Grafici, best-fit (analitico e numerico) e Python

francesco.fuso@unipi.it; http://www.df.unipi.it/∼fuso/dida

(Dated: version 3 - FF, 5 ottobre 2016)

Questa nota intende richiamare alcuni metodi che sono normalmente coinvolti nell'analisi dei dati, cioè la preparazione di grafici e la realizzazione di best-fit. L'accento è posto soprattutto sugli aspetti pratici e a questo scopo si fa riferimento all'impiego del software Python (saccheggiando ampiamente gli appunti di Luca Baldini e Carmelo Sgrò, che conoscete, lots of kudos to them!). Lo scopo della nota è anche quello di chiarire l'ambito concettuale delle nostre analisi, cioè stabilire l'utilità del best-fit e gli scopi principali che esso avrà per noi.

I. GRAFICI

La rappresentazione di coppie di dati (sperimentali o no che siano) in un grafico è una tecnica diffusa e nota al punto che è certamente fuori luogo discuterne l'importanza. Spesso il grafico è il risultato principale, o addirittura unico, di un'attività scientifica. Esso deve pertanto contenere in modo ordinato e chiaro quante più informazioni possibile. Dunque le grandezze riportate sugli assi devono essere correttamente indicate assieme alla loro unità di misura [1], i valori numerici sugli assi devono essere ben leggibili [2], il range dei valori graficati deve essere opportuno (gli spazi vuoti su un grafico non servono a molto), e, se i dati sono di origine sperimentale, devono *sempre* comparire le barre di errore, a prescindere dal fatto che esse vengano usate, o meno, in sede di analisi.

Un grafico decente deve poter mostrare al primo colpo se i dati seguono un qualche andamento. A tale scopo può essere utile la rappresentazione logaritmica o semilogaritmica. Per esempio, un andamento di tipo esponenziale decrescente (il decadimento temporale della carica sulle armature di un condensatore) è rappresentato in carta semi-logaritmica come un andamento lineare, con una pendenza inversamente proporzionale alla costante tempo di decadimento; un andamento secondo una legge di potenza è rappresentato in scala logaritmica come un andamento lineare, con una pendenza proporzionale all'esponente. Questi sono esempi di *rappresentazione linearizzata*, in cui i dati originari non sono manipolati, ma è la particolare scala non-lineare della rappresentazione che li fa apparire disposti secondo un andamento lineare [3]. Inoltre, a prescindere dall'esigenza di individuare al primo colpo un certo andamento, l'uso della rappresentazione logaritmica è opportuno ogni volta che si devono rappresentare dati che spaziano su un ampio range.

Per la sua importanza, la realizzazione di un grafico dovrebbe essere sempre la *prima* operazione da compiere in un processo di analisi dei dati. Dunque, anche se l'analisi comprende altre fasi, per esempio un best-fit come nell'esempio che tratteremo in seguito, è *fondamentale* che la visualizzazione del grafico *preceda ogni altra operazione*. Quando si usa un software come Python, in cui è necessario scrivere uno script che poi viene interpretato per eseguire le operazioni richieste, è necessario che la parte relativa alla realizzazione del grafico *preceda* quella relativa al best-fit. L'utilità di questo approccio è almeno duplice: da un lato esso consente di verificare immediatamente la presenza di eventuali errori nella "acquisizione" dei dati, manuale o automatizzata che sia; dall'altro, la visualizzazione del grafico permette di escludere errori nella parte di script che arriva alla preparazione del grafico stesso. Questa parte comprende tipicamente la lettura di un file di testo, per cui è possibile si verifichino errori di varia natura, per esempio nell'indirizzamento della directory.

A. Grafico in Python

Richiamiamo qui le principali operazioni (concettuali e pratiche) da compiere per realizzare un grafico in Python. Si fa riferimento ai pacchetti, o librerie, di uso comune per questi scopi, cioè `pylab`, `matplotlib` ed eventualmente `numpy`, da richiamare all'inizio dello script. Immaginiamo poi di avere un set di dati, corredato da incertezze, registrato in un file di testo presente nel computer. Come ben sapete, questa non è l'unica possibilità pratica per "comunicare" i dati al computer. Qualcuno, infatti, potrebbe preferire creare degli arrays di dati direttamente all'interno dello script: per motivi di carattere pratico (possibilità di riusare lo stesso script, verifica immediata e visuale della correttezza del set di dati) e di "eleganza", è fortemente *sconsigliato* l'impiego di questa tecnica rispetto alla realizzazione e lettura del file di dati. In questo esempio supponiamo che i dati siano stati inseriti in quattro colonne che riportano rispettivamente i valori x, Δx, y e Δy, con ovvio significato dei simboli.

