Lezioni di Calcolo Numerico
Lezione 03: Approssimazione di dati e funzioni

Alberto Tibaldi

5 maggio 2018

Indice

Introduzione

Nell'introduzione avevamo discusso il calcolo dell'integrale definito

$$I = \int_4^7 \mathrm{e}^{-x^2}\,\mathrm{d}x \tag{1}$$

e avevamo capito che non ci sono limitazioni *filosofiche* che ci impediscono di calcolarlo: il problema è unicamente la nostra incapacità di scrivere la primitiva in termini di funzioni elementari. Si tratta dunque di una limitazione legata al tipo di procedura di calcolo: cercare in tutti i modi di applicare il teorema fondamentale del calcolo integrale a una primitiva scritta in forma chiusa. Immaginiamo a questo punto di cambiare tattica: se non è possibile integrare una funzione $f(x)$, potremmo cercare di sostituirla con un'altra funzione $\widetilde{f}$, che in qualche modo *le assomigli* o, per usare il titolo della lezione, *la approssimi*. Se le due funzioni si assomigliano, evidentemente anche i loro integrali si assomiglieranno! Un problema differente, ma che comunque è riconducibile a quanto appena descritto, è l'approssimazione di **dati**, ovvero coppie (x_i, y_i) fornite per esempio da misure sperimentali o osservazioni di un qualche tipo.

Purtroppo, tra il dire e il fare c'è sempre di mezzo il mare: approssimare una funzione o un insieme di dati, in effetti, potrebbe essere più difficile di quanto sembri. Per capire quali possono essere le problematiche, guardiamo l'esempio di Fig. 1. La curva blu riporta la funzione di Runge

$$f(x) = \frac{1}{1+x^2}, \tag{2}$$

disegnata[1] su una griglia costituita da 1001 punti. Volendo cercare di approssimarla, il primo passo è valutare questa funzione in alcuni valori delle ascisse, che chiameremo *nodi*. In questo

[1] lo script MATLAB® utilizzato per disegnare questa figura è stato riportato in Appendice A

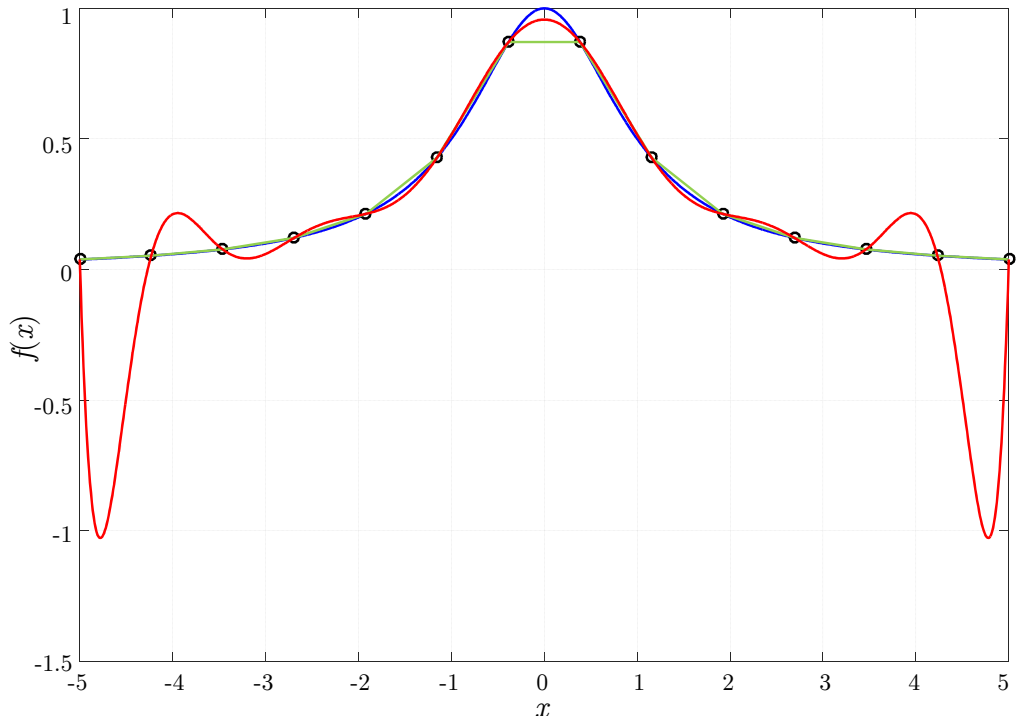

Figura 1: Tentativi di interpolazione della funzione di Runge, disegnata con una curva blu. I 14 nodi di interpolazione equidistanti sono stati rappresentati mediante dei marker circolari neri. Il risultato dell'interpolazione lineare è rappresentato mediante una curva verde. La curva rossa riporta il polinomio interpolante.

