1 Lezione del 05-05-25

Continuiamo la trattazione dell'approssimazione di integrali attraverso le formule di quadratura interpolatorie. Un problema che avevamo fino adesso è che se h è grande, allora $E_n(f)$ è grande per qualche termine proporzionale a h^d, dove d è il grado dell'errore.

Un'idea potrebbe essere di aumentare n, quindi aumentare i punti campionati per ridurre il passo h, ma come abbiamo visto questo è instabile (risente di fenomeni simili al fenomeno di Runge).

1.0.1 Formule di Newton-Cotes generalizzate

Decidiamo quindi di spezzare l'integrale su sottointervalli, e in ciascuno di essi applicare una formula di quadratura. Questo metodo ci porterà alle formule di Newton-Cotes **generalizzate**, anche dette *composite*.

Quindi, posto ad esempio un certo intervallo $[a, b]$ contenente i punti ordinati c, d, e, dividiamo l'integrale:

$$\int_a^b f(x)\,dx = \int_a^c f(x)\,dx + \int_c^d f(x)\,dx + \int_d^e f(x)\,dx + \int_e^b f(x)\,dx$$

e si applica la formula di quadratura ad ogni sottointervallo, sommando.

Più formalmente, avremo che si divide $[a, b]$ in L sottointervalli equispaziati, con:

$$x_0 = a, \quad x_1 = x_0 + \frac{b-a}{L}, \quad ..., \quad x_L = b$$

e si divide l'integrale come:

$$\int_a^b f(x)\,dx = \sum_{i=1}^{L} \int_{x_{i-1}}^{x_i} f(x)\,dx$$

In ogni intervallo $[x_{i-1}, x_i]$ si applica quindi una formula di Newton Cotes con $n+1$ nodi.

Avremo quindi bisogno, in ogni sottointervallo, di:

$$n + 1 - 2 = n - 1$$

nodi aggiuntivi oltre agli estremi dell'intervallo per poter applicare la formula di quadratura $J_n(f)$. Il numero di nodi totali dovrà quindi essere:

$$L + 1 + (n-1) \cdot L = nL + 1$$

Osserviamo che il numero di nodi corrisponde al numero di valutazioni di f, e quindi in genere è un'indicazione del costo computazionale di una formula.

1.0.2 Esempio: formula dei trapezi generalizzata

Prendiamo la forma che otteniamo per $n = 1$:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{L} \frac{b-a}{2L}\left(f(x_{i-1}) + f(x_i)\right) = \frac{b-a}{2L}\sum_{i=1}^{L}\left(f(x_{i-1}) + f(x_i)\right)$$

$$= \frac{b-a}{2L}\left(f(x_0) + 2\sum_{i=1}^{L-1} f(x_i) + f(x_L)\right) = J_1^{(G)}(f)$$

1.0.3 Esempio: formula di Cavalieri-Simspon

Vediamo quindi la generalizzazione della formula di Cavalieri, cioè ciò che otteniamo per $n = 2$:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{L} \frac{b-a}{6L}\left(f(x_{i-1}) + 4f\left(\frac{x_{i-1}+x_i}{2}\right) + f(x_i)\right)$$

$$= \frac{b-a}{6L}\left(f(x_0) + 2\sum_{i=1}^{L-1} f(x_i) + 4\sum_{i=1}^{L} f\left(\frac{x_{i-1}+x_i}{2}\right) + f(x_L)\right) = J_2^{(G)}(f)$$

1.0.4 Errore nelle formule generalizzate

In ogni intervallo della forma $[x_{i-1}, x_i]$ conosciamo l'errore, che è quello della formula di Newton-Cotes che stiamo utilizzando:

$$e \cdot h^d \cdot f^{(d-1)}(\varepsilon), \quad f \in C^{(d-1)}([a,b])$$

L'errore sulla formula totale $J_n^{(G)}(f)$ sarà quindi:

$$E_n^{(G)}(f) = I(f) - J_n^{(G)}(f) = \sum_{i=1}^{L} c \cdot h^d \cdot f^{(d-1)}(\varepsilon_i), \quad \varepsilon_i \in [x_{i-1}, x_i]$$

Vediamo quindi che l'unico termine dipendente dall'intervallo è $f^{(d-1)}(\varepsilon)$, per cui possiamo dire:

$$= \sum_{i=1}^{L} c \cdot h^d \cdot L\frac{f^{(d-1)}(\varepsilon_i)}{L} = ch^d L \cdot f^{(d-1)}(\varepsilon), \quad \varepsilon \in [a,b]$$

sfruttando il teorema della media integrale.