La prima istruzione necessaria è quella che ordina al software di aprire e leggere il file, trasferendone il contenuto in quattro arrays, che chiamiamo `x, Dx, y, Dy`, con ovvio significato. Questo si può ottenere con il comando `x,Dx,y,Dy=pylab.loadtxt('nomefile.txt', unpack=True)`, dove il nome del file può contenere anche l'eventuale indirizzamento alla directory in cui esso si trova. A questo punto la realizzazione del grafico con le barre di errore può essere eseguita usando il comando (notate l'ordine degli argomenti) `pylab.errorbar(x,y,dy,dx,linestyle = '', color = 'black', marker = 'o')`, che contiene anche delle istruzioni molto ovvie che riguardano il formato

(senza linea che congiunge i punti), il colore (nero) e il tipo di marker (un pallino vuoto). Per visualizzare il grafico sullo schermo, che al momento è tutto quello che serve, è sufficiente inserire il comando `pylab.show()`. Se il grafico deve essere salvato in opportuno formato, per esempio `.pdf`, per ulteriori usi, o per la stampa, conviene usare i comandi a icona della finestra di rappresentazione grafica di Pylab. Prima dell'istruzione di visualizzazione vanno anteposte alcune istruzioni di stile, che comprendono anche la denominazione e l'unità di misura delle grandezze riportate sugli assi. Per fissare le idee, immaginiamo che i dati rappresentino delle intensità di corrente (I, misurate in milliAmpere, [mA]) in funzione di differenze di potenziale (ΔV, misurate in Volt, [V]).

Lo script di Python che produce il grafico di Fig. 1 è riportato qui nel seguito:

```
import pylab
import matplotlib
# data load (nomefile includes addressing to the correct folder)
x,Dx,y,Dy=pylab.loadtxt('nomefile.txt',unpack=True)
# scatter plot with error bars
pylab.errorbar(x,y,Dy,Dx,linestyle = '', color = 'black', marker = 'o')
# bellurie
pylab.rc('font',size=16)
# the $ symbol allows using LaTeX style characters
pylab.xlabel('$\Delta V$  [V]')
pylab.ylabel('$I$  [mA]')
pylab.title('Data plot')
pylab.minorticks_on()
# show the plot
pylab.show()
```

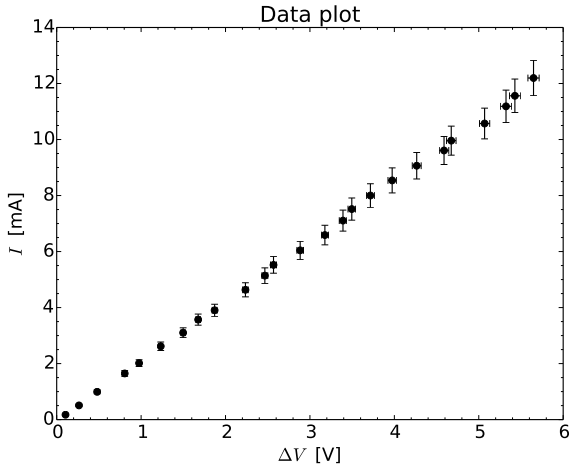

Figura 1. Grafico prodotto dallo script di Python discusso nel testo.

II. BEST-FIT

Eseguire un best-fit è una delle operazioni più frequenti nella pratica sperimentale. Dal punto di vista concettuale, il best-fit implica di analizzare quantitativamente le discrepanze tra osservazioni sperimentali (ottenute da esperimenti veri e propri o da simulazioni numeriche) e previsioni di un modello formulato sulla base di considerazioni fisiche. Il modello conduce generalmente a una funzione *analitica* $f(x)$, qui supposta dipendente da un'unica "variabile indipendente" (la x), munita di opportuni parametri. Nel produrre il modello si è confidenti che la funzione possa descrivere le osservazioni sperimentali. Dunque il best-fit analizza le discrepanze tra i valori y_i e le "previsioni" $f(x_i)$, con x_i e y_i coppie di dati sperimentali, e agisce sui parametri della funzione $f(x)$ in modo da minimizzare per quanto possibile tali discrepanze.

Sulla base di queste premesse, è ovvio che fare un best-fit *non* significa (solo, e in realtà affatto) far passare una curva analitica su un set di dati sperimentali, operazione in genere poco significativa. In particolare, l'obiettivo del best-fit non è (mai) quello di determinare una funzione $f(x)$ "qualsiasi" che sia in grado di descrivere al meglio le osservazioni. Infatti la costruzione della $f(x)$ deve essere fatta sulla base di considerazioni fisiche ben definite, con un grado di arbitrarietà idealmente molto limitato, o nullo. Sicuramente fare un best-fit conduce ad importanti risultati, tra cui:

1. determinare quantitativamente grandezze incognite del sistema sotto analisi che compaiono come parametri nella funzione di fit e stimare l'incertezza a loro associata, permettendo, in sostanza, di eseguirne una "misura indiretta";

2. quando possibile, confrontare diversi modelli di interpretazione dei dati, ovvero diverse funzioni di fit, per stabilire quale consenta la migliore descrizione dell'osservazione sperimentale;

3. nei (rari) casi in cui è disponibile una sufficiente "statistica" per le misure, determinare la significatività dell'interpretazione.

