Capitolo 4

La quadratura

4.1 Generalità

La misura delle aree è uno dei problemi scientifici più antichi. In epoca babilonese ed egiziana, ad esempio, venivano calcolate delle buone approssimazioni dell'area di un cerchio con il cosiddetto **metodo di esaustione**[1]. Problema analogo era la costruzione di un quadrato con area uguale a quella di un cerchio assegnato mediante l'ausilio di riga e compasso in un numero finito di passi. Tale problema era noto come **quadratura del cerchio**[2] e, più in generale, il termine **quadratura** veniva utilizzato per riferirsi a qualunque problema di calcolo dell'area o del volume di un oggetto di forma nota. Il problema del calcolo delle aree e dei volumi è tuttora di interesse in molte applicazioni di varia natura come, ad esempio, nella determinazione del volume di terra da rimuovere per la costruzione di strade, o nel calcolo dell'area della superficie della carrozzeria di un'automobile. In genere i problemi sono relativi a figure geometriche di forma irregolare, per cui non è sempre possibile applicare formule elementari per il calcolo delle aree (o dei volumi).

♣ **Esempio 4.1.** Si supponga di voler calcolare l'area della regione piana chiusa mostrata in Figura 4.1. Tale regione può essere divisa in cinque sottoregioni tracciando delle linee orizzontali e verticali come mostrato in Figura 4.2.

[1] Il metodo consisteva nel dimostrare che due grandezze sono uguali mostrando che la loro differenza è inferiore a qualsiasi quantità data, per quanto piccola. In particolare, per determinare l'area del cerchio, si inscrive al suo interno una successione di poligoni regolari con un numero di lati crescente. La superficie del poligono diventa sempre più grande ad ogni passo, ma rimane sempre inferiore a quella del cerchio. Inoltre la differenza tra le due aree diventa sempre più piccola. In tal modo l'area del cerchio veniva calcolata con la massima precisione possibile nonostante non la si conoscesse esattamente. Utilizzando tale metodo Archimede fornì anche un'approssimazione per il numero irrazionale π (il rapporto tra area del cerchio e quadrato del raggio), stimando che esso appartiene all'intervallo chiuso $[223/71, 22/7]$ commettendo un errore dell'ordine di 10^{-4}.

[2] Tale problema non ha soluzione. F. Lindemann, nel 1882 mostrò infatti che il numero π è un numero irrazionale.

Annulli

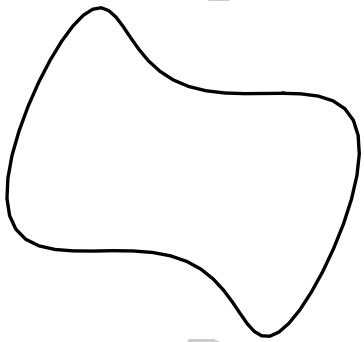

Figura 4.1: Regione piana chiusa

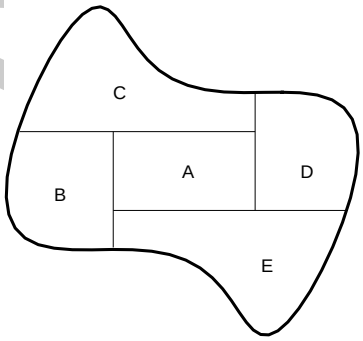

Figura 4.2: Suddivisione della regione in sottoregioni

A. Murli

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

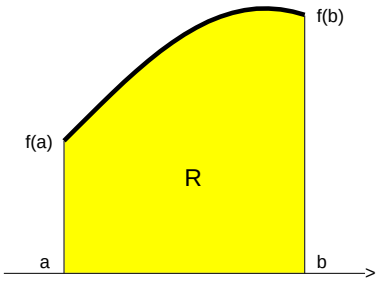

Figura 4.3: Area di un rettangoloide

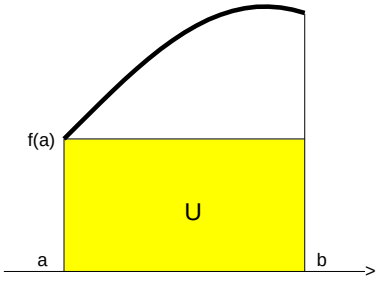

Figura 4.4: Approssimazione di $\mathcal{A}(R)$ mediante la formula rettangolare

La sottoregione A è un rettangolo la cui area può essere calcolata mediante il prodotto delle sue dimensioni. Le sottoregioni B, C, D, E hanno forma irregolare ed il calcolo della loro area può essere ricondotto al calcolo dell'area di una regione piana delimitata da una curva. ♣

Se si indica con $y = f(x)$ la funzione (non negativa) che rappresenta la curva, alla regione R in Figura 4.3 si dà il nome di **rettangoloide** di base l'intervallo $[a, b]$, relativo alla funzione $y = f(x)$, la cui area $\mathcal{A}(R)$ è detta **integrale definito di** $f(x)$ **tra** a **e** b ed è indicata con

$$\mathcal{A}(R) \equiv I[f] = \int_a^b f(x)\,dx \tag{4.1}$$

4.2 Formule di quadratura: primi esempi

Il metodo più naturale per approssimare l'area del rettangoloide R è quello di sovrapporre ad esso delle figure elementari e di calcolare l'area di ognuna di esse. Sia, ad esempio, R il rettangoloide di base l' intervallo $[a, b]$, relativo alla funzione $y = f(x)$ mostrato in Figura 4.3.

Un primo modo per approssimare l'area $\mathcal{A}(R)$ del rettangoloide R è quello di utilizzare un rettangolo U con lati di misura $(b - a)$ (la base) e $f(a)$ (l'altezza). Detta allora

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

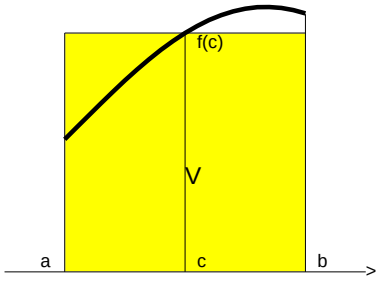

Figura 4.5: Approssimazione di $\mathcal{A}(R)$ mediante la formula del punto medio

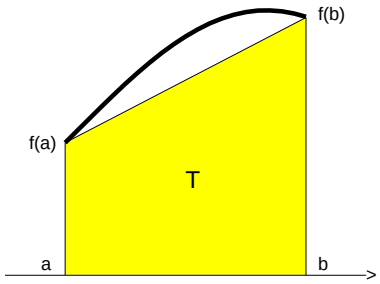

Figura 4.6: Approssimazione di $\mathcal{A}(R)$ mediante la formula trapezoidale

$\mathcal{A}(U)$ l'area del rettangolo U si ha che

$$I[f] \equiv \mathcal{A}(R) \simeq \mathcal{A}(U) = f(a)(b-a) \tag{4.2}$$

Tale formula è detta **formula rettangolare** (Figura 4.4). Inoltre è possibile scegliere il rettangolo V con lati di misura $(b-a)$ e $f(c)$ dove $c = (a+b)/2$ è il punto medio dell'intervallo $[a, b]$, ottenendo così la **formula del punto medio** (Figura 4.5):

$$I[f] \equiv \mathcal{A}(R) \simeq \mathcal{A}(V) = f(c)(b-a) \tag{4.3}$$

Un terzo metodo per approssimare l'area di R è quello di utilizzare un trapezio T con basi di misura $f(a)$ e $f(b)$ ed altezza $(b-a)$, cioè

$$I[f] \equiv \mathcal{A}(R) \simeq \mathcal{A}(T) = \frac{(b-a)}{2}\,[f(a) + f(b)] \tag{4.4}$$

Quest'ultima formula è detta **formula trapezoidale** ed è indicata anche con il simbolo $T[f]$. Come mostrato in Figura 4.6, la formula trapezoidale equivale a calcolare l'area al di sotto del segmento di retta passante per la coppia di punti $(a, f(a))$, $(b, f(b))$.

Analogamente è possibile ricavare altre formule. Infatti la parabola passante per i punti $(a, f(a))$, $(b, f(b))$ e $(c, f(c))$ (dove c è il punto medio di $[a, b]$) ha equazione:

$$y = f(a)\frac{(x-c)(x-b)}{(a-c)(a-b)} + f(c)\frac{(x-a)(x-b)}{(c-a)(c-b)} + f(b)\frac{(x-a)(x-c)}{(b-a)(b-c)}$$

per cui calcolando l'area al di sotto di tale parabola si ottiene:

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

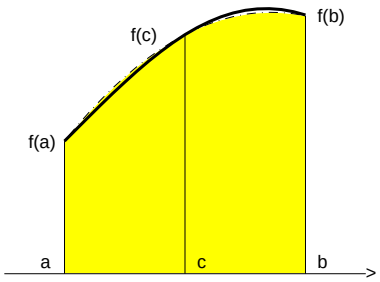

Figura 4.7: Approssimazione di $\mathcal{A}(R)$ mediante la formula di Simpson

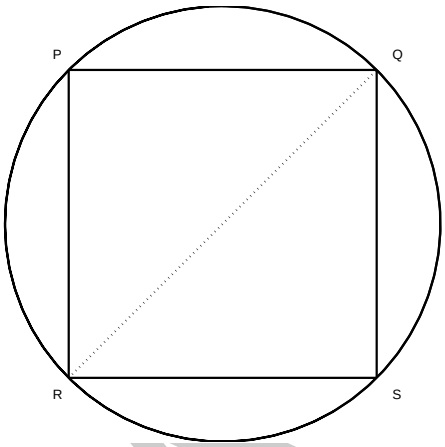

Figura 4.8: Decomposizione del cerchio in 5 sottoregioni

$$I[f] \equiv \mathcal{A}(R) \simeq \frac{(b-a)}{2}\left(\frac{f(a)}{3} + \frac{4}{3}f(c) + \frac{f(b)}{3}\right) \tag{4.5}$$

Tale formula è nota come **formula di Simpson**[3] (Figura 4.7) ed è indicata con $S[f]$.

♣ **Esempio 4.2.** Si voglia calcolare l'area del cerchio Ω di raggio 1 in un sistema aritmetico floating point con base $\beta = 10$ e precisione $t = 5$ (si ricorda che il valore di tale area è $\mathcal{A}(\Omega) = \pi = 3.1415926....$).

Il cerchio può essere diviso in 5 sottoregioni (Figura 4.8). Una di queste è un quadrato che ha come diagonale il segmento $\overline{QR}$ lungo 2 e come lato un segmento la cui lunghezza è $\sqrt{2}$. L'area del quadrato è quindi 2. Le restanti sottoregioni sono calotte circolari uguali tra loro e la relativa area può essere calcolata con la formula trapezoidale. Una di tali sottoregioni è mostrata in Figura 4.9.