esempio si è scelto di usare 14 nodi equispaziati, marcati corrispettivamente ai relativi valori della funzione con dei cerchietti neri. Al fine di ottenere quindi un'approssimazione della funzione f, vogliamo utilizzare il **criterio dell'interpolazione**: definire la nostra funzione approssimata $\widetilde{f}$ in modo tale che **passi per ciascuno dei cerchietti**. Ci sono diversi modi per ottenere un obiettivo del genere. Per esempio, la curva verde mostra il risultato di un'**interpolazione lineare**; in parole povere, unire ciascuna coppia di cerchietti adiacenti mediante un segmento di retta. In alcune regioni dell'intervallo mostrato questo approccio sembra essere vincente, dal momento che non si riesce a distinguere la curva verde da quella blu, originale. Tuttavia, questo approccio si dimostra inefficace in regioni dove la funzione varia rapidamente, per esempio in prossimità di $x = 0$, in cui si ha una forma molto *tondeggiante* che viene approssimata mediante un singolo segmento orizzontale. Volendo cercare di rappresentare meglio queste regioni *tondeggianti*, si può immaginare di sostituire questo approccio con un'**interpolazione polinomiale**, ovvero in cui si utilizza come funzione approssimante un polinomio *progettato* in modo tale da passare per i vari nodi di interpolazione; sfruttando quindi il fatto che i polinomi di grado elevato sono già di per sé funzioni rapidamente variabili (si pensi per esempio a una semplice parabola: ha una certa concavità in prossimità del vertice!). La curva rossa, però, dimostra che anche questa scelta si può rivelare problematica: per quanto infatti il polinomio passi per ciascuno dei nodi di interpolazione, è impossibile dire che *assomigli* alla funzione di partenza! Esso infatti interpola, ma tra un nodo e un altro presenta un comportamento completamente fuori controllo! Nel testo che segue, studieremo due possibili approcci atti a interpolare delle funzioni:

- l'interpolazione polinomiale, ossia, approssimare mediante un **unico polinomio** di grado n la funzione sull'intero intervallo;
- l'interpolazione mediante funzioni polinomiali a tratti (spline), ovvero, usare come interpolanti funzioni di ordine ridotto e raccordate mediante condizioni di regolarità.

1 Approssimazione polinomiale

In questa sezione ci concentreremo sul formalizzare quanto anticipato nell'introduzione effettuando approssimazione polinomiale, ossia, cercando di approssimare una funzione f in un intervallo mediante un singolo polinomio; questo non è per esempio il caso della curva verde di Fig. 1, dove sono usate segmenti di retta diversi su ciascun intervallo. A tal fine, è opportuno introdurre un po' di notazione. In questo capitolo noi scriveremo un generico polinomio di grado n nella forma

$$c_1 x^n + c_2 x^{n-1} + c_3 x^{n-2} + \ldots + c_n x + c_{n+1}. \tag{3}$$

Dire che il polinomio è di grado n implica che esiste un monomio che è elevato a potenza pari a n. Il polinomio è dato da una combinazione lineare di monomi pesati per opportuni coefficienti $\{c_i\}$. In questo testo, si sceglie di nominare c_1 l'indice del polinomio di grado massimo, e c_{n+1} il termine noto del polinomio, ovvero il coefficiente del monomio di grado zero[2]. Il nostro scopo è scegliere un polinomio che sia in grado di approssimare la nostra funzione di partenza f. Questo, in altre parole, significa **scegliere i coefficienti $\{c_i\}$ che permettano di effettuare questa approssimazione**[3].

1.1 Un esempio di approssimazione polinomiale

Al fine di chiarire di cosa si sta parlando, è opportuno ricordare che lo studente reduce di Analisi Matematica I ha già avuto a che fare, nella propria vita, con una tecnica di approssimazione polinomiale: lo sviluppo di Taylor! In particolare, l'approssimazione di Taylor si prepone come obiettivo trovare una funzione $\widetilde{f}$ che *assomigli* alla f non ovunque, ma solo nelle vicinanze di un certo punto x_0; per questo motivo, il corrispondente polinomio di Taylor si può indicare come Tf_{n,x_0}, specificando il grado n e il punto di sviluppo x_0. Quindi, è lecito scriverlo nella forma

$$f \simeq \widetilde{f} = Tf_{n,x_0} = c_{n+1} + c_n(x-x_0) + c_{n-1}(x-x_0)^2 + \ldots + c_2(x-x_0)^{n-1} + c_1(x-x_0)^n = \sum_{k=0}^{n} c_{n+1-k}(x-x_0)^k. \tag{4}$$