Il "risultato" di un best-fit, allora, non può essere il solo grafico dei dati con sovrapposta la linea continua della funzione di fit. Questo grafico è necessario per valutare immediatamente la correttezza del modello impiegato, ma esso non esaurisce la quantità di informazioni che possono essere tratte dal best-fit. Per gli scopi didattici che ci prefiggiamo, è *sempre* necessario aggiungere:

1. l'espressione analitica della funzione modello;

2. il valore dei parametri della funzione ottenuti dal best-fit, che diventano grandezze fisiche, correttamente dimensionate;

3. l'incertezza su tali parametri, ovvero sulla misura indiretta delle grandezze che compaiono come parametri nella funzione di best-fit, assieme alla *modalità* con cui questa incertezza è stata determinata, secondo quanto illustreremo in seguito;

4. il valore del χ^2 risultante, assieme ad eventuali commenti sulla scelta dell'incertezza dei dati sperimentali;

5. nel caso di fit a più di un parametro, l'esplicita indicazione della *covarianza normalizzata*, qualche volta detta anche correlazione tra i parametri, secondo quanto discuteremo nel seguito di questa nota.

In termini generali esistono diversi metodi per eseguire un best-fit. Qui useremo il cosiddetto metodo del *minimo* χ^2, che è un'estensione del fit a *minimi quadrati* adatta per trattare situazioni sufficientemente generali, in cui i dati hanno un'incertezza non "costante". Questo è quanto si verifica molto spesso nelle misure di segnali elettrici effettuate con i normali strumenti di laboratorio [4].

A. Richiami sul fit del minimo χ^2

Eseguire un best-fit dei *minimi quadrati* secondo la funzione $f(x)$ richiede di minimizzare la somma dei residui (quadrati), cioè di minimizzare la funzione

$$\Sigma_i (y_i - f(x_i))^2 \ , \tag{1}$$

dove y_i e x_i sono le coppie di dati sperimentali e la somma è estesa al numero N di coppie di dati disponibili. Come ben sapete, questo metodo ha lo svantaggio di non considerare l'incertezza dei dati sperimentali Δy_i, che però può essere inserita come peso della somma nel seguente semplicissimo modo:

$$\sum_i \frac{(y_i - f(x_i))^2}{(\Delta y_i)^2} \ ; \tag{2}$$

questa definizione permette di attribuire la maggiore importanza a quei dati sperimentali che hanno l'incertezza minore, e viceversa minore importanza a quelli più incerti, procedura sicuramente sensata.

Alla grandezza considerata in Eq. 2 si dà spesso (quasi sempre *impropriamente*) il nome di χ^2. Questo nome nasce dall'*analogia* con una variabile aleatoria χ^2 costruita come somma dei quadrati di un'altra variabile aleatoria *standard* ξ_i: $\chi^2 = \Sigma_i \xi_i^2$. La distribuzione di probabilità del χ^2 è nota (tabelle e calcoli numerici) e si sa che essa, per un numero N sufficientemente grande, tende ad assumere il valore medio $\mu_{\chi^2} \simeq N$ e la deviazione standard $\sigma_{\chi^2} \simeq \sqrt{2N}$.

Affinché ciò sia verificato, cioè affinché la grandezza creata abbia il significato di un χ^2, occorre che la variabile aleatoria ξ_i sia distribuita secondo una Gaussiana (significato del termine standard) a media nulla e *varianza unitaria (normalizzata)*. Dunque nel considerare la grandezza definita in Eq. 2 come un χ^2 stiamo facendo un'*importante e delicata assunzione*. Infatti riteniamo di poter sostituire la varianza σ_i^2 dell'(ipotetica) distribuzione delle ξ_i con $(\Delta y_i)^2$, un'operazione che, come vedremo un po' meglio nel seguito, non è sempre e necessariamente giustificata (anzi, nelle nostre condizioni non lo è quasi mai).

Per ora notiamo che *solo se* le condizioni che abbiamo posto sono ritenute ragionevoli il calcolo del χ^2 può essere usato per stabilire dei criteri per la valutazione *quantitativa* della significatività del best-fit. Il più noto di questi criteri è quello di Pearson, detto anche semplicemente del χ^2, che avete conosciuto lo scorso anno. Esso è direttamente collegato alla stima della probabilità che si possa avere un χ^2 più alto, o più basso, di quello ottenuto, attraverso l'uso di tabelle che riportano l'integrale dell'area sottesa alla curva di distribuzione (normalizzata), cioè la probabilità (normalizzata). Nei rari casi in cui può essere sensatamente impiegato, tale metodo risponde allora a uno dei "requisiti" che avevamo posto prima, quello di dare una valutazione quantitativa della significatività del best-fit.

B. Come impostare il best-fit

Definire operativamente il χ^2 come in Eq. 2, pur essendo una prassi comunissima, pone diversi problemi, alcuni dei quali sono elencati qui di seguito.