Abbiamo quindi gli errori:

- **Formula dei trapezi:**

$$E_1^{(G)}(f) = -\frac{(b-a)^3}{12L^2} \cdot f''(\varepsilon)$$

- **Formula di Cavalieri-Simpson:**

$$E_2^{(G)}(f) = -\frac{(b-a)^5}{2880L^4} \cdot f^{(4)}(\varepsilon), \quad \varepsilon \in [a,b]$$

e il caso generale:

$$E_n^{(G)}(f) = \begin{cases} c_n \cdot \frac{(b-a)^{n+2}}{L^{n+1}} \cdot f^{(n+1)}(\varepsilon), & n \text{ dispari} \\ c_n \cdot \frac{(b-a)^{n+3}}{L^{n+2}} \cdot f^{(n+2)}(\varepsilon), & n \text{ pari} \end{cases}$$

1.1 Formule di quadratura sulle derivate

Potremmo considerare formule di quadratura che coinvolgono le derivate di f. Si possono quindi generalizzare i concetti di grado di precisione, scelta dei nodi e pesi ottimali anche per formule di quadratura del tipo:

$$\int_a^b f(x)\rho(x)\,dx = \sum_{i=0}^{n_1} f(x_i) + \sum_{i=0}^{n_2} f'(y_i) + \sum_{i=0}^{n_3} f''(z_i)$$

chiaramente restringendo l'applicazione delle formule a funzioni sufficientemente derivabili.

Vediamo ad esempio come trovare i pesi ottimi per l'approssimazione con:

$$\int_0^1 f(x)\,dx = I(f) \approx a_1 f(0) + a_2 f'\left(\frac{1}{3}\right) + a_3 f'\left(\frac{2}{3}\right) + f(4)$$

sull'intervallo $[a, b] = [0, 1]$.

Imponendo errore nullo ai primi 4 gradi si otterrà:

$$\begin{cases} E_1(1) = 0 \\ E_1(x) = 0 \\ E_1(x^2) = 0 \\ E_1(x^3) = 0 \end{cases} \implies \begin{cases} \int_0^1 1\,dx = 1 = a_1 + a_4 \\ \int_0^1 x\,dx = \frac{1}{2} = a_2 + a_3 + a_4 \\ \int_0^1 x^2\,dx = \frac{1}{3} = \frac{2}{3}a_2 + \frac{4}{3}a_3 + a_4 \\ \int_0^1 x^3\,dx = \frac{1}{4} = \frac{1}{3}a_2 + \frac{4}{3}a_3 + a_4 \end{cases}$$

Ricaviamo allora dalle prime 3:

$$\begin{cases} a_4 = 1 - a_1 \\ a_2 = \frac{1}{2} - a_3 - a_4 \\ a_3 = \frac{3}{4}\left(\frac{1}{3} - \frac{2}{3}a_2 - a_4\right) = \frac{1}{4} - \frac{1}{2}a_2 - \frac{3}{4}a_4 \end{cases}$$

da cui sostituendo ulteriormente:

$$\begin{cases} a_4 = 1 - a_1 \\ a_2 = \frac{1}{2} + \frac{1}{2}a_4 - a_4 = \frac{1}{2} - \frac{1}{2}a_4 \\ a_3 = \frac{1}{4} - \frac{1}{2}\left(\frac{1}{2} - a_3 - a_4\right) - \frac{3}{4}a_4 = \frac{1}{2}a_3 - \frac{1}{4}a_4 \end{cases}$$

sostituendo tutto nella quarta si ha:

$$\frac{1}{4} = \frac{1}{3}\left(\frac{1}{2} - \frac{1}{2}a_4\right) + \frac{4}{3}\cdot -\frac{1}{2}a_4 + a_4 = \frac{1}{6} + \frac{1}{6}a_4 \implies a_4 \frac{1}{2}$$

da cui immediatamente:

$$a_1 = \frac{1}{2}, \quad a_2 = \frac{1}{4}, \quad a_3 = -\frac{1}{4}, \quad a_4 = \frac{1}{2},$$

da cui:

$$J_3(f) = \frac{f(0)}{2} + \frac{f'\left(\frac{1}{3}\right)}{4} - \frac{f''\left(\frac{2}{3}\right)}{4} + \frac{f(1)}{2}$$