Dire che il polinomio di Taylor *deve assomigliare alla funzione solo vicino a un punto*, significa richiedere che f e $\widetilde{f}$, ovvero il polinomio, soddisfino le seguenti condizioni:

$$\begin{aligned} f(x_0) &= \widetilde{f}(x_0) \\ f'(x_0) &= \widetilde{f}'(x_0) \\ f''(x_0) &= \widetilde{f}''(x_0) \\ &\vdots \\ f^{(n)}(x_0) &= \widetilde{f}^{(n)}(x_0), \end{aligned} \tag{5}$$

ovvero, *che tutte le derivate siano uguali intorno a $x = x_0$, ignorando cosa capiti lontano da qui.* Imponendo le condizioni (5) in (4), è possibile ottenere[4]

$$Tf_{n,x_0} = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \frac{f'''(x_0)}{3!}(x-x_0)^3 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n. \tag{6}$$

Al fine di chiarire il concetto senza sviluppare troppi conti, si consideri per esempio $x_0 = 0$; in questa situazione, l'espressione (6) diventa

$$Tf_{n,0} = \underbrace{f(0)}_{c_{n+1}} + \underbrace{\frac{f'(0)}{1!}}_{c_n} x + \underbrace{\frac{f''(0)}{2!}}_{c_{n-1}} x^2 + \underbrace{\frac{f'''(0)}{3!}}_{c_{n-2}} x^3 + \ldots + \underbrace{\frac{f^{(n)}(0)}{n!}}_{c_1} x^n, \tag{7}$$

[2] questa scelta può sembrare del tutto controintuitiva, ma sarà molto più chiara in seguito; volendole dare una qualche giustificazione, si può dire che c_1 è il coefficiente del monomio *più significativo*: quello di grado più elevato, quindi *più importante* da certi punti di vista

[3] infatti, un polinomio è determinato in modo univoco dai coefficienti dei vari monomi

[4] la dimostrazione non è qui proposta poiché non strettamente collegata al Calcolo Numerico

in cui è immediato identificare c_{n+1} come il termine noto, c_n come il termine che moltiplica x elevato a potenza 1, c_{n-1} che moltiplica x^2, e così via. Riassumendo, imporre i vincoli (5) in (4) permette di trovare le espressioni dei coefficienti $\{c_i\}$ elencate in (7) e, quindi, di determinare il polinomio approssimante in questione.

1.2 Il criterio dell'interpolazione

Come si è già detto, il criterio di Taylor per la determinazione dei coefficienti era: approssimare benissimo la funzione localmente, ossia intorno a un certo punto $x = x_0$, ma fregarsene completamente di cosa capiti lontano da esso. Certamente, una delle conseguenze di approssimare mediante polinomi di Taylor di ordine elevato è anche aumentare l'ampiezza dell'intorno in cui l'approssimazione è valida, ma questo è un *effetto collaterale*, per quanto positivo. Il nostro obiettivo per questa sezione è ottenere, al posto di qualcosa di *mozzafiato* intorno a un singolo punto, un'approssimazione $\widetilde{f}$ della funzione f che sia *decente* in un intervallo abbastanza largo.

A tal fine, ripensiamo un momento alle condizioni (5); richiedere qualcosa di *decente* può voler dire per esempio richiedere che solo la prima delle condizioni, ossia l'uguaglianza della funzione approssimata $\widetilde{f}$ e della funzione di partenza f, sia soddisfatta. Però, al fine di garantire che questa condizione venga soddisfatta su tutto un intervallo, possiamo decidere di partizionare il nostro intervallo in $n+1$ punti $\{x_i\}_{i=1,\dots,n+1}$ e, per ciascuno di essi, valutare la funzione $y_i = f(x_i)$. Quindi, perché questa condizione di *decenza* valga su tutto l'intervallo, richiederemo che

$$\widetilde{f}(x_i) = f(x_i) = y_i, \qquad i = 1, \dots, n+1.$$

Questo criterio è detto **criterio dell'interpolazione**: anziché imporre condizioni sulle derivate, si chiede che, una volta partizionato l'intervallo di interesse in $n+1$ nodi, la funzione approssimante $\widetilde{f}$ sia uguale alla funzione di partenza f in ciascuno dei nodi, ossia, che **passi per essi**[5]. Questo per esempio è il caso della curva rossa di Fig. 1, che, nonostante tutti i problemi che può esibire, passa per ciascuno dei punti marcati coi cerchietti neri.