1. Il valore del χ^2 risultante dal best-fit, cioè quello che si calcola al termine della procedura di minimizzazione, ovvero quello che viene eventualmente usato per valutare la significatività, dipende in modo inversamente proporzionale dal quadrato delle incertezze Δy_i. Una non corretta valutazione delle incertezze comporta una valutazione non corretta del χ^2. Per meglio dire, la grandezza espressa in Eq. 2 può essere interpretata come un χ^2 *solo se* le condizioni sopra elencate sono verificate, altrimenti

il suo significato è *solo* quello di una somma *pesata* degli scarti quadratici tra aspettativa di modello e dati sperimentali.

2. In termini generali, il carattere aleatorio della variabile χ^2 costruita con i dati sperimentali dipende dal fatto che le y_i sono risultato di una misura affetta da incertezza prevalentemente *stocastica*, e che anche la $f(x_i)$ ha un aspetto stocastico dovuto alla possibilità di aggiustare, idealmente in modo casuale, i parametri della funzione $f(x)$ per migliorare l'accordo con i dati. Tutto questo è vero solo se il campione è statisticamente significativo. Tra l'altro, ciò implica che il numero di dati a disposizione sia sufficientemente grande. Nella pratica, il metodo si ritiene valido se il numero di gradi di libertà (ndof = N - numero parametri della funzione) è superiore ad almeno 5 e se le incertezze sperimentali hanno prevalente natura stocastica.

3. Aspetto che combina i due punti precedentemente elencati: molto spesso le misure eseguite con gli strumenti tipici per le misure elettriche (tester, oscilloscopi, digitalizzatori) sono corredate da un errore dominato dall'incertezza strumentale *di calibrazione*, che tipicamente copre l'errore stocastico. In genere questo conduce a sovrastimare l'incertezza, con l'ovvia conseguenza di ottenere valori di χ^2 molto piccoli e prevenire *completamente* la possibilità di trarre conclusioni sulla significatività e il livello di confidenza del best-fit. Naturalmente, però, è sempre possibile confrontare i χ^2 ottenuti con due modelli diversi e affermare che il modello "migliore" è quello che conduce al χ^2 più basso.

La sostituzione $\sigma_i^2 \leftrightarrow (\Delta y_i)^2$ fatta in Eq. 2 implica evidentemente di poter trascurare l'incertezza Δx_i sulla misura della grandezza x_i. È chiaro che si possono verificare delle situazioni in cui l'incertezza sulla misura x_i sia affetta da un'incertezza relativa paragonabile a quella su y_i, essendo essa stessa il risultato di una misura.

Il metodo del minimo χ^2 non contiene un'estensione immediata e rigorosa per queste situazioni. Esistono alcuni metodi numerici in cui la minimizzazione della grandezza di Eq. 2 è cercata muovendosi su opportune "traiettorie" dello spazio x, y, permettendo di tenere conto di tutte e due le incertezze. Questi metodi, impementabili anche con Python, non hanno grande diffusione in ambito scientifico, poiché risultano normalmente affidabili solo per funzioni modello lineari, che si trovano piuttosto raramente.

Un modo (non pulito e non sicuro) per tenere conto delle incertezze Δx_i, che a voi è già noto, può essere quello di basarsi sulla propagazione degli errori. In pratica si valuta il contributo sull'incertezza in y_i, $\Delta y_i|_{\Delta x_i}$, detto talvolta *errore equivalente*, dovuto all'incertezza su x_i, Δx_i. Secondo le regole della propagazione degli errori, tale contributo può essere espresso come $\Delta x_i |\partial f(x)/\partial x|_{x=x_i}$ per cui si può individuare una sorta di *errore efficace* attraverso somma in quadratura:

$$\sigma_i^2 = (\Delta y_i)^2 + \left(\Delta x_i |\frac{\partial f(x)}{\partial x}|_{x=xi}\right)^2 . \qquad (3)$$

Questa espressione ha un'evidente limitazione, poiché richiede la conoscenza a priori della funzione $f(x)$ inclusi i suoi parametri, che invece è in genere proprio quanto si vuole determinare con la procedura di best-fit. Dunque la sua implementazione deve essere valutata attentamente caso per caso. Normalmente, è bene prima di tutto eseguire un best-fit trascurando le incertezze Δx_i; i parametri ottenuti da questo best-fit possono poi essere usati per la valutazione di $\Delta y_i|_{\Delta x_i}$ consentendo la realizzazione di un nuovo best-fit (con una nuova stima dell'incertezza, come da Eq. 3). Qualora i parametri ottenuti dal nuovo best-fit si discostino in modo significativo dalla prima valutazione (in genere non succede mai), è possibile iterare la procedura.

