Esercizio 5.1 *Scrivere una function Matlab che implementi la formula composita dei trapezi su $n+1$ ascisse equidistanti nell'intervallo $[a, b]$ relativamente alla funzione implementata da* `fun(x)`.
La function deve essere del tipo `If = trapcomp(a, b, fun, tol)`.
Soluzione:

```
function If = trapcomp(n, a, b, fun)
% function If = trapcomp(n , a, b, fun)
% Calcola l'integrale della funzione nell'intervallo a b,
% utilizzando la formula dei trapezi composita.
% fun funzione integranda
% If valore approssimato dell'integrale definito della funzione
If = 0;
h = (b-a) / n;
for i = 1:n-1
    If = If + fun(a + i*h);
end
If = (h/2) * (2*If + fun(a) + fun(b));
end
```

Esercizio 5.2 *Scrivere una function Matlab che implementi la formula composita di Simpson su $2n+1$ ascisse equidistanti nell'intervallo $[a, b,]$ relativamente alla funzione implementata da* `fun(x)`.
La function deve essere del tipo `If = simpcomp(a, b, fun, tol)`.
Soluzione:

```
function If = simpcomp(n, a, b, fun)
% function If = trapcomp(n , a, b, fun)
% Calcola l'integrale della funzione nell'intervallo a b,
% utilizzando la formula di Simpson composita
% fun funzione integranda
% If valore approssimato dell'integrale definito della funzione
If = fun(a) - fun(b);
h = (b-a) / n;
for i = 1:n/2
    If = If + 4*fun(a+(2*i-1)*h) + 2*fun((a+2*i*h));
end
If = If*(h/3);
end
```

Esercizio 5.3 *Scrivere una function Matlab che implementi la formula composita dei trapezi adattativa nell'intervallo* $[a, b]$ *relativamente alla funzione implementata da* `fun(x)` *e con tolleranza* `tol`.
La function deve essere del tipo `If = trapad(a, b, fun, tol)`.
Soluzione:

```
function If = trapad(a, b, fun, tol)
% function If = trapad(a, b, fun, tol)
% Calcola ricorsivamente l'integrale della funzione nell'
   intervallo a b
% utilizzando la formula dei trapezi adattiva.
% fun funzione integranda
% if approssimazione dell'integrale definito della funzione
h = (b-a)/2;
m = (b+a)/2;

If1 = h*(feval(fun, a) + feval(fun, b));
If = If1/2 + h*feval(fun, m);

err = abs(If - If1)/3;

if err > tol
    iSx = trapad(a, m, fun, tol/2);
    iDx = trapad(m, b, fun, tol/2);

    If = iSx + iDx;
end
end
```

Esercizio 5.4 *Scrivere una function Matlab che implementi la formula composita di Simpson adattativa nell'intervallo* $[a, b]$ *relativamente alla funzione implementata da* `fun(x)` *e con tolleranza* `tol`.
La function deve essere del tipo `If = simpad(a, b, fun, tol)`.
Soluzione:

```
function If = simpad(a, b, fun, tol)
% function If = simpad(a ,b ,fun, tol)
% Calcola l'integrale della funzione nell'intervallo a b
   utilizzando la
% formula di simpson adattiva
% fun funzione integranda
% If approssimazione dell'integrale definito della funzione
```

```
h = (b-a) / 6;
m = (a+b) / 2;
m1 = (a+m) / 2;
m2 = (m+b) / 2;

If1 = h*(feval(fun, a) + 4*feval(fun, m) + feval(fun, b));
If = If1/2 + h*(2*feval(fun, m1) + 2*feval(fun, m2) - feval(fun
    , m));

err = abs(If-If1) / 15;

if err>tol
    iSx = simpad(a, m, fun, tol/2);
    iDx = simpad(m, b, fun, tol/2);
    If = iSx+iDx;
end
end
```

Esercizio 5.5 *Calcolare quante valutazioni di funzione sono necessarie per ottenere una approssimazione di*

$$I(f) = \int_0^1 \exp(-10^6 x)dx$$

che vale 10^{-6} *in doppia precisione IEEE, con una tolleranza* 10^{-9}*, utilizzando le functions dei precedenti esercizi. Argomentare quantitativamente la risposta.*