Se si considera il punto medio M di $[P, Q]$ come origine del riferimento cartesiano si ha che i vertici P e Q hanno rispettivamente coordinate $P \equiv (-\frac{\sqrt{2}}{2}, 0)$ e $Q \equiv (\frac{\sqrt{2}}{2}, 0)$ e la curva che sottende il segmento $\overline{PQ}$ è il grafico della funzione:

$$f(x) = \sqrt{1-x^2} - \frac{\sqrt{2}}{2}$$

[3]Nonostante il nome, tale formula fu proposta per primo da Cavalieri nel 1639 con il nome di *regola parabolica* e ripresa da James Gregory nel 1668. Solo nel 1743 Thomas Simpson rese famosa tale formula. Per tale motivo essa è nota anche con il nome di **formula di Cavalieri-Simpson.**

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

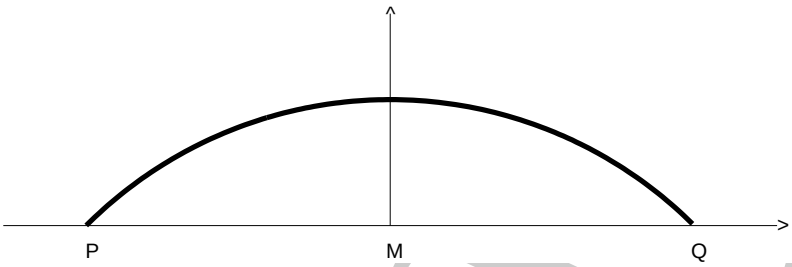

Figura 4.9: Area della calotta circolare

Tale regione è composta da due sottoregioni simmetriche relative ai due sottointervalli $[P, M]$ e $[M, Q]$. È sufficiente quindi calcolare solo l'area della sottoregione relativa al segmento $[M, Q]$. Cioè

$$I[f] = \int_0^{\sqrt{2}/2} \left(\sqrt{1 - x^2} - \frac{\sqrt{2}}{2} \right) dx = \frac{\pi - 2}{8} = 0.142699...$$

Utilizzando la formula trapezoidale (4.4) sull'intervallo $[M, Q]$ risulta $(b - a) = \sqrt{2}/2$ e

$$I[f] \simeq \mathcal{A}(T) = \frac{\sqrt{2}}{4} \left(1 - \frac{\sqrt{2}}{2} \right) = 0.10355 \times 10^0$$

mentre l'errore assoluto commesso nell'approssimazione di $I[f]$ con $\mathcal{A}(T)$ nella sottoregione relativa al segmento $[M, Q]$ è

$$|(\pi - 2)/8 - 0.10355 \times 10^0| \simeq 0.39149 \times 10^{-1}$$

Un'approssimazione dell'area del cerchio è quindi:

$$\mathcal{A}(\Omega) \simeq 2 + 8(0.10355 \times 10^0) = .28284 \times 10^1$$

e poiché l'area del cerchio di raggio 1 è $\pi = 3.141592..$, l'errore assoluto commesso nell'approssimare $\mathcal{A}(\Omega)$ con il valore ottenuto, è

$$|\pi - .28284 \times 10^1| \simeq 0.31315 \times 10^0$$

♣

Uno degli obiettivi che si pone la quadratura è quindi quello di determinare formule accurate per il calcolo di un integrale definito di una funzione $y = f(x)$ (non solo non negativa). Le formule finora proposte (4.2),(4.3),(4.4),(4.5) per il calcolo di (4.1) sono una combinazione lineare di valori della funzione integranda $f(x)$, con coefficienti dipendenti dall'ampiezza dell'intervallo di integrazione. Si può dare quindi la seguente definizione:

Definizione 4.1. (Formula di quadratura)

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

Fissata una funzione integrabile nel senso di Riemann [4] *ed un intero n, dati n punti $x_i \in [a, b]$ detti* **nodi** *e n valori A_i detti* **pesi**, *la combinazione lineare:*

$$Q[f] = A_1 f(x_1) + \cdots + A_n f(x_n)$$

prende il nome di **formula di quadratura**.

Una formula di quadratura rappresenta un'approssimazione dell'integrale $I[f]$, per cui è possibile dare anche la seguente definizione:

Definizione 4.2. (Errore di discretizzazione)
La differenza:

$$E[f] = I[f] - Q[f] = \int_a^b f(x)\,dx - \sum_{i=1}^{n} A_i f(x_i)$$

*è l'***errore di discretizzazione della formula di quadratura** $Q[f]$.

In Figura 4.10 è fornita la rappresentazione grafica dell'errore di discretizzazione della formula trapezoidale per il calcolo di (4.1).

♣ **Esempio 4.3.** Si calcoli l'errore della formula trapezoidale relativo all'integrale:

$$I[f] = \int_1^2 (3x + 1)\ dx = 5.5$$

Applicando la formula trapezoidale si ha:

$$T[f] = (4 + 7)/2 = 5.5$$

In tal caso l'errore di discretizzazione della formula trapezoidale è nullo. Si può facilmente osservare che tale proprietà è vera per ogni polinomio $p(x) \in \Pi_1$, cioè $E[p] = 0\ \forall p \in \Pi_1$. Per tale motivo si dice che la formula trapezoidale è **esatta per polinomi di primo grado**. ♣

[4] Si ricorda che una funzione è detta *integrabile nel senso di Riemann in [a,b]* se, detta $P_1, P_2, ..., P_m, ...$ una generica successione di partizioni dell'intervallo $[a, b]$ in sottointervalli $[t_{j-1}, t_j]$, tale che la successione dei sottointervalli di massima ampiezza Δ_1, $\Delta_2, ..., \Delta_m, ...$ converga a zero al crescere di m, le corrispondenti successioni di somme $S_1[f]$, $S_2[f], ..., S_m[f], ...$:

$$S_m[f] = \sum_{j=1}^{m} f(\xi_j)(t_j - t_{j-1}) \quad \xi_j \in [t_{j-1}, t_j]$$

(chiamate somme di Riemann), hanno un limite comune S (che è proprio l'integrale definito della funzione, detto anche integrale di Riemann) che è indipendente dalla scelta della partizione e dei punti ξ_j, cioè:

$$\lim_{m\to\infty} S_m[f] = S = \int_a^b f(x)\,dx.$$

In particolare sono funzioni integrabili nel senso di Riemann le funzioni continue e le funzioni limitate con un numero finito di discontinuità. Nel seguito diremo semplicemente funzioni integrabili per indicare le funzioni integrabili nel senso di Riemann.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

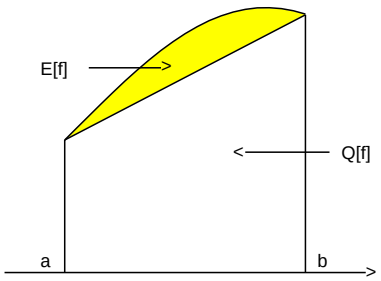

Figura 4.10: Rappresentazione grafica dell'errore di discretizzazione

4.2.1 Le formule di quadratura composite

Dalla Figura 4.10 è possibile notare che in generale l'errore di una formula trapezoidale dipende anche dall'ampiezza dell'intervallo $[a, b]$. Si può pensare allora di dividere l'intervallo $[a, b]$ in m sottointervalli (talvolta chiamati *pannelli*) di uguale ampiezza $\tau = (b - a)/m$ mediante i punti equidistanziati:

$$a = t_0 < t_1 < ... < t_m = b \quad \text{con} \quad t_j = a + j\tau, \quad j = 0, ..., m \tag{4.6}$$

e, poiché:

$$\int_a^b f(x)\,dx = \sum_{j=1}^{m} \int_{t_{j-1}}^{t_j} f(x)\,dx$$

si può utilizzare la formula trapezoidale in ognuno dei sottointervalli $[t_{j-1}, t_j]$[5].

♣ **Esempio 4.4.** Si utilizzi la formula trapezoidale con 4 sottointervalli per calcolare l'integrale:

$$I[f] = \int_a^b f(x)\,dx$$

In questo caso l'ampiezza dei sottointervalli coincide con la distanza tra i nodi in cui viene valutata la funzione integranda, ed è $h = \tau = t_j - t_{j-1} = \frac{b-a}{4}$ e per la proprietà additiva degli integrali risulta:

$$\int_a^b f(x)\,dx = \sum_{j=1}^{4} \int_{t_{j-1}}^{t_j} f(x)\,dx \simeq \sum_{j=1}^{4} \frac{(t_j - t_{j-1})}{2} \left[f(t_{j-1}) + f(t_j)\right] =$$

[5] L'idea di ridurre l'errore di una formula di quadratura applicando la stessa formula su più sottointervalli dell'intervallo di integrazione è basata sull'osservazione che, scelti m nodi t_j nell'intervallo $[a, b]$ la somma di Riemann è una formula rettangolare composita in cui l'altezza del rettangolo non coincide con il valore della funzione nell'estremo inferiore dell'intervallo.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

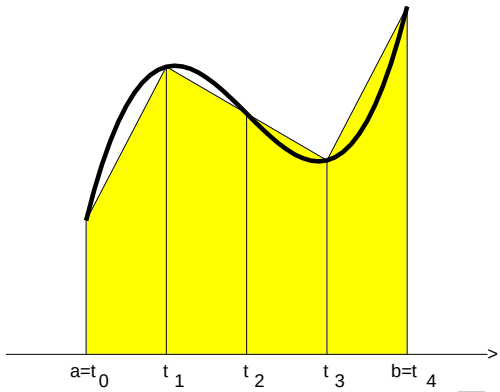

Figura 4.11: Formula trapezoidale composita su 4 sottointervalli

$$h\left[\frac{f(t_0)}{2}+f(t_1)+f(t_2)+f(t_3)+\frac{f(t_4)}{2}\right]$$

In Figura 4.11 è fornita la rappresentazione grafica della formula di quadratura ottenuta utilizzando la formula trapezoidale nei 4 sottointervalli $[t_0,t_1]$, $[t_1,t_2]$, $[t_2,t_3]$ e $[t_3,t_4]$, dell'intervallo $[a,b]$. ♣

♣ **Esempio 4.5.** Calcolare l'integrale (vedi esempio 4.2):

$$I[f]=\int_0^{\sqrt{2}/2}\left(\sqrt{1-x^2}-\frac{\sqrt{2}}{2}\right)dx$$

dividendo l'intervallo $[0,\sqrt{2}/2]$ rispettivamente in 2 e 4 sottointervalli in cui si applica la formula trapezoidale. Facendo riferimento alla Figura 4.12, mediante la formula trapezoidale in un solo sottointervallo, $T_1[f]$, si è avuta un'approssimazione dell'area del rettangoloide relativo all'intervallo $[M,Q]$ ed un errore $E_1[f]$ rispettivamente di:

$$T_1[f]=0.10355\times 10^0,$$

$$|E_1[f]|=|I[f]-T_1[f]|\simeq 0.39149\times 10^{-1}.$$

Se si utilizza invece la formula trapezoidale sui due sottointervalli $[M,R]$ e $[R,Q]$ con R punto medio di $[M,Q]$ si ha un'approssimazione

$$T_2[f]=\frac{\sqrt{2}}{4}\left[\frac{1}{2}\left(1-\frac{\sqrt{2}}{2}\right)+\left(\sqrt{\frac{7}{8}}-\frac{\sqrt{2}}{2}\right)\right]=0.13250\times 10^0$$

con un errore

$$|E_2[f]|=|I[f]-T_2[f]|=|(\pi-2)/8-0.13250\times 10^0|\simeq 0.10199\times 10^{-1}.$$

Dimezzando ancora l'ampiezza dei sottointervalli $[M,R]$ e $[R,Q]$, applicando la formula trapezoidale sui quattro sottointervalli, $[M,S]$, $[S,R]$, $[R,T]$ e $[T,Q]$, con S punto medio di $[M,R]$ e T punto medio