Fatte salve le considerazioni presentate finora, il problema pratico nella realizzazione di un best-fit è quello di minimizzare la funzione di Eq. 2. Esistono in generale due modi, uno basato sulla minimizzazione *analitica* e l'altro sulla minimizzazione *numerica*. Come rule of thumb generale, si può sicuramente affermare che la procedura analitica è più affidabile di quella numerica, dato che, per esempio, è in genere meno affetta da problemi di accuratezza matematica e di convergenza e non richiede alcuna conoscenza approssimata e preliminare (a priori) dei parametri della funzione. Tuttavia la minimizzazione analitica, a meno di non voler impazzire con paginate di conti, ha senso solo se eseguita per funzioni $f(x)$ *estremamente* semplici. Per noi, essa ha senso solo se applicata ad andamenti *lineari*.

III. BEST-FIT LINEARE ANALITICO

La rilevanza del best-fit lineare analitico è duplice: da un lato, alcuni (pochi) modelli di semplici esperimenti conducono a prevedere andamenti lineari, dall'altro, usando tecniche di *linearizzazione*, è spesso possibile manipolare matematicamente i dati per giungere a un modello lineare. Per fare qualche esempio di linearizzazione, consideriamo le seguenti situazioni:

1. un decadimento temporale esponenziale, descritto dalla funzione $y_i = A\exp(-x_i/\tau)$, è linearizzato usando il logaritmo naturale dei dati y_i. Infatti $y_i' = \ln(y_i) = \ln(A) - x_i/\tau$.

2. Un andamento di potenza del tipo $y_i = Ax_i^n$ è linearizzato usando il logaritmo naturale delle coppie di dati x_i, y_i. Infatti $y_i' = \ln(y_i) = \ln(A) + n\ln(x_i) = \ln(A) + nx_i'$.

3. Un andamento inversamente lineare, del tipo $y_i = A/x_i$, può essere linearizzato considerando il dato $x_i' = 1/x_i$. Infatti $y_i = Ax_i'$.

Nell'esempio di cui ci occupiamo in questa nota, il modello prevede già un andamento di tipo lineare, che deriva dalla legge di Ohm. Infatti la funzione che riteniamo possa descrivere i dati rappresentati in Fig. 1 è $I_i = \Delta V_i/R + I_0$, che ha appunto la forma di una "retta che non passa per l'origine", $f(x) = a + bx$.

A. Formule analitiche per la retta che non passa per l'origine

La minimizzazione dell'Eq. 2 rispetto ai parametri a, b nel caso di $f(x) = a + bx$ può essere eseguita usando il metodo dei moltiplicatori di Lagrange. Il calcolo, che è abbastanza laborioso (in Appendice se ne riporta una versione semplificata), conduce alle seguenti formule per i parametri a e b con le rispettive incertezze Δa e Δb [5]:

$$w_i = \frac{1}{\sigma_i^2} \tag{4}$$

$$\Delta' = \Sigma_i w_i \Sigma_i w_i x_i^2 - (\Sigma_i w_i x_i)^2 \tag{5}$$

$$a = \frac{\Sigma_i w_i x_i^2 \Sigma_i w_i y_i - \Sigma_i w_i x_i \Sigma_i w_i x_i y_i}{\Delta'} \tag{6}$$

$$\Delta a = \sqrt{\frac{\Sigma_i w_i x_i^2}{\Delta'}} \tag{7}$$

$$b = \frac{\Sigma_i w_i \Sigma_i w_i x_i y_i - \Sigma_i w_i x_i \Sigma_i w_i y_i}{\Delta'} \tag{8}$$

$$\Delta b = \sqrt{\frac{\Sigma_i w_i}{\Delta'}}\ , \tag{9}$$

dove le somme si intendono estese su tutte le N coppie di dati disponibili.

Le Eqq. 4-9 hanno un aspetto sicuramente poco simpatico, ma, con un minimo di attenzione, si può notare che in esse compaiono "solo" cinque espressioni indipendenti, tutte generate da somme di elementi (dati sperimentali x_i, y_i e *pesi statistici* $w_i = 1/\sigma_i^2$). Uno script di Python può assai facilmente calcolare tutto ciò. Lo script, che è disponibile in rete sotto il nome di `anal_lin_two_parms.py`, può per esempio avere la seguente forma:

```
import pylab
import numpy
# data load
x,Dx,y,Dy=pylab.loadtxt(''nomefile.txt',unpack=True)
# scatter plot with error bars
pylab.errorbar(x,y,Dy,Dx,linestyle = '', color = 'black', marker = 'o')
# bellurie
pylab.rc('font',size=16)
pylab.xlabel('$\Delta V$  [V]')
pylab.ylabel('$I$  [mA]')
pylab.title('Data plot w analytical fit')
pylab.minorticks_on()

# set the error and the statistical weight
sigma=Dy
w=1/sigma**2
# determine the coefficients
c1=(w*x**2).sum(); c2=(w*y).sum();c3=(w*x).sum()
c4=(w*x*y).sum(); c5=(w).sum()
Dprime=c5*c1-c3**2
a=(c1*c2-c3*c4)/Dprime
b=(c5*c4-c3*c2)/Dprime
Da=numpy.sqrt(c1/Dprime)
Db=numpy.sqrt(c5/Dprime)

# define the linear function
# note how parameters are entered
# note the syntax
def ff(x, aa, bb):
    return aa+bb*x

# calculate the chisquare for the best-fit funtion
chi2 = ((w*(y-ff(x,a,b))**2)).sum()

# determine the ndof
```

```
ndof=len(x)-2

# print results on the console
print(a,Da, b,Db)
print (chi2, ndof)

# prepare a dummy xx array (with 100 linearly spaced points)
xx=numpy.linspace(min(x),max(x),100)
# plot the fitting curve
pylab.plot(xx,ff(xx,a,b), color='red')
# show the plot
pylab.show()
```