2 Interpolazione polinomiale

2.1 Unicità (ed esistenza?) del polinomio interpolante

Al fine di introdurre l'interpolazione polinomiale, consideriamo il seguente teorema.

Assegnati $n+1$ dati (x_i, y_i), $i = 1, \dots, n+1$, esiste uno e un solo polinomio $p_n(x)$ di grado minore o uguale a n interpolante i dati assegnati, ovvero soddisfacente le condizioni di interpolazione

$$p_n(x_i) = y_i, \qquad i = 1, \dots, n+1. \tag{8}$$

Cercando di parafrasare il testo del teorema, questo ci garantisce che è sempre possibile trovare un polinomio $p_n(x)$ che sia in grado di passare per i punti (x_i, y_i). Il teorema però ci dà anche la garanzia che non solo questo polinomio esiste, ma anche che è unico, ossia, che non possono esistere due polinomi in grado di passare attraverso questi punti.

Questo è un teorema di esistenza e unicità, quindi la dimostrazione si articola in due parti: quella di unicità, e quella di esistenza. Per quanto riguarda l'esistenza, essa si può dimostrare banalmente trovando un polinomio che soddisfi la condizione di interpolazione (8). A questo fine, più avanti mostreremo un modo di costruire polinomi di questo tipo. Per quanto riguarda l'unicità, possiamo procedere per assurdo: cerchiamo di effettuare un ragionamento corretto applicato a un'ipotesi scorretta, e questo dovrebbe prima o poi portarci alla contraddizione di un principio primo. Visto che vogliamo verificare l'unicità, ipotizziamo per assurdo che esistano due polinomi interpolanti diversi tra loro, $p_n(x)$ e $q_n(x)$. Se entrambi sono interpolanti, allora entrambi devono soddisfare, per $i = 1, \dots, n+1$, le condizioni di interpolazione

$$\begin{aligned} p_n(x_i) &= y_i \\ q_n(x_i) &= y_i. \end{aligned}$$

[5] si noti che ovviamente questo approccio è applicabile, senza alcuna modifica, anche al caso in cui le $n+1$ coppie di valori (x_i, y_i) vengano non da una funzione matematica, ma da un esperimento di laboratorio o generici dati

A questo punto, costruiamo la differenza di questi polinomi, ovvero

$$p_n(x) - q_n(x);$$

questa differenza:

- è un polinomio, dal momento che la somma/sottrazione di polinomi è ancora un polinomio;
- è di grado al più n, dal momento che la somma/sottrazione di polinomi al più di grado n non può certamente far crescere in grado;
- non è nullo, dal momento che per ipotesi stiamo dicendo che $p_n(x)$ deve essere diverso da $q_n(x)$;
- ha $n+1$ zeri; infatti, se ci ricordiamo le ipotesi sulle condizioni di interpolazione appena scritte,

$$\underbrace{p_n(x_i)}_{y_i} - \underbrace{q_n(x_i)}_{y_i} = 0, \qquad i = 1, ..., n+1.$$

Tuttavia, abbiamo appena trovato un paradosso: un polinomio di grado n ha n radici, come suggerisce il teorema fondamentale dell'algebra; queste possono essere reali o complesse, ma devono essere esattamente pari a n. Dal momento che un ragionamento corretto applicato all'ipotesi iniziale ci ha portato a una contraddizione, il polinomio interpolante deve essere unico.

2.1.1 Interpretazione geometrica del teorema di esistenza e unicità del polinomio interpolante

Il teorema che abbiamo appena enunciato e dimostrato è in realtà la generalizzazione di un concetto molto famoso.

> «Dati due punti distinti su un piano, per essi passa una e una sola retta.»

In effetti, quando abbiamo due punti, $n+1 = 2$, e quindi $n = 1$. Come ben noto, una retta si può scrivere come un polinomio di primo grado nella forma

$$f(x) = mx + q.$$

Questo, e il fatto che per due punti passa una e una sola retta, sono dimostrati dal teorema che stiamo studiando. Ma c'è dell'altro!

> «Dati tre punti distinti e non collineari su un piano, per essi passa una e una sola parabola.»

Di nuovo, stiamo dicendo più o meno la stessa cosa! Se ho $n+1 = 3$, allora $n = 2$, e quindi la forma del generico polinomio di grado 2 è

$$y = ax^2 + bx + c,$$

che è proprio l'equazione della parabola. Il teorema in realtà prevede anche il caso di tre punti collineari; infatti, il polinomio interpolante è unico, ma ha grado **al più** pari a n; se quindi i tre punti sono collineari, il polinomio interpolante sarà ancora una retta!