Soluzione: Le functions degli esercizi precedenti esercizi sono state leggermente modificate e sono stati calcolati i seguenti valori:

Function	Valutazioni	Errore
Trapezi composita	10000001	8.3319×10^{-10}
Simpson composita	2000002	3.3715×10^{-10}
Trapezi adattiva	77823	1.1252×10^{-14}
Simpson adattiva	1038	1.6469×10^{-14}

Possiamo vedere che per la funzione presa in esame performano meglio le formule adattive in quanto individuano i nodi della partizione in base al comportamento locale della funzione, questo permette di minimizzare l'errore e di raggiungere la soglia di tolleranza prestabilita.
Viceversa le formule con ascisse equispaziate si sono rilevate inadeguate per questo tipo di funzione che presenta una rapida variazione di valore in una porzione dell'intervallo molto ristretta.

Il codice utilizzato per ottenere i risultati posti sopra:

```
% dati
If = 10^(-6);
fun = @(x) exp((-10^6)*x);
tol = 10^(-9);

[If_appr, nval] = trapcomp_val(10^7, 0, 1, fun);
print_res('Trapezi Composita', nval, abs(If-If_appr))

[If_appr, nval] = simpcomp_val(2*10^6, 0, 1, fun);
print_res('Simpson Composita', nval, abs(If-If_appr))

[If_appr, nval] = trapad_val(0, 1, fun, tol);
print_res('Trapezi Adattativa', nval, abs(If-If_appr))

[If_appr, nval] = simpad_val(0, 1, fun, tol);
print_res('Simpson Adattativa', nval, abs(If-If_appr))

% servono sempre n+1 valutazioni
function [If, nval] = trapcomp_val(n, a, b, fun)
    If = 0;
    nval = 0;
    h = (b-a) / n;
    for i = 1:n-1
        If = If + fun(a + i*h);
        nval = nval + 1;
    end
    If = (h/2) * (2*If + fun(a) + fun(b));
    nval = nval + 2;
end

% servono sempre n+2 valutazioni
function [If, nval] = simpcomp_val(n, a, b, fun)
    If = fun(a) - fun(b);
    nval = 2;
    h = (b-a) / n;
    for i = 1:n/2
        If = If + 4*fun(a+(2*i-1)*h) + 2*fun((a+2*i*h));
        nval = nval + 2;
    end
```

```
40      If = If*(h/3);
41  end
42
43  % tre valutazioni di fun ad ogni call
44  function [If, neval] = trapad_val(a, b, fun, tol)
45      h = (b-a)/2;
46      m = (b+a)/2;
47
48      If1 = h*(feval(fun, a) + feval(fun, b));
49      If = If1/2 + h*feval(fun, m);
50      neval = 3;
51
52      err = abs(If - If1) / 3;
53
54      if err > tol
55          [iSx, nSx] = trapad_val(a, m, fun, tol/2);
56          [iDx, nDx] = trapad_val(m, b, fun, tol/2);
57
58          If = iSx + iDx;
59          neval = neval + nSx + nDx;
60      end
61  end
62
63
64  % sei valutazioni di fun ad ogni call
65  function [If, neval] = simpad_val(a, b, fun, tol)
66      h = (b-a) / 6;
67      m = (a+b) / 2;
68      m1 = (a+m) / 2;
69      m2 = (m+b) / 2;
70
71      If1 = h*(feval(fun, a) + 4*feval(fun, m) + feval(fun, b));
72      If = If1/2 + h*(2*feval(fun, m1) + 2*feval(fun, m2) - feval
            (fun, m));
73
74      neval = 6;
75      err = abs(If-If1) / 15;
76
77      if err > tol
78          [iSx, nSx] = simpad_val(a, m, fun, tol/2);
79          [iDx, nDx] = simpad_val(m, b, fun, tol/2);
```

```
80
81          If = iSx+iDx;
82          neval = neval + nSx + nDx;
83      end
84  end
85
86  % function per stampare il risultato
87  function print_res(fun_name, nval, err)
88      disp(strcat(fun_name, ' - valutazioni f:'))
89      disp(nval)
90      disp(strcat(fun_name, ' - errore:'))
91      disp(err)
92  end
```