In esso si è fatto uso della particolare sintassi di Python che permette di eseguire operazioni matematiche sugli arrays mediante "concatenazione" (le istruzioni che terminano con `.sum()`). Inoltre, per comodità e per facilitare le ulteriori varianti che vedremo in seguito, si è definita la funzione $f(x)$, chiamata qui `ff`, mediante opportuni comandi. Infine, poiché siamo abituati a "vedere" le funzioni come delle linee continue, lo script contiene la definizione di un array ausiliario, chiamato `xx`, composto di 100 punti equispaziati nell'intervallo tra il minimo e il massimo valore dei dati x_i. La funzione di fit è graficata usando come variabile indipendente questo array al solo scopo di evitare antiestetiche spezzate. Questo è in realtà rilevante solo nel caso di funzioni non-lineari, ma viene qui fatto anche nel caso lineare per esigenze didattiche. Nei casi non-lineari è talvolta anche necessario usare un numero di punti maggiore di 100.

Il grafico che si ottiene è riportato in Fig. 2: esso mostra un ottimo accordo "visivo" tra dati e best-fit, segno che il modello è qualitativamente adeguato per la descrizione dei dati. Per scrivere il risultato del best-fit occorre tornare indietro al significato fisico della funzione utilizzata. Si vede che il parametro a ha le dimensioni di una intensità di corrente, e viene chiamato I_0. Invece il parametro b ha le dimensioni dell'inverso di una resistenza elettrica, $b = 1/R$. Poiché fisicamente il valore interessante è proprio R, esprimere il risultato del fit richiede una semplice operazione matematica di inversione e di impiegare le altrettanto semplici regole di propagazione dell'errore per determinare l'incertezza su R. In definitiva si ha (notate la forma e, soprattutto, la corretta scelta di unità di misura e *numero di cifre significative*):

$$I_0 = (-48 \pm 24)\ \text{mA} \tag{10}$$

$$R = (470 \pm 6)\ \text{ohm} \tag{11}$$

$$\chi^2/\text{ndof} = 1.4/23\ . \tag{12}$$

Per rendere completo il set di risultati del best-fit (a più di un parametro), manca la covarianza normalizzata, che qui non determiniamo perché il suo calcolo in forma analitica è piuttosto noioso (la considereremo in seguito, quando useremo l'approccio numerico). Osservate inoltre che il χ^2 ottenuto corrisponde a un valore di χ^2 *ridotto* $\chi^2_{rid} = \chi^2/\text{ndof} \sim 0.06$, molto minore del valore 1 ± 0.6

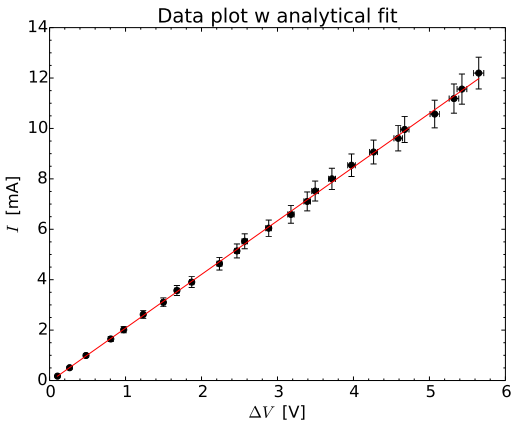

Figura 2. Grafico dei dati mostrati in Fig. 1 con sovrapposto il best-fit ottenuto per via analitica (linea continua rossa).

che ci aspetteremmo se il χ^2 fosse un "vero" χ^2 (e non la somma pesata degli scarti quadratici) e se il livello di confidenza fosse $\approx 95\%$ [6]. Questo non deve spaventare se si considera che le incertezze sui dati sperimentali che abbiamo usato come peso statistico non hanno, necessariamente, il valore di deviazioni standard dei dati misurati (non si sa come esse siano state determinate). In queste condizioni, come già affermato, *non ha alcun senso* eseguire il test del χ^2 per determinare il livello di confidenza, che quindi *non va eseguito*.