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

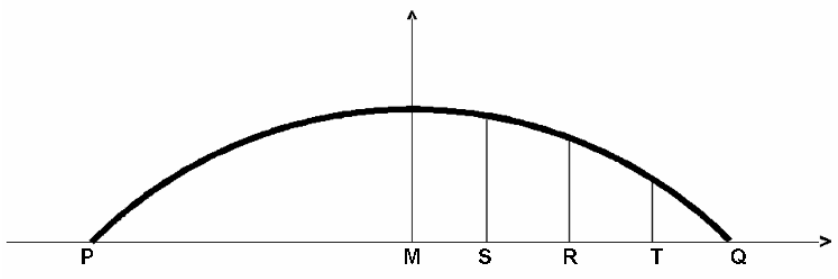

Figura 4.12: Area della calotta circolare

di $[R, Q]$, si ha:

$$\begin{aligned} T_4[f] &= \frac{\sqrt{2}}{8}\left[\frac{1}{2}\left(1-\frac{\sqrt{2}}{2}\right)+\left(\sqrt{\frac{31}{32}}-\frac{\sqrt{2}}{2}\right)+\left(\sqrt{\frac{7}{8}}-\frac{\sqrt{2}}{2}\right)+\left(\sqrt{\frac{23}{32}}-\frac{\sqrt{2}}{2}\right)\right] \\ &= 0.14014 \times 10^0, \end{aligned}$$

$$|E_4[f]| = |I[f] - T_4[f]| = |(\pi - 2)/8 - 0.140140 \times 10^0| \simeq 0.25591 \times 10^{-2}.$$

Dall'esempio si può notare che dimezzando l'ampiezza dei sottointervalli l'errore si riduce di circa 4 volte. ♣

Definizione 4.3. (Formula trapezoidale composita)
Suddiviso l'intervallo $[a, b]$ in m sottointervalli di ampiezza

$$\tau = \frac{(b-a)}{m},$$

mediante i punti equidistanziati:

$$a = x_0 < x_1 < ... < x_m = b \quad con \quad x_i = a + i\tau, \quad i = 0, ..., m \tag{4.7}$$

la formula di quadratura $T_m[f]$ che si ottiene applicando, in ognuno degli intervalli $[x_{j-1}, x_j]$, con $j = 1, ..., m$, la formula trapezoidale $T[f]$, è detta **formula trapezoidale composita**. *In questo caso la distanza tra i nodi h coincide con τ, per cui, posto*

$$x_i = a + ih, \quad i = 0, ..., m-1, \qquad x_m = b,$$

si ha:

$$\int_a^b f(x)\,dx \simeq T_m[f] = h\left(\frac{f(x_0)}{2} + f(x_1) + \cdots f(x_{m-1}) + \frac{f(x_m)}{2}\right) \tag{4.8}$$

È possibile definire altre formule composite, come, ad esempio, la formula di Simpson composita $S_m[f]$ su m sottointervalli $[x_{j-1}, x_j]$ di ampiezza $\tau = (b-a)/m$. In tal caso, dalla (4.5) si ha che la distanza tra i nodi è $h = \tau/2 = (b-a)/2m$, per cui posto

$$x_0 = a \qquad\qquad x_i = a + ih \quad i = 1, ..., 2m$$

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

la formula di Simpson composita risulta:

$$S_m[f] = h\left(\frac{f(x_0)}{3} + \frac{4}{3}f(x_1) + \frac{2}{3}f(x_2) + \cdots + \frac{2}{3}f(x_{2m-2}) + \frac{4}{3}f(x_{2m-1}) + \frac{f(x_{2m})}{3}\right)$$

In generale si ha:

Definizione 4.4. (Formula di quadratura composita)
Fissata una formula di quadratura

$$Q[f] = \sum_{i=1}^{n} A_i f(x_i)$$

ed un intervallo $[a, b]$*, si divida tale intervallo in* m *sottointervalli*

$$[t_{j-1}, t_j] \quad (j = 1, .., m).$$

Si definisce $Q_m[f]$ **formula di quadratura composita** *la formula che si ottiene applicando* $Q[f]$ *in ogni sottointervallo* $[t_{j-1}, t_j]$ *di ampiezza* $\tau_j = (t_j - t_{j-1})$*, cioè:*

$$Q_m[f] = \sum_{j=1}^{m} Q^{(j)}[f] = \sum_{j=1}^{m}\sum_{i=1}^{n} A_i^{(j)} f(x_i^{(j)}). \tag{4.9}$$

dove con $Q^{(j)}[f]$ *si è indicata la formula* $Q[f]$ *nel* j*-mo intervallo* $[t_{j-1}, t_j]$*, con* $x_i^{(j)}$ *si è indicato l'*i*-mo nodo di* $Q^{(j)}[f]$ *e con* $A_i^{(j)}$ *l'*i*-mo peso di* $Q^{(j)}[f]$*.*

Si noti il ruolo differente che giocano gli n nodi x_i della Definizione 4.1 e gli $m+1$ punti t_j della Definizione 4.4. I primi rappresentano gli n nodi di una formula di quadratura $Q[f]$ mentre i secondi sono utilizzati per suddividere l'intervallo $[a, b]$ in m sottointervalli in ognuno dei quali viene applicata la formula di quadratura fissata. In generale $Q_m[f]$ ha mn nodi, tranne nel caso in cui il primo e l'ultimo nodo di $Q[f]$ coincidano con gli estremi di ogni intervallo $[t_{j-1}, t_j]$ (come ad esempio accade per la formula trapezoidale e per la formula di Simpson). In tal caso $Q_m[f]$ possiede $m(n-1)+1$ nodi.

Definizione 4.5. (Formule convergenti)
Data una funzione integrabile $y = f(x)$*, la successione di formule composite* $\{Q_m[f]\}$ *è detta* **convergente** *se*

$$\lim_{m\to\infty} E_m[f] = 0 \quad \Leftrightarrow \quad \lim_{m\to\infty} Q_m[f] = \int_a^b f(x)\,dx$$

dove $E_m[f]$ *è l'errore dell'*m*-ma formula di quadratura* $Q_m[f]$*.*

L'esempio 4.5 ha messo in luce che l'errore di discretizzazione della formula trapezoidale composita si riduce quando si dimezza l'ampiezza dei sottointervalli h.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

Teorema 4.1. *Sia $T[f]$ la formula trapezoidale e $f(x)$ una funzione integrabile nell'intervallo $[a, b]$. Detta $T_m[f]$ la formula composita su m sottointervalli di ampiezza $\tau = (b-a)/m$ definita dalla (4.8) si ha:*

$$\lim_{m\to\infty} E_m[f] = 0$$

dove $E_m[f] = I[f] - T_m[f]$ è l'errore della formula trapezoidale composita $T_m[f]$.

Dimostrazione La formula trapezoidale $T[f]$ applicata all'intervallo $[t_{j-1}, t_j] \subseteq [a, b]$ di ampiezza τ è:

$$\int_{t_{j-1}}^{t_j} f(x)\,dx \simeq \frac{(t_j - t_{j-1})}{2}\,[f(t_{j-1}) + f(t_j)]$$

La formula trapezoidale composita $T_m[f]$ è allora:

$$\begin{aligned} T_m[f] &= \frac{1}{2}\sum_{j=1}^{m}(t_j - t_{j-1})\,(f(t_{j-1}) + f(t_j)) \\ &= \frac{1}{2}\left[\sum_{j=1}^{m}(t_j - t_{j-1})f(t_{j-1}) + \sum_{j=1}^{m}(t_j - t_{j-1})f(t_j)\right] \end{aligned}$$

Ma per l'integrabilità della funzione $f(x)$ si ha

$$\lim_{m\to\infty}\sum_{j=1}^{m}(t_j - t_{j-1})f(t_{j-1}) = \lim_{m\to\infty}\sum_{j=1}^{m}(t_j - t_{j-1})f(t_j) = \int_a^b f(x)\,dx = I[f]$$

da cui:

$$\lim_{m\to\infty} E_m[f] = I[f] - \lim_{m\to\infty} T_m[f] = I[f] - \frac{1}{2}\,[2I[f]] = 0$$

cioè la tesi. ■

Questo tipo di comportamento di riduzione dell'errore dimostrato per la formula trapezoidale composita al crescere del numero di sottointervalli è comune ad altre formule composite[6].

Un aspetto centrale della quadratura è fornire, oltre al valore dell'integrale $I[f]$, anche una stima dell'errore $E[f]$ basata su quantità calcolabili.

♣ **Esempio 4.6.** Si determini una stima calcolabile dell'errore della formula trapezoidale composita. Preliminarmente si dimostra il:

Teorema 4.2. *Sia $f \in C^2[a, b]$ e*

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

[6] La dimostrazione di ciò è data nel **Teorema 1.7** del **§1.2.3**, **Capitolo 1** del **Volume 2**.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

la formula trapezoidale. Detto $E[f] = I[f] - T[f]$ l'errore della formula trapezoidale, si ha:

$$|E[f]| \leq \frac{5M(b-a)^3}{12} \qquad M = \max_{x\in[a,b]} |f''(x)|.$$

Dimostrazione Siano

$$\begin{array}{ll} f(x) = f(a) + f'(a)(x-a) + f''(\xi_1)(x-a)^2/2 & \xi_1 \in [a,x] \\ f(x) = f(b) + f'(b)(x-b) + f''(\xi_2)(x-b)^2/2 & \xi_2 \in [b,x] \end{array}$$

le formule di Taylor per la funzione $f(x)$ rispettivamente di punti iniziali a e b. Sommando membro a membro, dividendo per 2 e integrando termine a termine si ha:

$$I[f] = \int_a^b f(x)\,dx = \int_a^b \frac{[f(a)+f(b)]}{2}\,dx +$$

$$+ \int_a^b \frac{[f'(a)(x-a) + f'(b)(x-b)]}{2}\,dx + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4}\,dx \tag{4.10}$$

da cui, applicando la formula trapezoidale $T[f]$ ai primi due integrali al secondo membro (le cui funzioni integrande sono polinomi di grado zero e di primo grado per cui la formula trapezoidale è esatta), si ha:

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

$$+ \frac{[f'(a)(b-a) + f'(b)(a-b)](b-a)}{4} + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4}\,dx$$

Da cui si ha:

$$E[f] = I[f] - T[f] = \frac{[f'(a) - f'(b)](b-a)^2}{4} + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4}\,dx \tag{4.11}$$

Per il Teorema di Lagrange applicato alla funzione $f'(x)$ è possibile trovare un punto $\xi \in [a,b]$ tale che:

$$f'(b) - f'(a) = f''(\xi)(b-a)$$

da cui, sostituendo nella (4.11), si ha

$$E[f] = -\frac{f''(\xi)(b-a)^3}{4} + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4}\,dx$$

Posto $M = \max_{x\in[a,b]} |f''(x)|$ è allora possibile scrivere:

$$|E[f]| \leq \left|\frac{M(b-a)^3}{4}\right| + \frac{M}{4}\left|\int_a^b \left[(x-a)^2 + (x-b)^2\right] dx\right| \tag{4.12}$$

Per l'integrale a secondo membro della (4.12) con facili calcoli si ha:

$$\int_a^b \left[(x-a)^2 + (x-b)^2\right]\,dx = \frac{2(b-a)^3}{3}$$

e quindi:

$$|E[f]| \leq \frac{M(b-a)^3}{4} + \frac{M(b-a)^3}{6} = \frac{5M(b-a)^3}{12}$$

cioè la tesi.