Per determinare il grado di precisione vediamo l'errore per x^4:

$$E_3(x^4) = \int_0^1 x^4\,dx - \left(\frac{1}{2}\cdot 0 + \frac{1}{4}\cdot 4 \cdot \left(\frac{1}{3}\right)^3 - \frac{1}{4}\cdot 4 \cdot \left(\frac{2}{3}\right)^3 + \frac{1}{2}\right) = \frac{1}{5} - \left(\frac{1}{27} - \frac{8}{27} + \frac{1}{2}\right) = \frac{1}{5} - \frac{13}{54} \neq 0$$

da cui il grado di precisione è esattamente 3.

1.2 Implementazione MATLAB dei metodi di approssimazione di integrali

Possiamo quindi implementare quanto abbiamo studiato finora in funzioni MATLAB che ci permettano di approssimare gli integrali di certi function handle su dati intervalli.

Definiamo in particolare una funzione di approssimazione composita di grado 1 (metodo dei trapezi) e 2 (metodo di Cavalieri-Simpson).

1.2.1 Implementazione MATLAB del metodo dei trapezi

Potremo implementare direttamente la funzione descritta in 18.0.2, con un pò di logica aggiuntiva per tracciare un grafico delle rette approssimazione, come:

```
function j = trapez_int(f, a, b, L)
    xi = linspace(a, b, L + 1)';
    yi = f(xi);

    % stampa
    clf;
    hold on;

    % funzione
    fplot(f, [a, b], 'r');

    % punti
    plot(xi, yi, 'ob');

    % rette
    for i = 1:L
        x0 = xi(i); x1 = xi(i + 1);
        y0 = yi(i); y1 = yi(i + 1);

        m = (y1 - y0) / (x1 - x0);
        r = @(x) m * (x - x0) + y0;

        fplot(r, [x0, x1], 'b')
    end

    hold off;

    % integra
    j = yi(1) + yi(L + 1) + 2 * sum(yi(2:L));
    j = j * (b - a) / (2 * L);
end
```

1.2.2 Implementazione MATLAB del metodo di Cavalieri-Simpson

Anche qui otremo implementare direttamente la funzione descritta in 18.0.3, con un pò di logica aggiuntiva per tracciare un grafico delle parabole di approssimazione, come:

```
function j = simpson_int(f, a, b, L)
    xi = linspace(a, b, L + 1)';
    pi = xi(1:L) + (b - a) / (2 * L);
    yi = f(xi);
    ypi = f(pi);

    % stampa
    clf;
    hold on;

    % funzione
    fplot(f, [a, b], 'r');

    % punti
    plot(xi, yi, 'ob');

    % parabole
    for i = 1:L
```

```
        x0 = xi(i); x1 = xi(i + 1);
        p = pi(i); yp = ypi(i);
        y0 = yi(i); y1 = yi(i + 1);

        p = polyfit([x0, p, x1], [y0, yp, y1], 2);
        q = @(x) polyval(p, x);

        fplot(q, [x0, x1], 'b')
    end

    hold off;

    % integra
    j = yi(1) + 2 * sum(yi(2:L)) + 4 * sum(ypi) + yi(L + 1);
    j = j * (b - a) / (6 * L);
end
```

dove notiamo che manteniamo esplicitamente i punti intermedi separati dai punti estremi degli intervalli di integrazione, ergo L determina il numero di intervalli di integrazione.

Potremo quindi provare gli algoritmi, ad esempio sulla funzione:

$$f(x) = \sin(x) + \tan\left(\frac{x}{15}\right) + e^{-x}$$

sull'intervallo $[a, b]] = [0, 15]$.