B. Retta che passa per l'origine e "media pesata"

Le formule di Eqq. 4-9 possono essere considerate come un'estensione del caso, matematicamente più semplice, di una retta che passa per l'origine (cioè una funzione del tipo $f(x) = bx$, che esprime un andamento direttamente proporzionale tra y e x). La minimizzazione analitica del χ^2 corrispondente può essere eseguita in maniera molto semplice, grazie alla presenza di un unico parametro. In alternativa, si può lavorare sulle Eqq. 4 "imponendo" $a =$

0. Alla fine si ottiene

$$w_i = \frac{1}{\sigma_i^2} \tag{13}$$

$$\Delta' = \Sigma_i w_i \Sigma_i w_i x_i^2 - (\Sigma_i w_i x_i)^2 \tag{14}$$

$$b = \frac{\Sigma_i w_i x_i y_i}{\Sigma_i w_i x_i^2} \tag{15}$$

$$\Delta b = \sqrt{\frac{\Sigma_i w_i}{\Delta'}} \ , \tag{16}$$

cioè un set di equazioni un po' più maneggevole che non per il caso precedente.

Se, invece, si impone $b = 0$, allora il calcolo si riconduce a quello della *media pesata*. Il risultato, come noto, è

$$w_i = \frac{1}{\sigma_i^2} \tag{17}$$

$$a = \frac{\Sigma_i w_i y_i}{\Sigma_i w_i} \tag{18}$$

$$\Delta a = \frac{\sigma}{\sqrt{N}} \ , \tag{19}$$

con σ la deviazione standard di una delle misure che compongono la media pesata.

IV. BEST-FIT NUMERICO

L'approccio numerico è molto più "potente" di quello analitico, dato che può essere usato per funzioni virtualmente di ogni tipo, incluse ovviamente quelle *non-lineari* (da qui il nome di fit non-lineare). In questo approccio ci si deve fidare di procedure di calcolo eseguite da un computer (il cosiddetto "algoritmo", in gergo). Nella quasi totalità dei software di trattamento dati, l'algoritmo è il Levenberg-Marquardt Algorithm (LMA), una procedura numerica per la ricerca di minimi locali generalmente efficiente e affidabile, che viene applicata alla minimizzazione del χ^2.

In Python la routine di minimizzazione è per esempio contenuta nel pacchetto `scipy.optimize`, da cui deve essere "estratta" e importata per renderla utilizzabile. I comandi relativi, come mostrato nel seguito, sono piuttosto semplici e di comprensione immediata. Prima di procedere nella descrizione della tecnica di implementazione, occorre sottolineare un aspetto generale, critico, e potenzialmente laborioso, nell'uso dell'approccio numerico.

A prescindere dal tipo di funzione, è *sempre necessario fornire dei valori iniziali* alla routine per tutti i parametri contenuti nella funzione, da cui essa possa partire per cercare il minimo. Soprattutto nel caso di funzioni "complicate" (basta che ci siano funzioni trigonometriche, o periodiche), oppure quando è a priori possibile che ci sia più di un minimo locale nel χ^2, è opportuno che le condizioni iniziali siano tali da fornire un "ragionevole" accordo (iniziale) con i dati da fittare. Se questo non si verifica, è possibile che l'algoritmo di minimizzazione non converga, oppure che converga a valori che, sulla base di considerazioni fisiche, sono palesemente sbagliati.

La scelta dei valori iniziali può essere fatta sulla base di diverse considerazioni. Per esempio, spesso la costruzione del modello permette di determinare a priori dei valori approssimati dei parametri della funzione di best-fit. Altre volte il modello stesso è così semplice (il caso lineare discusso in questa nota rappresenta bene questa condizione) che è possibile determinare valori ragionevoli dei parametri semplicemente ispezionando i dati. In generale, comunque, è sempre possibile "tentare" dei valori iniziali e vedere immediatamente dal grafico quant'è la distanza tra la funzione modello calcolata con tali valori iniziali e i dati. Infatti è molto facile su Python creare una funzione e graficarla, usando le istruzioni che già abbiamo incontrato. Dunque, iterando i tentativi e guardando l'effetto direttamente sul grafico, è in genere facile arrivare a scelte ragionevoli per i parametri iniziali.

Di conseguenza, è *estremamente opportuno* concepire l'approccio al best-fit come una successione di passi: i primi, che abbiamo già discusso, riguardano l'importazione dei dati e il loro grafico. A questi seguono quelli in cui si definisce la funzione modello, si impostano i parametri iniziali e si esegue il grafico dei dati sperimentali con sovrapposto quello della funzione *calcolata per i parametri iniziali*. Se organizzate il vostro lavoro, e anche lo script, secondo questa logica, vi troverete assai avvantaggiati quando dovrete eseguire dei best-fit un po' più complicati rispetto agli andamenti lineari trattati in questo esempio, riuscendo a terminare con la debita efficienza l'analisi dei dati.

Qui ci limitiamo ad applicare il best-fit numerico al set di dati già considerato nelle Figg. 1 e 2. Si sa che il modello che è atteso descrivere tali dati è lineare, quindi una semplice ispezione dei dati stessi conduce a determinare in maniera pressoché immediata i valori iniziali dei parametri a, b (nel caso specifico, si può prendere per esempio $a_{in} = 0$ e $b_{in} = 2$).

