nodi xx(k)
    for k=1:n+1, xI=linspace(0,xx(k),2*m+1); y(k)=II(xI);end
% calcolo spline
    s=spline(xx(:),y(:),t);
% display dell'errore
    disp([n,max(abs(s(:)-II(x)))])

    plot(t,s,'LineWidth',4)

end

hold off
legend('f','s, n=4', 's, n=6', 's, n=8')




%======================================

function I1m=TrapeziComp(a,b,m,f)

H=(b-a)/m;
x=linspace(a,b,m+1);
I1m=H*( 0.5*f(x(1)) + sum(f(x(2:m))) + 0.5*f(x(m+1)) );

end

%======================================

function I2m=Cavalieri_Simpson(a,b,m,f)

H=(b-a)/m;
```

```
x=linspace(a,b,2*m+1);
I2m=H/6*( f(x(1))+ 2*sum(f(x(3:2:2*m))) + 4*sum(f(x(2:2:2*m))) + f(x(2*m+1)) );

end
```