■

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

Corollario 4.1. *Nelle stesse ipotesi del teorema precedente, detta $T_m[f]$ la formula trapezoidale composita su m sottointervalli $[t_{j-1}, t_j]$ di ampiezza $h = \tau = (b-a)/m$ si ha che l'errore $E_m[f] = I[f] - T_m[f]$ è:*

$$|E_m[f]| \leq \frac{5M(b-a)h^2}{12} \quad con \quad M = \max_{x\in[a,b]} |f''(x)|$$

Dimostrazione Per il Teorema 4.2, in ogni sottointervallo $[t_{j-1}, t_j]$ di ampiezza $h = \tau = (b-a)/m$ si ha:

$$|E[f]| \leq \frac{5Mh^3}{12}$$

e per ottenere una maggiorazione di $E_m[f]$ basta sommare il contributo dell'errore in ogni sottointervallo $[t_{j-1}, t_j]$, cioè:

$$|E_m[f]| \leq \sum_{j=1}^{m} \frac{5Mh^3}{12} = m\frac{5Mh^3}{12} = \frac{5M(b-a)h^2}{12} \tag{4.13}$$

da cui la tesi. ■

Osserviamo che, dalla (4.13), se si dimezza la distanza tra i nodi h, l'errore si riduce di circa quattro volte. Detti allora $E_m[f]$ ed $E_{2m}[f]$ rispettivamente gli errori delle due formule trapezoidali composite $T_m[f]$ e $T_{2m}[f]$ assumiamo per il corollario del Teorema 4.2 che $|E_m[f]| \simeq 4|E_{2m}[f]|$. Approssimando $I[f]$ con le due formule considerate si ha:

$$\begin{aligned} I[f] - T_m[f] &= E_m[f] \\ I[f] - T_{2m}[f] &= E_{2m}[f], \end{aligned}$$

da cui, sottraendo membro a membro e passando ai valori assoluti, supponendo che E_m ed E_{2m} abbiano lo stesso segno, si ha

$$|T_{2m}[f] - T_m[f]| = |E_{2m}[f] - E_m[f]| \simeq 3|E_{2m}[f]|$$

e quindi

$$|E_{2m}[f]| \simeq |T_{2m}[f] - T_m[f]|/3.$$

L'errore $E_{2m}[f]$ può essere, quindi, stimato utilizzando la differenza tra le due formule $T_m[f]$ e $T_{2m}[f]$. ♣

È utile osservare che, sfruttando la proprietà di convergenza delle formule composite, se ad esempio è stata già calcolata la formula trapezoidale composita $T_m[f]$ per avere una stima più accurata dell'integrale (mantenendo l' equispaziatura tra i nodi) è possibile dimezzare la distanza tra i nodi, h, ottenendo così la formula $T_{2m}[f]$ i cui nodi hanno distanza $h/2$. Tale scelta comporta da un lato un miglioramento dell'accuratezza del risultato e dall'altro l'utilizzo del valore di $T_m[f]$ nel calcolo di $T_{2m}[f]$, richiedendo la valutazione della funzione integranda solo nei punti medi degli m sottointervalli $[x_{j-1}, x_j]$ di $[a, b]$:

$$T_{2m}[f] = \frac{1}{2}T_m[f] + \frac{h}{2}\left[f\left(a + \frac{h}{2}\right) + f\left(a + \frac{3h}{2}\right) + \ldots + f\left(b - \frac{h}{2}\right)\right]. \tag{4.14}$$

In questo caso le due formule trapezoidali composite $T_m[f]$ e $T_{2m}[f]$ si dicono innestate.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

Definizione 4.6. (Formule innestate)
Due formule $Q'[f]$ e $Q''[f]$, relative ad uno stesso intervallo, tali che l'insieme dei nodi di $Q'[f]$ è contenuto nell'insieme dei nodi di $Q''[f]$, costituiscono una **coppia di formule innestate**.

Data quindi una coppia di formule innestate $Q'[f]$ e $Q''[f]$ l'errore $E''[f] = I[f] - Q''[f]$ può essere stimato mediante la differenza:

$$|E''[f]| \simeq \alpha |Q'[f] - Q''[f]| \qquad \alpha > 0 \tag{4.15}$$

dove α è una costante che dipende dalle formule di quadratura utilizzate.

La disponibilità di formule innestate è particolarmente utile per ridurre la complessità computazionale delle formule di quadratura (e più in generale di un algoritmo per la quadratura). Infatti essa è determinata attraverso **il numero di valutazioni della funzione integranda** nei nodi x_i, perchè il numero di operazioni relative a tale calcolo è predominante (la funzione integranda può infatti essere combinazione di polinomi algebrici, funzioni razionali, irrazionali, trigonometriche e così via).

4.3 Esempio di sviluppo di software matematico per la quadratura

4.3.1 L'algoritmo

I dati di un problema di quadratura sono la funzione integranda f, l'intervallo di integrazione $[a, b]$ ed una tolleranza `Tol` che rappresenta l'accuratezza richiesta. Il modo più naturale per progettare un algoritmo di quadratura è quindi quello di prevedere come dati di input tali valori[7] e di produrre come output un valore `Res` (che sia un'approssimazione di $I[f]$) con un errore `Err` (che rappresenti una stima attendibile dell'errore $E[f]$) minore di `Tol`. Si noti quindi che un buon algoritmo di quadratura deve essere in grado di fornire, oltre ad una stima del valore dell'integrale, anche una stima dell'errore commesso.

Caratteristiche importanti di un tale algoritmo devono essere l'**affidabilità**, cioè la garanzia che il risultato soddisfi i vincoli della tolleranza richiesta, l'**efficienza**, cioè la capacità di calcolare il risultato con il minor numero possibile di valutazioni della funzione integranda e la **robustezza**, cioè la capacità dell'algoritmo di far fronte a situazioni anomale come ad esempio il caso di funzioni non integrabili.

Strategia implementativa. Al fine di soddisfare i vincoli richiesti sull'errore, un algoritmo per la quadratura può essere basato su uno schema iterativo che calcoli una successione convergente ad $I[f]$. Tale successione può essere costruita in base al

[7] La funzione integranda f va fornita all'algoritmo sotto forma di procedura.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

Teorema 4.1, sfruttando cioè la proprietà di convergenza delle formule composite al crescere del numero di sottointervalli in cui si divide l'intervallo di integrazione $[a, b]$.

Scelta della famiglia di formule. Poiché l'efficienza maggiore si raggiunge quando l'algoritmo calcola un risultato soddisfacente i requisiti di accuratezza con il minor numero di valutazioni di $f(x)$, le formule composite utilizzate in algoritmi di quadratura sono tali che, ad ogni passo dell'algoritmo, è possibile riutilizzare tutte, o quasi, le informazioni ottenute nei passi precedenti, in modo da ridurre il numero di valutazioni della funzione integranda. In particolare le formule più utilizzate sono quelle di tipo innestato come, ad esempio, le formule trapezoidali composite

$$\{T_m[f]\}_{m=2^k}, \quad k = 0, 1, 2, ... \qquad (T_1[f] \equiv T[f])$$

ottenute raddoppiando ad ogni passo il numero di intervalli.

Criterio di arresto. La scelta del criterio di arresto di una successione convergente al valore dell'integrale è legata alla disponibilità di stime ragionevoli ed affidabili dell'errore associato alle formule di quadratura utilizzate. Si è detto che una stima per la formula trapezoidale composita è fornita dalla (4.15) con $\alpha = 1/3$. In questo caso un criterio di arresto può essere basato sulla relazione[8]:

$$|T_m[f] - T_{2m}[f]|/3 < \texttt{Tol} \tag{4.16}$$

Oltre a verificare se il risultato soddisfa la tolleranza (assoluta) richiesta `Tol`, il criterio di arresto dell' algoritmo deve prevedere un controllo sul numero di valutazioni della funzione integranda, il quale deve essere minore di una costante `Maxval` fornita in input, in quanto, come in tutti gli algoritmi iterativi, è possibile che un criterio di arresto basato solo sulla stima dell'errore sia soddisfatto solo dopo un numero eccessivo di passi. Una variabile di output `Iflag` segnala tale eventualità.

Tali considerazioni sono alla base della seguente Procedura 4.1.

[8]In generale i criteri di arresto delle procedure implementate nelle librerie di software matematico sono più sofisticati della (4.16) e spesso richiedono che l'errore relativo, o una combinazione dell'errore relativo e dell'errore assoluto sia minore di una tolleranza assegnata, cioè, ad esempio

$$\alpha|Q_m[f] - Q_{2m}[f]| < max(tol_{abs}, tol_{rel}|Q_{2m}[f]|)$$

dove tol_{abs} e tol_{rel} sono rispettivamente la tolleranza richiesta per l'errore assoluto e per l'errore relativo. Controlli ulteriori possono essere effettuati anche sull'ampiezza degli intervalli. Infatti nel caso in cui la funzione integranda presenti delle singolarità, l'algoritmo può suddividere indefinitamente il dominio di integrazione senza generare un risultato soddisfacente la tolleranza richiesta.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

procedure Trapez1(**in:** $a, b, Tol, f, Maxval$; **out:** $Int, Err, Iflag$)

/# SCOPO: calcolo di un integrale definito mediante la formula trapezoidale composita con stima dell' errore di discretizzazione

/# SPECIFICHE PARAMETRI:

/# PARAMETRI DI INPUT:

var: a	: **reale**	{*primo estremo dell' intervallo di integrazione*}
var: b	: **reale**	{*secondo estremo dell' intervallo di integrazione*}
var: Tol	: **reale**	{*tolleranza da soddisfare*}
var: f	: **reale**	{*funzione integranda*}
var: $Maxval$	: **intero**	{*massimo numero di valutazioni*}

/# PARAMETRI DI OUTPUT:

var: Int	: **reale**	{*stima dell' integrale*}
var: Err	: **reale**	{*stima dell' errore di discretizzazione*}
var: $Iflag$	: **intero**	{*indicatore d' errore*}

Procedura 4.1: Algoritmo per la formula trapezoidale composita - continua

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
/# VARIABILI LOCALI:
  var: h       : reale    {distanza tra i nodi}
  var: x       : reale    {nodo attuale per il calcolo della formula}
                          {di quadratura}
  var: oldint  : reale    {formula trapezoidale composita}
                          {al passo precedente, necessaria}
                          {per il calcolo dell' errore}
  var: sum     : reale    {somma parziale per la costruzione della}
                          {formula trapezoidale composita}
  var: m       : intero   {numero dei nodi per la formula}
                          {trapezoidale composita}
  var: fval    : intero   {numero di valutazioni della}
                          {funzione integranda}

/# INIZIO ISTRUZIONI:
    m := 2;
    h := (b − a)/m;
    x := (a + b)/m;
    oldint := h ∗ (f(a) + f(b));
    Int := oldint/2 + h ∗ f(x);
    fval := 3;
    Err := |Int − oldint|/3;
    while (Err>Tol and                    {ciclo principale di}
      fval<Maxval) do                     {iterazione con test sull'errore}
      m := 2 ∗ m;
      h := h/2;
      sum := 0.;
      for k = 1, m, 2 do                  {calcolo della formula trapezoidale}
                                          {composita su m punti medi}
        x := a + k ∗ h;
        sum := sum + f(x);
```

Procedura 4.1: Algoritmo per la formula trapezoidale composita - continua

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
        fval := fval + 1;
      endfor
      oldint := Int;
      Int := oldint/2 + h * sum;
      Err := |Int − oldint|/3                    {stima dell'errore }
    endwhile
    if (fval < Maxval) then        {definizione della variabile Iflag}
      Iflag := 0;
    else
      Iflag := 1;
    endif
  end Trapez1
```

Procedura 4.1: Algoritmo per la formula trapezoidale composita - fine

4.3.2 Il software

Lo scopo di una routine per il calcolo di un integrale definito su un intervallo $[a, b]$ è quello di rendere trasparente all'utente i vari dettagli relativi all'algoritmo implementato e di fornire il risultato a partire da un numero minimo di informazioni indispensabili da precisare in input. In particolare:

Definizione 4.7. (Integratore automatico)
Si dice **integratore automatico** *(Figura 4.13) una routine basata su un algoritmo che richieda almeno i parametri di input:*

- *una routine per il calcolo di* `f(x)`*;*
- *gli estremi dell'intervallo* `a` *e* `b`*;*
- *una tolleranza* `Tol` *da soddisfare;*
- *un limite di valutazioni massime,* `Maxval` *per la funzione integranda;*

e fornisca almeno i parametri di output:

- *una stima* `Int` *di* $I[f]$ *ed* `Err` *di* $E[f]$
- *un indicatore* `Iflag` *che avvisi l'utente se è stata raggiunta la tolleranza* `Tol` *oppure il motivo per cui non è stata raggiunta.*

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

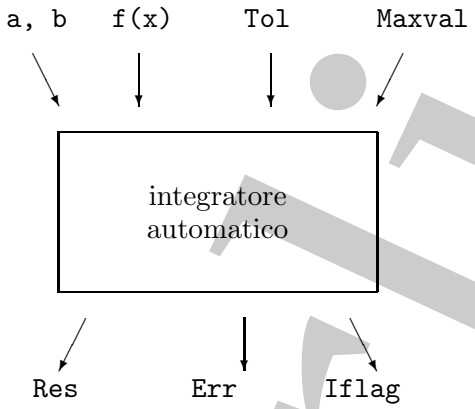

Figura 4.13: Schema di integratore automatico

A partire dalla procedura `Trapez1` è possibile costruire un elemento di software matematico per la quadratura. L'utente deve fornire un programma chiamante ed una routine `F(X)` che restituisca il valore della funzione integranda in ogni nodo previsto dalla formula di quadratura. Se, ad esempio, si utilizza il linguaggio di programmazione Fortran, la testata di tale subroutine può essere:

```
SUBROUTINE TRAPEZ1(A,B,F,TOL,MAXVAL,INT,ERR,IFLAG,FVAL)
```

Il programma chiamante `INTEG` illustra l'uso della subroutine `TRAPEZ1` nel caso si voglia integrare la funzione $f(x) = \sqrt{x(1-x)}$ nell'intervallo $[0, 1]$ con una tolleranza `TOL`=0.00001 e `MAXVAL`=10000. Il parametro, `FVAL`, fornisce in output il numero di valutazioni di funzioni effettuate. Separatamente, nella `FUNCTION F(X)`, è implementata la funzione integranda. Tale unità di programma deve essere fornita dall'utente insieme al programma chiamante e nel caso si voglia calcolare l'integrale di una diversa funzione è sufficiente modificare solo l'istruzione `F=SQRT(X*(1.-X))` che definisce la funzione.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
      PROGRAM INTEG
C
C dichiarazione delle variabili
C
      REAL A, B, TOL, INT, ERR, F
      INTEGER FVAL, IFLAG, MAXVAL
      EXTERNAL F
C
C inizializzazione delle variabili di input
C
      A=0.
      B=1.
      TOL=.00001
      MAXVAL=10000
C
C chiamata di TRAPEZ1
C
      CALL TRAPEZ1(A,B,F,TOL,MAXVAL,INT,ERR,IFLAG,FVAL)
C
C stampa dei risultati
C
      IF (IFLAG.eq.0) THEN
            PRINT*, `Integrale   =`,INT
            PRINT*, `Errore      =`,ERR
            PRINT*, `Val.  fun.  =`,FVAL
            PRINT*, `Iflag       =`,IFLAG
      ELSE
            PRINT*, 'tolleranza non raggiunta'
      ENDIF
      STOP
      END
C
C FUNCTION F(X)
C
      REAL FUNCTION F(X)
      REAL X
      F=SQRT(X*(1.-X))
      RETURN
      END
```

Esempio di programma chiamante FORTRAN per la subroutine TRAPEZ1

Un programma FORTRAN, che implementa la versione in singola precisione della subroutine `TRAPEZ1` e del programma chiamante precedente, è stato compilato ed ese-

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

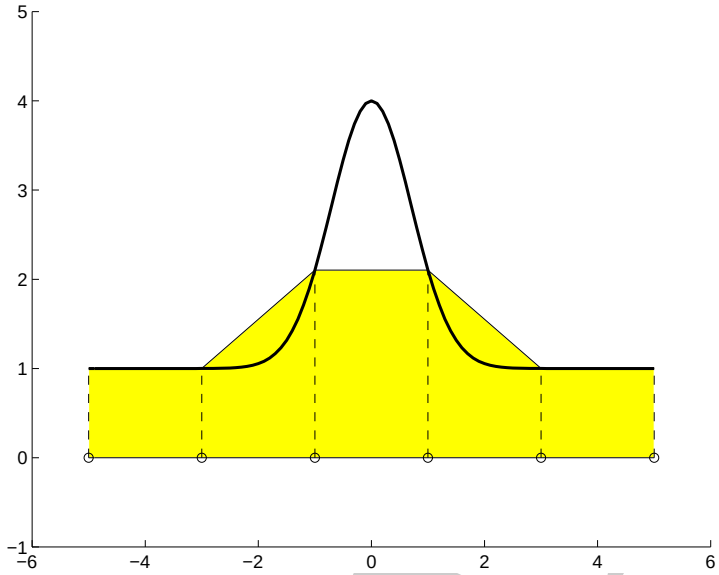

Figura 4.14: Approssimazione di $I[f]$ mediante la formula trapezoidale composita su 5 sottointervalli dell' intervallo $[-5, 5]$

guito su un calcolatore che utilizza il sistema aritmetico standard IEEE. L'esecuzione ha dato come risultati:

```
Integrale    = 0.3926973641
Errore       = 0.3695487976E-05
Val.  Fun.   = 4097
Iflag        = 0
```

4.3.3 Gli algoritmi adattativi per la quadratura

Nel paragrafo 4.3.1 è stato sviluppato un algoritmo per la quadratura basato sulla famiglia di formule trapezoidali composite. Tale algoritmo dimezza successivamente l'ampiezza dei sottointervalli finché una stima dell'errore commesso non risulti minore di una tolleranza assegnata.

Qualunque sia la funzione integranda, l'algoritmo utilizza sempre la stessa distribuzione dei nodi nell'intervalllo $[a, b]$ e ciò lo rende, in alcuni casi, poco efficiente.

♣ **Esempio 4.7.** Si calcoli

$$I[f] = \int_{-5}^{5} (3e^{-x^2} + 1)\, dx$$

mediante la formula trapezoidale composita con 5 e 10 intervalli.

Nella Figura 4.14 si nota come la prima scelta dei nodi non tiene ben conto dell'andamento della funzione nell'intervallo $[-1, 1]$, mentre in Figura 4.15 si vede come raddoppiando il numero di intervalli non si migliora significativamente l'approssimazione di $I[f]$ negli intervalli $[-5, -2]$ e $[2, 5]$. ♣

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

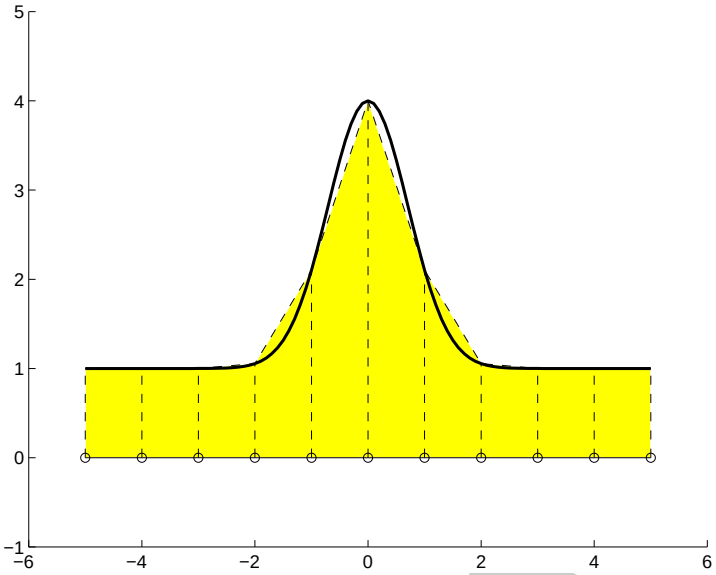

Figura 4.15: Approssimazione di $I[f]$ mediante la formula trapezoidale composita su 10 sottointervalli dell' intervallo $[-5, 5]$

L'esempio precedente mostra che la strategia di raddoppiare il numero degli intervalli della partizione di $[a, b]$ è in genere poco efficiente, in quanto non tiene conto dell'andamento della funzione integranda.

Una valida alternativa consiste nel cercare di adattare la distribuzione dei nodi alle proprietà della funzione integranda.

Definizione 4.8. (Algoritmo adattativo)
Un **algoritmo adattativo** *per la quadratura è un algoritmo che sceglie dinamicamente (cioè durante l'esecuzione) la distribuzione dei nodi, in maniera da adattare il partizionamento dell' intervallo di integrazione al particolare andamento della funzione integranda. Un algoritmo in cui l'insieme dei nodi è scelto secondo uno schema fissato, indipendentemente dalla funzione integranda, è detto* **algoritmo non adattativo**.

Gli algoritmi non adattativi possono essere utilizzati efficacemente nel caso in cui sia noto l'andamento della funzione integranda, scegliendo preventivamente la distribuzione più adatta alla funzione da integrare. Viceversa gli algoritmi adattativi sono più utili nel caso in cui non si hanno informazioni sufficienti sull'andamento della funzione, poiché, in questo caso, è l'algoritmo stesso a scegliere in maniera opportuna i nodi. All'interno della classe degli algoritmi adattativi sono possibili due strategie: la strategia **adattativa globale** e la strategia **adattativa locale**.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

La **strategia adattativa globale** opera come segue. Sia ε la tolleranza richiesta sull'intervallo iniziale $\Delta = [a, b]$; vengono calcolati, in tale intervallo, l'integrale $Q(\Delta)$ (ad esempio mediante la formula trapezoidale e la trapezoidale composita $T_2[f]$)) e l'errore $E(\Delta)$ (mediante la (4.16) con $m = 1$). Se $E(\Delta) < \varepsilon$ allora l'algoritmo termina e $Q(\Delta)$ è il risultato. In caso contrario si divide Δ in due sottointervalli Δ_1 e Δ_2 nei quali vengono calcolati $Q(\Delta_i)$ ed $E(\Delta_i)$, $i = 1, 2$. Se $E = E(\Delta_1) + E(\Delta_2) < \varepsilon$ allora si termina e $Q = Q(\Delta_1) + Q(\Delta_2)$ è il risultato. In caso contrario si divide a metà il sottointervallo in cui $E(\Delta_i)$ è maggiore. In generale al k-mo passo dell'algoritmo, l'intervallo $[a, b]$ è diviso in k+1 intervalli disgiunti di differente ampiezza. Se $\sum E(\Delta_i) < \varepsilon$ l'algoritmo termina e $\sum Q(\Delta_i)$ è il risultato. Altrimenti, viene diviso a metà il sottointervallo $\overline{\Delta}$ con massima stima dell'errore, cioè:

$$|E(\overline{\Delta})| = \max_{i=1,..,k+1} |E(\Delta_i)|$$

La strategia adattativa globale può essere così riassunta:

Posto l' intervallo corrente $\Delta = [a, b]$, inizializza ε
calcola un' approssimazione dell' integrale, $Q(\Delta)$,
mediante una formula di quadratura opportuna e la formula di quadratura composita
calcola $E(\Delta)$ (mediante la (4.15), con α opportuno)
while $(E(\Delta) > \varepsilon)$
    cerca il sottodominio $\overline{\Delta}$ con stima dell'errore massima
    dividi $\overline{\Delta}$ in 2 parti uguali
    calcola in tali parti una stima dell'integrale e dell'errore
    aggiorna $Q(\Delta)$ e $E(\Delta)$
endwile

Algoritmo adattativo globale

Nella **strategia adattativa locale** al primo passo vengono calcolati $E(\Delta)$ e $Q(\Delta)$ analogamente alla strategia globale. Se $E(\Delta) < \varepsilon$ allora l'algoritmo termina e $Q(\Delta)$ è il risultato. In caso contrario si divide Δ in due sottointervalli Δ_1 e Δ_2 nei quali vengono calcolati $Q(\Delta_i)$ ed $E(\Delta_i)$, $i = 1, 2$. Si esamina dapprima l'intervallo Δ_1. Fissata una *tolleranza locale* ε_1 (generalmente $\varepsilon_1 = amp(\Delta_1)\varepsilon/(b - a)$ con $amp(\Delta_1)$ ampiezza dell'intervallo Δ_1), se $E(\Delta_1) < \varepsilon_1$ si passa ad esaminare l'intervallo Δ_2, altrimenti si divide ulteriormente Δ_1. In generale, al passo k-mo l'intervallo $\Delta = [a, b]$ è diviso in $k + 1$ sottointervalli. Detto Δ_k l'intervallo in esame, se $E(\Delta_k) < \varepsilon_k$ si passa ad esaminare Δ_{k+1}, altrimenti si suddivide ulteriormente Δ_k. Ogni volta che viene soddisfatto il criterio di tolleranza locale relativo al sottointervallo Δ_k viene aggiunto $Q(\Delta_k)$ ad una somma parziale che, al termine della procedura, fornirà il valore di $I[f]$. L'algoritmo termina quando l'ultimo intervallo soddisfa il criterio di tolleranza locale.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

La strategia adattativa locale può essere così riassunta:

```
Posto l'intervallo corrente Δ = [a, b], inizializza ε
calcola Q̄(Δ) mediante una formula di quadratura opportuna
repeat
    dividi l'intervallo corrente in 2 parti
    calcola Q(Δ) mediante la formula di quadratura composita
    calcola E(Δ) (mediante la (4.15), con α opportuno)
    if (E(Δ) > ε_loc) then
        poni l'intervallo corrente Δ = intervallo piu' a sinistra
    else
        somma Q(Δ) al valore della somma parziale dell'integrale
        poni l'intervallo corrente Δ = successivo intervallo
    endif
until ( intervalli tutti esaminati )
```

Algoritmo adattativo locale

La Figura 4.16 mostra come la routine di `MATLAB quad` per la quadratura, basata su un algoritmo adattativo, scelga i nodi per la funzione

$$f(x) = 3e^{-x^2} + 1 \qquad \text{nell'intervallo} \qquad [-2, 8].$$

Si nota che dove la funzione integranda presenta un picco, l'algoritmo utilizza dei sottointervalli di ampiezza inferiore rispetto ai sottointervalli in cui la funzione è pressoché costante.

Dal punto di vista implementativo la strategia globale può essere realizzata mediante l'uso di una lista ordinata[9] secondo le stime crescenti degli errori $E(\Delta_i)$, in cui vengono conservate le informazioni relative a tutti gli intervalli esaminati (estremi degli intervalli e valori di $E(\Delta_i)$ e $Q(\Delta_i)$). In questo modo l'intervallo da suddividere (cioè quello con massima stima dell'errore) è sempre in testa alla lista. Viceversa la strategia locale può essere realizzata mediante l'uso di una pila[10], costruita inserendo durante ogni suddivisione i due intervalli così ottenuti nella testa della pila, prima quello di destra poi quello di sinistra; in tal modo l'intervallo da esaminare (quello più a sinistra) si trova sempre in testa alla pila. Entrambe le strategie presentano vantaggi e svantaggi, anche se le routine più efficienti attualmente esistenti sono basate sulla strategia globale.

[9]Una lista ordinata è una struttura dati dinamica (nel senso che la sua dimensione può variare durante l'esecuzione del programma). Tale struttura è composta da un insieme di nodi, ognuno dei quali formato da due campi: l'informazione e un puntatore al successivo nodo nella lista. Mediante il campo puntatore è possibile accedere ad ogni nodo della lista ed è possibile modificare l'ordine degli elementi solo intervenendo su tale campo.

[10]Una pila è una struttura dati dinamica in cui è possibile inserire ed estrarre gli elementi solo da un estremo della struttura dati. Tale estremo è detto *testa della pila*. Una struttura dati con tali caratteristiche è detta di tipo *Last In First Out* (LIFO).

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

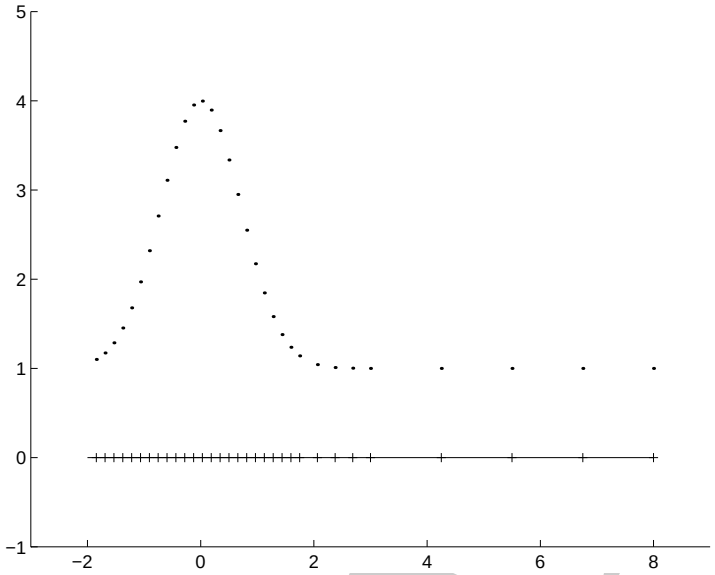

Figura 4.16: Scelta dei nodi da parte di una routine adattativa

Quest'ultima, infatti, a parità di tolleranza richiesta, utilizza generalmente un numero minore di valutazioni della funzione integranda rispetto alla strategia locale. Inoltre, se l'algoritmo termina per aver raggiunto il massimo numero di valutazioni della funzione integranda, è sempre disponibile una stima di $I[f]$. Diversamente, la strategia locale produce una stima dell'integrale della funzione solo fino all'intervallo che ha esaminato per ultimo procedendo dall'estremo inferiore dell'intervallo $[a, b]$ verso quello superiore.

Tuttavia la strategia locale richiede un'occupazione minore di memoria per la pila, in quanto, se essa contiene ad esempio 50 sottointervalli, l'intervallo in testa alla pila ha ampiezza al più $2^{-50}(b-a) \simeq 10^{-15}(b-a)$ che, per usuali valori di $(b-a)$, è un numero minore della precisione relativa dei sistemi floating point utilizzati dalla maggior parte dei calcolatori. In tal caso si ha che la pila non conterrà mai più di 50 intervalli. Per tale motivo la strategia locale ha avuto un maggior successo rispetto alla strategia globale negli anni '70, quando la capacità limitata della memoria dei calcolatori costituiva un serio vincolo alla dimensione dei programmi eseguibili. Con la tecnologia odierna tale problema è superato e la maggior parte degli algoritmi adattativi sono basati sulla strategia globale.

Esaminiamo ora due algoritmi adattativi per la quadratura. Il primo algoritmo, chiamato `Qlocal` (Procedura 4.2), fa uso della strategia locale. In particolare esso utilizza una procedura `Trap(c,d,Q)` che calcola l'approssimazione `Q` di $I[f]$ in un generico intervallo $[c, d]$ mediante la formula trapezoidale. L'algoritmo utilizza tre pile per conservare rispettivamente i valori di `Q` e dei relativi estremi degli intervalli. Nei linguaggi di programmazione dove non sono previste[11], le tre pile possono essere realizzate con

[11] Ad esempio nel linguaggio FORTRAN 77 non è prevista l'allocazione dinamica della memoria e quindi tutte le strutture dati dinamiche come la pila, la lista e la coda non possono essere definite. Tale problema è stato superato nel linguaggio FORTRAN 90 che permette l'allocazione dinamica della

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

array monodimensionali, dove il primo elemento dell'array rappresenta il fondo della pila e l'n-mo ne rappresenta la testa. L'algoritmo effettua inoltre un controllo sul massimo numero di valutazioni della funzione integranda e sulla minima ampiezza di ogni intervallo[12].

memoria come accade nel linguaggio C.

[12]Si tenga presente che, lavorando in un sistema aritmetico a precisione finita, l'ampiezza dell'intervallo, in cui determinare una stima dell'errore e dell'integrale, è, almeno, uguale all'epsilon macchina relativo ad inf, $\epsilon \cdot |inf|$, con inf estremo inferiore dell'intervallo corrente ed ϵ l'epsilon macchina.
Si osserva, comunque, che i controlli descritti non sono gli unici possibili. Numerose routine prevedono ad esempio controlli sul tempo massimo di esecuzione del programma, sull'accumulo degli errori di round-off o sulla dimensione massima delle aree di lavoro utilizzate per le strutture dati.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

procedure Qlocal(**in:** $a, b, Tol, f, Maxvalminlen$; **out:** $Int, Err, Iflag$)

/# SCOPO: calcolo di un integrale definito mediante un algoritmo adattativo locale

/# SPECIFICHE PARAMETRI:

/# PARAMETRI DI INPUT:

var: a	: **reale**	{*primo estremo dell' intervallo di integrazione*}
var: b	: **reale**	{*secondo estremo dell' intervallo di integrazione*}
var: Tol	: **reale**	{*tolleranza da soddisfare*}
var: f	: **reale**	{*funzione integranda*}
var: $Maxval$	: **intero**	{*massimo numero di valutazioni*}
var: $minlen$	: **reale**	{*minima ampiezza dell' intervallo*}

/# PARAMETRI DI OUTPUT:

var: Int	: **reale**	{*stima dell' integrale*}
var: Err	: **reale**	{*stima dell' errore di discretizzazione*}
var: $Iflag$	: **intero**	{*indicatore d' errore*}

/# VARIABILI LOCALI:

var: $Q1$	: **reale**	{*stima dell'integrale mediante la formula*} {*trapezoidale*}
var: $Q2$	: **reale**	{*stima dell'integrale mediante la formula*} {*trapezoidale*}
var: $Locerr$	: **reale**	{*errore di discretizzazione della formula*} {*trapezoidale composita*}
var: $p1(100)$	: **reale**	{*array monodimensionale per la memorizzazione*} {*degli estremi sinistri degli intervalli*}

Procedura 4.2: Algoritmo adattativo locale - continua

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
var: p2(100)   : reale   {array monodimensionale per la memorizzazione}
                         {degli estremi destri degli intervalli}
var: p3(100)   : reale   {array monodimensionale per la memorizzazione}
                         {dei risultati parziali}
var: inf       : reale   {estremo sinistro dell' intervallo corrente}
var: sup       : reale   {estremo destro dell' intervallo corrente}
var: res       : reale   {stima corrente dell' integrale definito}
var: newres    : reale   {stima dell' integrale mediante la formula}
                         {trapezoidale composita}
var: n         : intero  {dimensione delle liste}
var: nval      : intero  {numero di valutazioni di funzione}

/# INIZIO ISTRUZIONI:
    Int := 0.;
    Err := 0.;
    call Trap(a, b, Q1)                    {chiamata della routine}
    n := 1;                                {per la formula trapezoidale}
    nval := 3;
    p1(n) := a;                            {inizializzazione della pila}
    p2(n) := b;
    p3(n) := Q1;
    repeat                                 {ciclo principale di iterazione}
        inf := p1(n);                      {il sottointervallo in testa}
        sup := p2(n);                      {alla pila viene suddiviso}
        res := p3(n);                      {in due parti}
        n := n − 1;
        call Trap(inf, (inf + sup)/2, Q1);
        call Trap((inf + sup)/2, sup, Q2);
        newres := Q1 + Q2;
        Locerr:=|res − newres|/3;
        if (Locerr < Tol(sup − inf)/(b − a)    {test sull'}
```

