Integrazione numerica

Sia $y = f(x)$ una funzione definita in un intervallo $[a, b]$, il metodo dei trapezi 1 approssima l'integrale della funzione sul dominio attraverso la seguente forumula:

$$I = \frac{b-a}{2}[f(b) + f(b)],$$

quello che si fa é approssimare la funzione con piú trapezi (o rettangoli come si vede in Fig. 2),

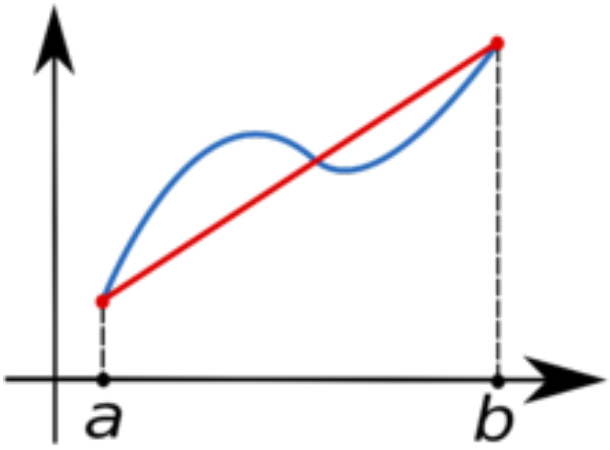

Figure 1: Approssimazione integrale attraverso trapezi

sia allora Δx la lunghezza del passo, vale che se $\Delta x \to 0$, l'integrale approssimato convergerá a quello esatto. in Matlab esistono comandi che cacolano integrali attraverso metodi numerici

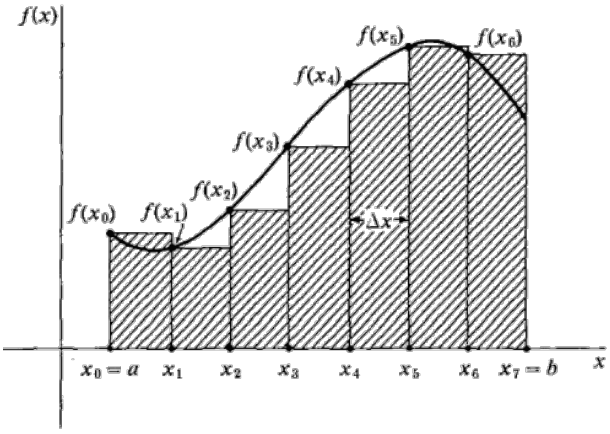

Figure 2: Approssimazione integrale attraverso rettangoli

(Gauss, Simpson, trapezi, rettangoli per funzioni di due variabili).

Esempio 1: Si vuole calcolare

$$\int_0^1 \cos(x)\mathrm{dx}$$

il comando "trapz" calcola il valore approssimato di un integrale di una funzione in un dato intervallo, ecco un esempio di codice:

```
>> x=0:0.1:1;  (suddivide l'intervallo [0,1] in sottointervalli di lunghezza 0.1)
>> y=cos(x);   (genera il vettore dei valori della
               funzione nei nodi della suddivisione)
>> s=trap(x,y)
  s=
        0.847..
```

Esercizio: Confrontare il risultato dell'integrale con una suddivisione $\Delta x = 0.05$ e $\Delta x = 0.025$.

Vediamo adesso un m-file che calcola iterativamente l'integrale di una funzione su un intervallo dato, diminuendo man mano il passo Δx, si genera allora una successione di integrali $\{S_k\}$, in cui ad ogni iterazione viene raddoppiato il numero di nodi, fino a che non é verificata la condizione che $|S_{k+1} - S_k| < \varepsilon$

```
clear
k=1; (numero di intervalli della suddivizione)
x=linspace(0,1,k+1) ;
y=cos(x);
S0=trapz(x,y);
errore=1;  (precisione iniziale)
contatore=0;
while (errore>=0.00001)&(contatore<=100)
   k=k*2;
   x=linspace(0,1,k+1);
   y=cos(x);
   S1=trapz(x,y);
   errore=abs(S1-S0);
   S0=S1;
   contatore=contatore+1;
end
```

Salviamo il file come "trapezio.m" ed eseguiamolo dalla Command Window.

```
>> trapezio
     S0=
              0.84142
>> contatore
     contatore=
                   16
```

Rendiamo ora piú generale la routine passando la funzione $f(x)$ e gli estremi dell'intervallo allo script "trapezi"

```
clear
k=1; (numero di intervalli della suddivizione)
x=linspace(a,b,k+1);
y=f(x);
S0=trapz(x,y);
errore=1;  (precisione iniziale)
contatore=0;
while (errore>=0.00001)&(contatore<=100)
   k=k*2;
   x=linspace(a,b,k+1);
   y=f(x);
   S1=trapz(x,y);
   errore=abs(S1-S0);
   S0=S1;
   contatore=contatore+1;
end
```

costruiamo la funzione che passa la "f"

```
function [y]=f(x)
y=sqrt(1-x.*x);
```

che salveremo come f.m, ed eseguiamo dalla Command Window

```
>> clear
>> a=0;
>> b=1;
>> trapezi
   S0 =
            0.7853   (e' il valore dell'integrale =  pi/4 )
>> pi/4     (valore esatto di pi/4)
  ans =
            0.7853
```

Ricordiamo peró che a e b sono variabili globali, é quindi preferibile lavorare con variabili locali in una data funzione

```
function [S]=integrale(a,b,precisione,F)
k=1;
x=linspace(a,b,k+1)
y=feval(F,x);    (calcola la f nei nodi di x)
S0=trapz(x,y);
errore=1;
contatore=0;
while (errore>=0.00001)&(contatore<=100)
   k=k*2;
   x=linspace(a,b,k+1);
   y=feval(F,x);
   S1=trapz(x,y);
   errore=abs(S1-S0);
   S0=S1;
   contatore=contatore+1;
end
```

Salviamo il file come "integrale.m" ed eseguiamolo

```
>> clear
>> c=0;
>> d=1;
>> eps=0.0001;  (chiamo le variabili, non essendo globali, come voglio)
>> g=inline('1./(1+x.*x.*x)'); (inline e' un oggetto che definisce la funzione)
>> z=integrale(c,d,eps,g);
>> z
   z =
         0.8356355
```

Altre istruzioni per il calcolo di integrali:

```
>> clear
>> S=quad(@f,0,1);
   S =
              0.8353    (pi/4)
```

Tale funzione utilizza la formula di Simpson.

Osservazione: La precisione di default é di 10^{-6} ma che puó essere anche migliorata ad esempio a 10^{-9}.

```
>> S=quad(@f,0,1,1.0e-9);    (1.0e-9 e' la precisione)
```

Puó essere anche definita attraverso oggetti inline

```
>> g=inline('1./(1+x.*x.*x)');
>> S=quad(g,0,1,1.0e-9);    (1.0e-9 e' la precisione)
    S =
              0.8353    (pi/4)
```

É possibile ritornare in outupt il numero dei nodi

```
>> [S,nn]=quad(@f,0,1,1.0e-9);   (nn e' il numero di nodi)
    S =
              0.8353
    nn =
              138
```

Osservazione: É immediato verificare che l'integrale sará piú preciso all'aumentare dei nodi.