2.2 Rappresentazione monomiale del polinomio di interpolazione

La sezione precedente propone argomentazioni riguardo l'unicità del polinomio interpolante, ma non fornisce alcuna indicazione né riguardo a come *scriverlo*, né tantomeno a come poi si faccia *di fatto* a ottenere; inoltre, si è detto che questo polinomio esiste, ma non si è ancora dimostrato perché.

Prima di tutto, concentriamoci sul problema di *come si fa a scrivere* questo polinomio. Volendo usare una terminologia adeguata, più che di *scrivere* dovremmo parlare di *rappresentare* il polinomio. E, a tal scopo, riprendiamo la **rappresentazione monomiale** già introdotta in (3):

$$p_n(x) = c_1 x^n + c_2 x^{n-1} + \ldots + c_{n-1} x^2 + c_n x + c_{n+1}, \tag{9}$$

dove i $\{c_k\}$ sarebbero, se questo ipotetico polinomio esistesse, i suoi coefficienti, **che lo determinano completamente**.

A questo punto è possibile proporre una *prima* dimostrazione dell'esistenza del polinomio interpolante: esiste un comando MATLAB®in grado di calcolare i coefficienti di (9). Se questo comando calcola questi coefficienti, vuol dire che il polinomio deve necessariamente esistere ;-). Segue un esempio di codice che utilizza questi comandi.

```
clear
close all
clc

x = [0  1 2 1/2];              % definisco i nodi in cui sono definiti i dati da interpolare...
y = [1 -1 1   2];              % ...e i dati da interpolare

% il polinomio di interpolazione DEVE avere grado n, a partire da n+1 nodi.
c = polyfit(x,y,length(x)-1);  % c sono i coefficienti nella rappresentazione monomiale
z = linspace(min(x),max(x));   % z e` la griglia "fine" sulla quale andiamo a vedere l'interpolato↩
    ...
p = polyval(c,z);              % vado a valutare il polinomio di coefficienti c nella griglia z

plot(x,y,'ro',z,p,'b')         % disegno i nodi di interpolazione, cerchi rossi, e l'interpolante
```

Può essere di interesse investigare un po' meglio come agisce questo codice. In particolare se, dopo averlo eseguito, chiediamo nella command window di visualizzare il contenuto della variabile `c`, otteniamo

```
>> c
c =
    6.6667  -18.0000    9.3333    1.0000
>>
```

Come vediamo, il codice ha 4 nodi/dati di interpolazione, e si richiede che il polinomio in questione abbia grado pari a 4 – 1 = 3 (infatti il comando `length` restituisce la lunghezza di un vettore). Quindi, un polinomio di grado 3 ha 4 coefficienti: c_1, c_2, c_3, c_4. Il vettore `c` appena stampato a schermo contiene proprio questi coefficienti[6]. Il comando `polyval` permette, una volta noti i coefficienti `c` calcolati grazie a `polyfit`, di calcolare il corrispettivo polinomio interpolante in una griglia di punti più fitta, che in questo esempio è chiamata `z`. I coefficienti calcolati con `polyfit` sono calcolati secondo la forma monomiale (9); questo è il motivo per cui viene usata questa notazione. In effetti, il comando

```
p = polyval(c,z);
```

si potrebbe rimpiazzare con

```
c1 = c(1);
c2 = c(2);
c3 = c(3);
c4 = c(4);
p = c1.*z.^3 + c2.*z.^2 + c3.*z + c4;
```

Al fine di non appesantire eccessivamente il testo, l'appendice A riporta alcuni altri esempi di script MATLAB®atti a calcolare il polinomio interpolante per alcune funzioni[7]. Si può notare che, a volte, eseguendo script di questo genere, possa apparire il messaggio

```
Warning: Polynomial is badly conditioned. Add points with distinct X values,
reduce the degree of the polynomial, or try centering and scaling as described
in HELP POLYFIT.
> In polyfit (line 75)
  In Lez03_EsempioInterpAvanzato (line 35)
```

Ossia, MATLAB®denuncia problemi di **cattivo condizionamento**[8]. Quando impareremo come

[6] come faccia il comando polyfit a produrre questi coefficienti sarà argomento del prossimo capitolo del corso ;-)

[7] in particolare, il secondo di questi esempi sfrutta alcune sintassi più avanzate che sarebbe opportuno conoscere; viene presentato un codice molto commentato che descrive ogni passaggio.

[8] ricorda nulla? lezione precedente!

MATLAB®calcola questi coefficienti, capiremo esattamente da dove nasce questo cattivo condizionamento.