Procedura 4.2: Algoritmo adattativo locale - continua

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
        or(sup − inf) ≤ minlen)then                {accuratezza locale}
          Int := Int + newres;
          Err := Err + Locerr;
      else
          p1(n + 1) := (inf + sup)/2;              {test sull'accuratezza}
          p2(n + 1) := sup;                        {fallito: i due nuovi}
          p3(n + 1) := Q2;                         {sottodomini vengono}
          p1(n + 2) := inf;                        {inseriti nella pila}
          p2(n + 2) := (inf + sup)/2;
          p3(n + 2) := Q1;
          n =: n + 2;
      endif
      nval := nval + 6;
    until(n = 0   or   nval ≥ Maxval)
    if(n = 0)then                                  {definizione della variabile}
      Iflag := 0;                                  {Iflag}
    else
      Iflag := 1;
    endif
    return Int, Err, Iflag
end Qlocal
```

Procedura 4.2: Algoritmo adattativo locale - fine

Il secondo algoritmo, chiamato `Qglobal` (Procedura 4.3), è invece basato su una strategia globale. Esso utilizza una procedura `quadra(a,b,Q,E)` per la stima del valore di $I[f]$ e dell'errore nell'intervallo $[a, b]$ che può essere realizzata mediante una qualunque coppia di formule di quadratura. Per conservare i sottodomini ancora da esaminare, sono state utilizzate 4 liste contenenti gli estremi degli intervalli (`alist` e `blist`), le stime dell'integrale nei vari sottodomini (`qlist`) e le stime dell'errore (`elist`). Tali liste sono ordinate secondo i valori dell'errore presenti in `elist`, in maniera che le informazioni relative all'intervallo con massima stima dell'errore si trovino sempre nella testa della lista. Anche in questo caso, nei linguaggi dove non sono previste, le liste ordinate possono essere realizzate mediante array monodimensionali dove la testa è rappresentata dall'n-mo elemento dell'array. Una procedura `sort(alist,blist,qlist,elist,n)`

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

è dedicata all'ordinamento. In realtà poichè si è interessati solo all'errore di massimo modulo, la procedura `sort` esegue solo la ricerca del massimo e lo posiziona nell'n-mo elemento delle liste. L'algoritmo termina quando si è raggiunto il massimo numero di valutazioni della funzione integranda o quando la stima dell'errore globale è minore di `Tol`.

procedure Qglobal(**in:** $a, b, Tol, f, Maxval$; **out:** $Int, Err, Iflag$)

/# SCOPO: calcolo di un integrale definito mediante un algoritmo adattativo globale

/# SPECIFICHE PARAMETRI:

/# PARAMETRI DI INPUT:

var: a	: **reale**	{*primo estremo dell' intervallo di integrazione*}
var: b	: **reale**	{*secondo estremo dell'intervallo di integrazione*}
var: Tol	: **reale**	{*tolleranza da soddisfare*}
var: f	: **reale**	{*funzione integranda*}
var: $Maxval$	: **intero**	{*massimo numero di valutazioni*}

/# PARAMETRI DI OUTPUT:

var: Int	: **reale**	{*stima dell' integrale*}
var: Err	: **reale**	{*stima dell' errore di discretizzazione*}
var: $Iflag$	: **intero**	{*indicatore d' errore*}

/# VARIABILI LOCALI:

var: $Q1$	: **reale**	{*prima approssimazione dell'integrale*}
		{*mediante la formula trapezoidale*}
var: $Q2$	: **reale**	{*stima dell'integrale mediante*}
		{*la formula trapezoidale*}
var: c	: **reale**	{*estremo sinistro dell' intervallo corrente*}
var: d	: **reale**	{*estremo destro dell' intervallo corrente*}
var: $Iold$	: **reale**	{*stima dell' integrale definito nel sottointervallo*}
		{*in cui è massima la stima dell' errore*}
var: $Eold$	: **reale**	{*massima stima dell' errore*}

Procedura 4.3: Algoritmo adattativo globale - continua

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
var: E          : reale   {errore di discretizzazione della formula}
                          {trapezoidale, sull' intervallo di integrazione}
var: E1         : reale   {errore di discretizzazione della formula}
                          {trapezoidale, sul sottintervallo sinistro}
var: E2         : reale   {errore di discretizzazione della formula}
                          {trapezoidale, sul sottintervallo destro}
var: alist(100) : reale   {array monodimensionale per la memorizza-}
                          {zione degli estremi sinistri degli intervalli}
var: blist(100) : reale   {array monodimensionale per la memorizza-}
                          {zione degli estremi destri degli intervalli}
var: qlist(100) : reale   {array monodimensionale per la memorizza-}
                          {zione dei risultati parziali}
var: elist(100) : reale   {array monodimensionale per la memorizza-}
                          {zione degli errori parziali}
var: n          : intero  {dimensione delle liste}
var: nval       : intero  {numero di valutazioni di funzione}

/# INIZIO ISTRUZIONI:

    call quadra(a, b, Q1, E)  {prima approssimazione dell'integrale con la}
                              {formula trapezoidale mediante la}
                              {procedura quadra}
    nval := 3;                {Si suppone che la complessita'}
    n := 1;                   {computazionale di quadra sia}
                              {di tre valutazioni di funzione}
    alist(n) := a;            {inizializzazione delle liste}
```

Procedura 4.3: Algoritmo adattativo globale - continua

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
blist(n) := b;
qlist(n) := Q1;
elist(n) := E;
Int := Q1;
Err := E;
while(Err > Toland                                  {ciclo principale di}
  nval < Maxval)do                                  {iterazione con test sull'errore}
   c := alist(n);                                   {il sottointervallo con massimo}
   d := blist(n);                                   {errore viene diviso}
   Iold := qlist(n);                                {in due sottointervalli}
   Eold := elist(n);
   n := n − 1;
   call quadra(c, (c + d)/2, Q1, E1)
   call quadra((c + d)/2, d, Q2, E2)
   Int := Int − Iold + Q1 + Q2;                     {aggiornam. dei valori dell'integrale}
   Err := Err − Eold + E1 + E2;                     {e dell'errore}
   nval := nval + 6;
   alist(n + 1) := c;                               {i due nuovi sottointervalli}
   blist(n + 1) := (c + d)/2;                       {vengono aggiunti alla lista}
   qlist(n + 1) := Q1;
   elist(n + 1) := E1;
   alist(n + 2) := (c + d)/2;
   blist(n + 2) := d;
   qlist(n + 2) := Q2;
   elist(n + 2) := E2;
   n := n + 2;
```

Procedura 4.3: Algoritmo adattativo globale - continua

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
    call sort(alist, blist, qlist, elist, n)      {le liste vengono ordinate}
                                                  {in base agli errori; il}
                                                  {sottointervallo con massimo}
                                                  {errore si trova nell'n-mo nodo}
endwhile
if(nval < Maxval)then                             {definizione della variabile Iflag}
    Iflag := 0;
else
    Iflag := 1;
endif
return Int, Err, Iflag
end Qglobal
```

Procedura 4.3: Algoritmo adattativo globale - fine

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

4.3.4 MATLAB e la quadratura

MATLAB è un *Problem Solving Environment* (PSE), per il calcolo numerico e la visualizzazione di dati.

Originariamente scritto per la risoluzione di problemi di algebra lineare[13], si è evoluto negli anni ed attualmente copre tutti i principali campi della matematica computazionale.

Per l'integrazione numerica, MATLAB mette a disposizione due function: quad e quad8. Entrambe le function sono basate su un algoritmo automatico di tipo adattativo con strategia locale. Come nucleo computazionale quad utilizza la formula di Simpson, mentre quad8 utilizza la formula di Newton-Cotes con 9 nodi[14].

Per far uso della funzione quad è possibile utilizzare una delle funzioni seguenti:

```
>> res = quad('fun',a,b)
>> res = quad('fun',a,b,tol)
>> res = quad('fun',a,b,tol,graf)
```

Tale funzione restituisce nella variabile res il valore calcolato di un integrale definito tra i limiti a e b della funzione fun. Quest'ultima deve essere resa disponibile a quad mediante un file di nome fun.m.

♣ **Esempio 4.8.** Si calcoli:

$$I[f] = \int_0^{2\pi} |\sin(x)|\, dx = 4$$

mediante la funzione quad e si valuti il numero di operazioni floating point effettuate. È necessario creare il file fun.m contenente le operazioni che definiscono la funzione integranda:

```
function y=fun(x)
y = abs(sin(x));
end
```

e quindi eseguire i comandi:

```
>>  a=0.;
>>  b=2*3.1415926;
>>  res = quad('fun',a,b)
    res =
        4.00000207024612
>>  flops
    ans =
        840
```

[13] Il nome sta infatti per MATrix LABoratory.

[14] Le formule di Newton-Cotes sono trattate nel **§1.1.3** del Cap. 1 del **Volume 2**.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

♣

In generale `quad` calcola il valore dell'integrale con una tolleranza pari a 10^{-3}, a meno che non venga specificato il parametro opzionale `tol`.

♣ **Esempio 4.9.** Si calcoli l'integrale precedente, a meno di una tolleranza 10^{-4}. In tal caso è necessario eseguire i comandi:

```
>>  a=0.;
>>  b=2*3.1415926;
>>  tol=10^ (-4);
>>  res = quad('fun',a,b,tol)
    res =
        4.00000013081371
>>  flops
    ans =
        1688
```

Si noti come l'accuratezza maggiore richieda un maggior numero di operazioni floating point. ♣

Infine, se si utilizza il parametro opzionale `graf`, assegnando ad esso un valore diverso da 0, `quad` visualizza un grafico con i valori della funzione nei nodi utilizzati dalla routine per il calcolo dell'integrale. L'utilizzo di `quad8` è analogo a quello di `quad`. Entrambe le funzioni hanno un limite di 10 suddivisioni per ogni sottointervallo onde evitare di raffinare indefinitamente un sottointervallo nel caso in cui esso contenga un' eventuale singolarità. In tal caso viene prodotto il messaggio:

```
Recursion level limit reached in quad.  Singularity likely.
```

che indica la presenza di tale situazione. Conseguenza di ciò è che sia `quad` sia `quad8` non sono in grado di calcolare con efficienza, ad esempio, integrali del tipo:

$$I[f] = \int_0^1 \frac{1}{\sqrt{x}}\, dx$$

♣ **Esempio 4.10.** Si calcoli l'integrale dell'esempio 4.8 nell'intervallo $[0, 5]$ con le due funzioni `quad` e `quad8`. Si valutino il numero di operazioni floating point e si mostrino anche i nodi utilizzati dalla funzione `quad8`. Per tale esempio si ha:

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

```
>>  a=0.;
>>  b=5;
>>  tol=10^ (-4);
>>  graf = 1;
>>  flops(0);
>>  res = quad('fun',a,b,tol)
    Recursion level limit reached
    in quad.  Singularity likely.
    Recursion level limit reached
    in quad.  Singularity likely.
    res =
        3.28366224139591
>>  flops
    ans =
        1116
>>  flops(0);
>>  res = quad8('fun',a,b,tol,graf)
    res =
        3.28366221766898
>>  flops
    ans =
        1903
```

La presenza del parametro `graf` nella chiamata di `quad8` fa sì che, oltre al risultato venga visualizzata anche la Figura 4.17.