Iniziamo col definire le variabili e un valore di *ground truth*, usando la `integral()` fornita da MATLAB:

```
>> f = @(x) sin(x) .* tan(x / 15) + exp(-x); % definiamo f
>> a = 0; b = 15;
>> i = integral(f, a, b)

i =

    2.2912
```

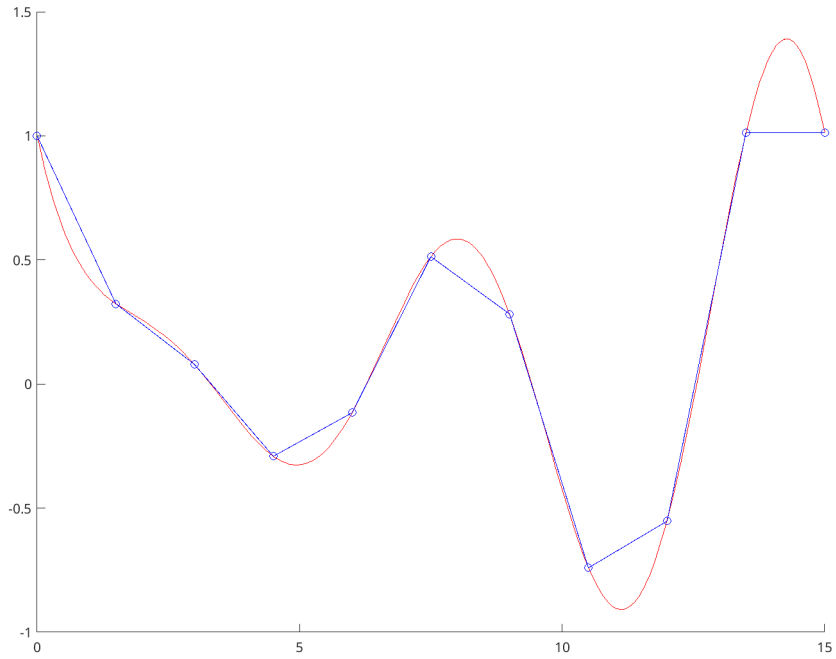
Potremo quindi provare il metodo dei trapezi come:

```
>> jt = trapez_int(f, a, b, 10)

jt =

    2.2734
```

Da cui otteniamo un valore di $i - j_t \approx 0.0178$, e il grafico:



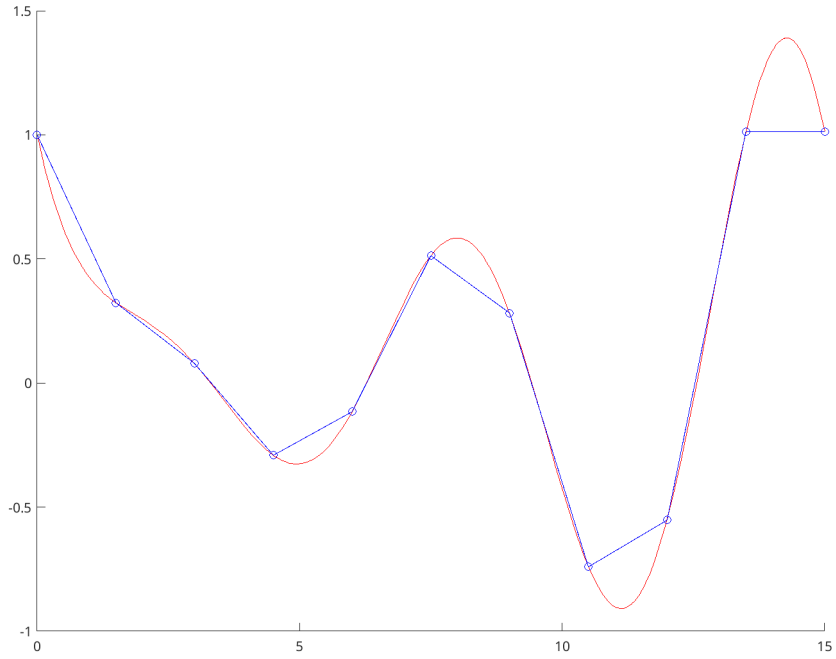
Vediamo quindi se l'approssimazione col metodo di Cavalieri-Simpson è più accurata:

```
>> js = simpson_int(f, a, b, 10)

js =

    2.2940
```

In questo caso otteniamo il valore $i - j_s \approx -0.0028$, e il grafico:



cioè notiamo che sia il grafico che l'appossimazione integrale sono molto pià vicini alla funzione originale.