2.3 Rappresentazione di Lagrange del polinomio di interpolazione

La precedente sezione ha proposto, come argomentazione dell'esistenza del polinomio interpolante, il fatto che MATLAB®*sa calcolarlo*. Volendo però essere un po' più seri e proporre un'argomentazione più solida e rigorosa, possiamo fare riferimento a uno dei risultati più importanti di Lagrange[9]. Prima di chiamarlo in causa, però, è opportuno che un concetto sia assolutamente chiaro: **la rappresentazione monomiale e la rappresentazione di Lagrange sono due modi diversi di scrivere esattamente lo stesso polinomio**. Come già detto e dimostrato, il polinomio interpolante è unico; nonostante si possa scrivere in due modi apparentemente molto diversi, **per un insieme di dati e nodi, il polinomio interpolante è lo stesso a prescindere da tutti i modi in cui possiamo scriverlo**. Per chiarire ulteriormente il concetto, proponiamo un esempio banale: se uno scrive

$$(x-3)(x+2),$$

o scrive

$$x^2 - x - 6,$$

in verità sta scrivendo la stessa, identica cosa: la prima è una **rappresentazione fattorizzata**, la seconda è una **rappresentazione estesa**, ma il polinomio **è lo stesso**.

Al fine di spiegare la rappresentazione di Lagrange, probabilmente il metodo più facile è basarsi su un caso specifico. Si consideri un insieme di 4 nodi, che potrebbero essere per esempio quelli dello script riportato sopra:

$$x_i = \{x_1, x_2, x_3, x_4\} = \left\{0, 1, 2, \frac{1}{2}\right\}.$$

Si noti che non è necessario né che siano in ordine, né che siano equispaziati. Per ciascuno di questi nodi, è possibile definire un polinomio. In questo caso, il primo di questi polinomi è:

$$\ell_1(x) = \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)}.$$

Cerchiamo di capire come è costruito: sia al numeratore, sia al denominatore, si può vedere il prodotto di 3 fattori (3 sarebbe il numero di nodi meno 1). La variabile x appare solo al numeratore, mentre il denominatore contiene solo i nodi[10]. Il numeratore e il denominatore hanno una forma estremamente simile, eccetto che la variabile x è rimpiazzata con x_i, dove i è l'indice del polinomio. Infine, si nota chiaramente che x_1 al numeratore non appare. Avendo capito come si costruisce questo polinomio, è possibile intuire che i successivi saranno:

$$\ell_2(x) = \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)},$$

$$\ell_3(x) = \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)},$$

$$\ell_4(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)}.$$

Vale sempre la regola generale: per il polinomio ℓ_i, al numeratore ci sono tutti i prodotti $(x-x_j)$, dove j assume tutti i valori tranne i, così come il denominatore contiene tutti i prodotti (x_i-x_j), tranne $i \neq j$.

Concentriamoci ancora su $\ell_1(x)$; possiamo notare che, se valutiamo ℓ_1 in $x = x_1$, esso vale:

[9] data l'importanza del personaggio, riporto qualche pillola di storia in appendice B.

[10] ovviamente: altrimenti, se x fosse sia al numeratore sia al denominatore, non sarebbe un polinomio ma una funzione razionale!

$$\ell_1(x_1) = \frac{(x_1 - x_2)(x_1 - x_3)(x_1 - x_4)}{(x_1 - x_2)(x_1 - x_3)(x_1 - x_4)} = 1.$$

Ovviamente, questa è una proprietà generale di questi polinomi: $\ell_i(x_i) = 1$. Se invece provassimo a valutare $\ell_1(x)$ in $x = x_3$, avremmo

$$\ell_1(x_3) = \frac{(x_3 - x_2)(\mathbf{x_3} - \mathbf{x_3})(x_3 - x_4)}{(x_1 - x_2)(x_1 - x_3)(x_1 - x_4)} = 0.$$

Allo stesso modo, $\ell_i(x_j)$, per $i \neq j$, è sempre uguale a 0.