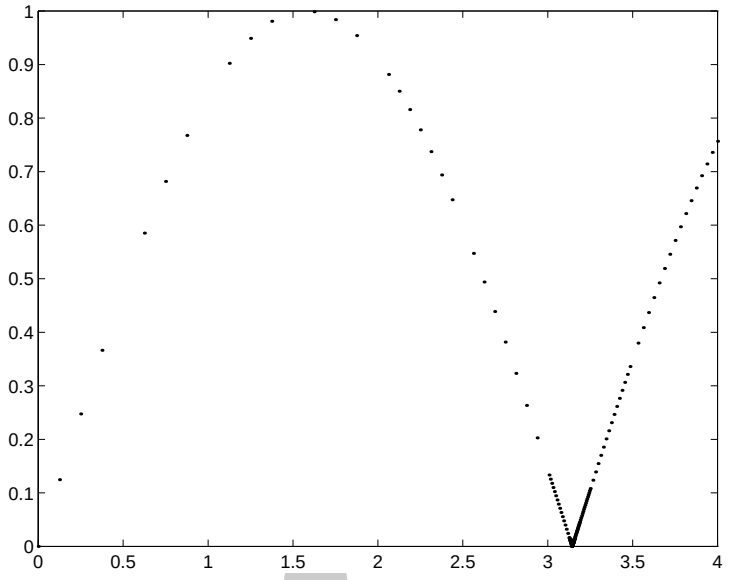

Figura 4.17: Generazione dei nodi da parte della funzione `quad8`

In questo caso la funzione `quad8` utilizza un maggior numero di nodi proprio in corrispondenza di π dove la derivata prima è singolare. Inoltre da un punto di vista computazionale la funzione `quad8` effettua un maggior numero di valutazioni di funzione rispetto a `quad`, ma è in grado di integrare con più accuratezza ed efficienza funzioni aventi discontinuità nella derivata prima. ♣

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

A. Murli

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

4.4 Computer problems sulla quadratura

Esercizio 1 Sviluppare una subroutine Fortran `TRAP1` per il calcolo dell'area sottesa da una funzione data $f(x)$ nell'intervallo $[a, b]$, basata sulla formula trapezoidale composita su m sottointervalli, i cui parametri di input siano gli estremi dell'intervallo a e b, la funzione $f(x)$ ed il numero di sottointervalli m ed il cui parametro di output sia il valore dell'area T.

Dato un cerchio di raggio $r = 1$, si utilizzi la subroutine per il calcolo dell'area della calotta circolare delimitata dalla circonferenza e da un lato del quadrato inscritto del cerchio. Sapendo che l'area della calotta è:

$$Area = \frac{\pi - 2}{4}$$

si utilizzino di seguito $m = 1, 2, 4, 8, ..., 1024$ sottointervalli e si osservi come si riduce l'errore ad ogni passo.

Esercizio 2 Data una parabola intersecante l'asse delle ascisse nei punti $P_1 = (0, 0)$ e $P_2 = (4, 0)$ ed avente come vertice il punto $P_2 = (2, 8)$, ripetere l'esercizio 1 calcolando l'area racchiusa tra la parabola e l'asse delle ascisse. Si ricordi, che Archimede (287 a.c.) ha dimostrato che l'area di un segmento parabolico è uguale a 4/3 dell'area di un triangolo avente la stessa base e la stessa altezza del segmento parabolico.

Esercizio 3 Si ripetano gli esercizi 1 e 2 sostituendo alla formula trapezoidale la formula rettangolare e la formula del punto medio. Si commentino i risultati.

Esercizio 4 Si determini una stima calcolabile dell' errore relativo e si modifichi la subroutine `TRAP1` dell'esercizio 1 in una subroutine `TRAP2` in modo tale da sostituire il parametro di input relativo al numero di sottointervalli m con un parametro di input *tol* che rappresenti la tolleranza sull'errore relativo commesso ed aggiungendo i parametri di output E, che rappresenta una stima dell'errore relativo commesso, e V, che rappresenta il numero di valutazioni di funzioni effettuate.

Si calcoli quindi l'area della calotta circolare dell'esercizio 1 e l'area del segmento parabolico dell'esercizio 2 con tolleranze 0.001, 0.0001 e col la massima accuratezza del sistema aritmetico utilizzato, specificando il numero di valutazioni della funzione integranda.

Esercizio 5 Mediante la subroutine `TRAP2` sviluppata nell'esercizio 4, si calcoli l'area

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

al di sotto delle funzioni di equazione

i)	$y = e^x$	$x \in [0,1]$	(val. esatto = 1.718281 ...)
ii)	$y = x^2 e^{-x^2}$	$x \in [0,4]$	(val. esatto = 0.4431132 ...)
iii)	$y = \log(x)$	$x \in [1,10]$	(val. esatto = 14.02585 ...)
iv)	$y = sin(x^2)$	$x \in [0,5]$	(val. esatto = 0.5279172 ...)
v)	$y = \sqrt{(1-x^2)}$	$x \in [0,1]$	(val. esatto = 1.570796 ...)
vi)	e^{-x^2}	$x \in [-1,1]$	(val. esatto = 1.493648 ...)
vii)	$1/x^2$	$x \in [0.1,1]$	(val. esatto = 9.000000 ...)
viii)	$\frac{sin(e^x+cos(x))}{(1+(1+x)log(1+x))}$	$x \in [0.1,1]$	(val. esatto = 1.5782736 ...)

con tolleranze 0.001, 0.0001 e con la massima accuratezza del sistema aritmetico floating-point utilizzato. Si fornisca anche una stima dell'errore relativo commesso e del numero di valutazioni di funzioni effettuate.

Esercizio 6 Si sviluppi un integratore automatico `ADALOC` basato su un algoritmo adattativo con strategia locale che faccia uso della formula trapezoidale e si ripeta l'esercizio 5, confrontando il numero di valutazioni di funzioni effettuate dalle subroutine `TRAP2` e `ADALOC`.

Esercizio 7 Si sviluppi un integratore automatico `ADAGLO` basato su un algoritmo adattativo, con strategia globale, che faccia uso della formula trapezoidale e si ripeta l'esercizio 5, confrontando il numero di valutazioni di funzioni effettuate dalle subroutine `TRAP2`, `ADALOC` e `ADAGLO`.

Esercizio 8 Utilizzare le subroutine `ADALOC` e `ADAGLO` per calcolare l'area al di sotto delle funzioni nell'intervallo $[0,1]$:

$$f_1(x) = x^2(1.2-x)(1-e^{\cdot 2(x-1)}) \quad \text{val. esatto} = 0.00954996...$$
$$f_2(x) = x^{0.1}(1.2-x)(1-e^{20(x-1)}) \quad \text{val. esatto} = 0.602297...$$

utilizzando la tolleranza $tol = 0.001$ e la massima accuratezza del sistema aritmetico utilizzato. Si confronti il numero di valutazioni di funzioni effettuate.

Esercizio 9 Ripetere l'esercizio 8, fornendo in output, la dimensione massima della pila utilizzata dalla subroutine `ADALOC` e della lista utilizzata dalla subroutine `ADAGLO`.

Esercizio 10 Mediante la subroutine `ADALOC` si ripeta l'esercizio 8 fornendo, in output, gli estremi degli intervalli in cui la stima dell'errore soddisfa il criterio di tolleranza locale. Si raffronti il risultato con il grafico delle funzioni.

Esercizio 11 Si ripeta l'esercizio 5 utilizzando le funzioni matlab `QUAD` e `QUAD8`. Si valuti la complessità computazionale con la funzione `FLOPS`

Esercizio 12 Si scriva una funzione `MATLAB ADAGLO` basata sulla strategia adattativa globale che utilizzi la formula del punto medio. Tale funzione, oltre a fornire una stima del valore dell'area al di sotto della funzione data e dell'errore relativo commesso, disegni anche il grafico della funzione con i nodi utilizzati dall'algoritmo. Si calcolino nuovamente le aree dell'esercizio 8.

A. Murli, *Matematica Numerica: metodi, algoritmi e software*
Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.

Bibliografia

[1] Chawla M. - *Convergence of Newton-Cotes quadratures for analytic functions* - BIT, vol. 11 (1971), pp. 159–167

[2] Davis P. e Rabinowitz P. - *Methods of numerical integration* - Academic Press Inc. (1984)

[3] Delves L. and Mohamed J. - *Computational methods for integral equations* - Cambridge University Press (1985)

[4] Engels P. - *Numerical quadrature and cubature* - Academic Press Inc. (1980)

[5] Evans G. - *Practical Numerical Integration* - Wiley (1993)

[6] Fox J. et al. - *Solving problems on concurrent processors* - vol. 1, Prentice Hall, 1988.

[7] Gautschi W. e Inglese G. - *Lower bounds for the condition number of Vandermonde matrices* - Numer. Math. vol. 52 (1988), pp. 241-250

[8] Genz A. e Malik A. - *An inbedded family of fully symmetric numerical integration rule* - SIAM J. Num. Anal., vol. 20, No.3, June 1983, pp. 580–588

[9] Hammersley J. e Handscomb D. - *Monte Carlo methods* - Methuen, 1964

[10] Krommer A. e Ueberhuber C. - *Numerical Integration on advanced computer systems* - Springer Verlag (1994)

[11] Krommer A.R. e Ueberhuber Ch.W. - *Computational integration* - SIAM (1998)

[12] Kronrod A. - *Nodes and weights of quadrature formulas* - Consultants bureau, 1965

[13] Krylov V. - *Approximate calculation of integrals* - Mc Millan Comp. (1962)

[14] Malcom L. e Simpson R. - *Local versus Global strategies for adaptive quadrature* - ACM Trans. Math. Soft., vol. 1 (1975), pp. 129–146

[15] Math Works Inc. - `MATLAB` - *Reference guide* - 1992

[16] Monegato G. - *Stieltjes polynomials and related quadrature rules* - SIAM Review, vol. 24 (1982), pp. 137–158

A.Miali

[17] Monegato G. - *Positivity of weights of extended Gauss-Legendre quadrature rules* - Math. Comp (1978)

[18] Numerical Algorithm Group - *NAg library, Mark 17* - NAg - Oxford (1996)

[19] Numerical Algorithm Group - *NAg Numerical Parallel library, release 2* - NAg - Oxford (1997)

[20] Piessens R. et al - *QUADPACK, a subroutine package for automatic integration* - Springer Verlag (1983)

[21] Sloan I. e Joe S. - *Lattice Methods for Multiple Integration* - Clarendon Press 1994

[22] Szegö G. - *Ortogonal polynomials* - American Math. Soc., 1939

[23] Stroud A.H. - *Approximate calculation of multiple integrals* - Prentice Hall (1971)

[24] Visual Numerics - *IMSL Math Library version 2* - 1993

[25] Ueberhuber Ch.W. - *Numerical Computation 2, Methods Software and Analysis* - Springer (1997)

A. Murli

Versione in corso di stampa, solo per uso personale, soggetta ad errori.
Non è autorizzata la diffusione. Tutti i diritti riservati.