Note tutte queste proprietà, possiamo scrivere

$$p_n(x) = \sum_{j=1}^{n+1} y_j \ell_j(x).$$

Il nome $p_n(x)$ non è casuale, dal momento che questo è esattamente la rappresentazione del polinomio interpolante che stavamo cercando. Infatti, scrivendolo più esplicitamente per il caso in esame, abbiamo che:

$$p_n(x) = y_1\ell_1(x) + y_2\ell_2(x) + y_3\ell_3(x) + y_4\ell_4(x).$$

Partendo da tutto ciò che abbiamo scritto, è molto semplice vedere che questo è un polinomio interpolante. Infatti, se andiamo a verificare le condizioni di interpolazione, notiamo che

$$p_n(x_1) = y_1 \underbrace{\ell_1(x_1)}_{1} + y_2 \underbrace{\ell_2(x_1)}_{0} + y_3 \underbrace{\ell_3(x_1)}_{0} + y_4 \underbrace{\ell_4(x_1)}_{0} = y_1$$

$$p_n(x_2) = y_1 \underbrace{\ell_1(x_2)}_{0} + y_2 \underbrace{\ell_2(x_2)}_{1} + y_3 \underbrace{\ell_3(x_2)}_{0} + y_4 \underbrace{\ell_4(x_2)}_{0} = y_2$$

$$p_n(x_3) = y_1 \underbrace{\ell_1(x_3)}_{0} + y_2 \underbrace{\ell_2(x_3)}_{0} + y_3 \underbrace{\ell_3(x_3)}_{1} + y_4 \underbrace{\ell_4(x_3)}_{0} = y_3$$

$$p_n(x_4) = y_1 \underbrace{\ell_1(x_4)}_{0} + y_2 \underbrace{\ell_2(x_4)}_{0} + y_3 \underbrace{\ell_3(x_4)}_{0} + y_4 \underbrace{\ell_4(x_4)}_{1} = y_4.$$

Inoltre: è evidente che $\ell_i(x)$ è un polinomio di grado 3, quindi p_n è davvero il polinomio interpolante. In particolare, questi $\ell_i(x)$ vengono detti **polinomi fondamentali di Lagrange.**

A Esempi di implementazione MATLAB del polinomio di interpolazione

Viene ora riportato lo script MATLAB®che genera il grafico di Fig. 1.

```
clear       % svuoto la memoria
close all   % chiudo tutte le figure precedentemente aperte
clc         % pulisco la command window

n=13; % definisco il grado del polinomio interpolante
x=linspace(-5,5,n+1); % vettore di nodi di interpolazione, equidistanti
f=1./(1+x.^2); % valutazione della funzione nei nodi di interpolazione

xref=linspace(-5,5,1001); % griglia, fitta, di riferimento per i plot
fref=1./(1+xref.^2); % valutazione della funzione nella griglia fitta

c=polyfit(x,f,n); % calcolo dei coefficienti del polinomio interpolante
pol=polyval(c,xref); % valutazione del polinomio interpolante su griglia fitta

figure % apro una figura
```

```
set(gcf,'Position',[368 212 955 638]) % imposto dimensioni e posizione della figura
grid on % ordino di mettere una griglia sullo schermo
hold on % ordino alla funzione di mantenere le tracce precedenti
box on % per gradevolezza, faccio chiudere il plot da un "bordo"
plot(xref,fref,'b') % ordino di disegnare la funzione sulla griglia fitta, in colore blu
plot(x,f,'ko') % ordino di disegnare i nodi di interpolazione e marcarli con dei cerchietti (o) neri↩
    (k)
plot(x,f,'g') % ordino di disegnare la funzione, in colore verde, sulla griglia di interpolazione; ↩
    questo coincide
            % con disegnare l'interpolazione lineare dei nodi (  simile a interp1)
plot(xref,pol,'r') % disegno il polinomio interpolante valutato sulla griglia fitta, in colore rosso
axis([-5,5,-1.5,1]) % imposto gli assi che verranno visualizzati: x va da -5 a 5, y va da -1.5 a 1
xlabel('x') % imposto l'etichetta dell'asse delle ascisse
ylabel('f(x)')  % imposto l'etichetta dell'asse delle ordinate
```

Inoltre, viene riportato un secondo esempio, che contiene esempi di utilizzo di sintassi più avanzate rispetto a quelle viste fino a ora.

```
clear
close all
clc

f = @(x) x.*sin(x); % questa sintassi permette di definire un function
                    % handle; si tratta di un modo di lavorare di MATLAB
                    % che permette di trattare "f" non come una variabile,
                    % ma come una funzione che si puo` valutare con una
                    % sintassi tipo f(x), f(z) o quello che e`, dove x e z
                    % sono dei vettori di punti

z = linspace(0,2*pi); % definisco la griglia "fine" per vedere gli interpolanti

% segue un esempio un po' particolare di utilizzo del ciclo for. Il
% concetto del ciclo for e` lo stesso del C, pero` qui la variabile sulla
% quale si cicla va definita come un vettore, e MATLAB eseguira` tanti cicli
% quanti sono gli elementi nel vettore n. Nell'esempio in questione, n
% contiene due elementi, quindi avremo due cicli. Per ogni ciclo, n assume
% il valore corrispettivo.
% In questo caso, diversi valori di n corrispondono, come si puo` vedere
% all'interno del codice, a diversi ordini del polinomio interpolante
for n = [5 11]
    % il comando linspace(a,b,N) permette di definire un vettore di
    % elementi equispaziati a partire da a, fino a b, composto da N
    % elementi. Si tratta di un comando un po' diverso dalla sintassi a:b,
    % dal momento che anche in quel caso erano equispaziati, ma in quel
    % caso non si fissava il numero di elementi, quanto piuttosto il passo.
    x = linspace(0,2*pi,n+1); % definisco un vettore di n+1 nodi di interpolazione
                              % poiche' voglio un polinomio di grado n

    y = f(x); % sfrutto il function handle per poter scrivere velocemente che
              % i dati da interpolare, y, sono uguali a f valutata nei nodi
              % x

    c = polyfit(x,y,n); % calcolo coefficienti polinomio interpolante grado n...
    p = polyval(c,z);   % ... e valuto il polinomio sulla griglia fine z

    plot(x,y,'ro',z,f(z),'r',z,p,'b') % i nodi di interpolazione sono disegnati con
                                      % cerchietti rossi, la funzione di
                                      % partenza nella griglia fine e`
                                      % disegnata con una curva rossa, e il
                                      % polinomio interpolante con una
                                      % curva blu

    pause % metto una pausa, per permettere di visualizzare per ogni polinomio la figura
end
```

B Pillole di storia: Joseph-Louis Lagrange

Nonostante il nome non sembrerebbe suggerirlo, Joseph-Louis Lagrange[11] era italiano e, in particolare, torinese; questo è il motivo per il quale gli è stato dedicato il Dipartimento di Scienze Matematiche (DISMA) del Politecnico di Torino, nonché una delle principali vie del centro di Torino. Giuseppe Lodovico Lagrangia nasceva a Torino il 25 gennaio 1736. Volendo usare un termine che va di moda, è stato un *cervello in fuga*, in quanto, dopo Torino, andò prima a Berlino verso il 1760, poi a Parigi intorno al 1787, dove si stabilì più o meno definitivamente. Esistono diverse versioni della sua firma, tutte attribuibili a lui, perché in quegli anni c'era la rivoluzione

[11] altri dettagli su https://it.wikipedia.org/wiki/Joseph-Louis_Lagrange

francese e utilizzare nomi che potevano evocare origini nobili poteva portarti dal boia. Il punto di arrivo fu il cognome *Lagrange* con cui ora lo conosciamo tutti.

Per capire un po' meglio il personaggio, vorrei citare le parole[12] di un suo illustre studente, Jean Baptiste Joseph Fourier, padre della teoria del calore.

> Lagrange, il principale accademico d'Europa, sembra avere tra i 50 e i 60 anni di età, anche se in realtà è più giovane (*n.d.r.: all'epoca in realtà aveva proprio 59 anni...*); ha un forte accento italiano e pronuncia una *s* come se fosse una *z*; si veste con molta discrezione, in nero o marrone; parla informalmente e con qualche difficoltà, con l'esitante semplicità di un bambino. Tutti sanno che è un uomo straordinario, ma bisogna averlo visto per riconoscerlo come tale. Parla solo in discussione, e parte di ciò che dice eccita derisioni. L'altro giorno ha detto:
> "Ci sono molte cose importanti da dire su questo argomento, ma io non le dirò."
> Gli studenti sono incapaci di apprezzare il suo genio, ma i professori lo riconoscono chiaramente.

L'importanza di Lagrange nel progresso della Matematica è conclamata grazie ai suoi contributi in diversi ambiti, che vale la pena di riassumere.

- La fondazione del Calcolo delle Variazioni. I metodi variazionali permettono di attribuire rigore matematico ad alcuni principi fondamentali della Fisica, quali il principio di minima azione, e permettono di fornire una delle formulazioni del metodo degli elementi finiti (FEM) per la soluzione di equazioni alle derivate parziali.
- La formulazione della Meccanica Analitica, anche detta Meccanica Razionale.
- L'ideazione dei moltiplicatori di Lagrange, che permettono di risolvere problemi di ottimizzazione vincolata, ossia, cercare il minimo di una qualche funzione a molte variabili, dati alcune condizioni che devono essere soddisfatte.
- Ovviamente, la rappresentazione di Lagrange del polinomio interpolante, anch'essa impiegata nella forma basilare del metodo degli elementi finiti.

[12] Fonte: http://www-groups.dcs.st-and.ac.uk/history/Extras/Fourier_teachers.html