L'istruzione di Python che lancia la routine di minimizzazione è `curve_fit`. Gli argomenti dell'istruzione sono, nell'ordine, il nome della funzione (che conviene definire nello stesso script), l'array da usare come variabile indipendente (nel nostro caso `x`), l'array dei dati dipendenti (nel nostro caso `y`), l'array dei valori iniziali dei parametri (che ha lunghezza pari al numero di parametri stesso, dunque 2 nel caso esaminato), e infine l'array da usare come σ_i nel calcolo del χ^2. Inoltre l'istruzione contiene un ulteriore *importante* comando che istruisce la routine sul "significato" da dare alle incertezze sperimentali: a questa istruzione si accede con l'opzione `absolute_sigma` ed essa, per la sua rilevanza, verrà trattata a parte nella sezione V.

Al termine della sua esecuzione, la routine restituisce nell'ordine questi due oggetti matematici: un array (nello script di esempio chiamato `pars`), che contiene il valore dei parametri ottenuto dal best-fit, e una matrice quadrata (`covm` nell'esempio), detta *matrice di covarianza* (o degli errori), il cui significato sarà discusso nella pros-

sima sezione. Sia array che matrice hanno dimensioni date dal numero di parametri della funzione: dunque nel nostro esempio l'array è un vettore di due elementi e la matrice comprende quattro elementi.

Un possibile script di Python per il best-fit numerico ai dati considerati (lo script si trova in rete con il nome `numer_lin_two_parms.py`) è riportato qui di seguito.

```
import pylab
import numpy
from scipy.optimize import curve_fit
# data load
x,Dx,y,Dy=pylab.loadtxt('nomefile.txt',unpack=True)
# scatter plot with error bars
pylab.errorbar(x,y,Dy,Dx,linestyle = '', color = 'black', marker = 'o')
# bellurie
pylab.rc('font',size=16)
pylab.xlabel('$\Delta V$  [V]')
pylab.ylabel('$I$  [mA]')
pylab.title('Data plot w numerical fit')
pylab.minorticks_on()

# make the array with initial values
init=(0,2)

# set the error
sigma=Dy
w=1/sigma**2

# define the linear function
# note how parameters are entered
# note the syntax
def ff(x, aa, bb):
    return aa+bb*x

# call the routine with the option absolute_sigma
pars,covm=curve_fit(ff,x,y,init,sigma,absolute_sigma=False)

# calculate the chisquare for the best-fit funtion
# note the indexing of the pars array elements
chi2 = ((w*(y-ff(x,pars[0],pars[1]))**2)).sum()

# determine the ndof
ndof=len(x)-len(init)

# print results on the console
print(pars)
print(covm)
print (chi2, ndof)
# print the same in a slightly more easy-to-read format
# (not very useful, though)
print('a = ', pars[0], '+/-', numpy.sqrt(covm[0,0]))
print('b = ', pars[1], '+/-', numpy.sqrt(covm[1,1]))
print('norm cov = ', covm[0,1]/(numpy.sqrt(covm[0,0]*covm[1,1])))

# prepare a dummy xx array (with 100 linearly spaced points)
xx=numpy.linspace(min(x),max(x),100)

# plot the fitting curve
pylab.plot(xx,ff(xx,pars[0],pars[1]), color='red')
```

```
# show the plot
pylab.show()
```

Osservate che uno script concepito in questo modo rende estremamente semplice eseguire il controllo preliminare sulla correttezza dei valori iniziali dei parametri di cui si trattava prima. A questo scopo è sufficiente "saltare" l'esecuzione del best-fit, commentando le istruzioni comprese tra `# call the routine` e `# prepare a dummy ....` Per commentare delle parti di script, è sufficiente farle precedere e seguire da un triplo apice (’’’). Inoltre, affinché la funzione venga calcolata con i dati iniziali, occorre sostituire l'istruzione `pylab.plot(xx,ff(xx,pars[0],pars[1]), color='red')` con `pylab.plot(xx,ff(xx,init[0], init[1]), color='red')`, che ha l'ovvio scopo di usare per il calcolo della funzione i valori dei parametri contenuti nell'array `init`. In pochi secondi il gioco è fatto.

Lo script contiene naturalmente anche tutte le istruzioni necessarie per creare il grafico, che però qui non viene riportato, visto che esso è del tutto indistinguibile da quello di Fig. 2. I risultati del fit, includendo anche la covarianza normalizzata, o correlazione, e l'indicazione dell'opzione di calcolo delle incertezze sui parametri, di cui discuteremo in seguito, sono:

$$I_0 = (-48 \pm 8)\ \text{mA} \tag{20}$$

$$R = (470 \pm 4)\ \text{ohm} \tag{21}$$

$$\text{norm.cov.} = -0.51 \tag{22}$$

$$\chi^2/\text{ndof} = 1.4/23 \tag{23}$$

$$\texttt{absolute\_sigma} = \texttt{False}\ . \tag{24}$$

Come si può facilmente notare, i valori ottenuti sono in completo accordo con quelli determinati attraverso il fit analitico (vedi Eq. 20), anche se è evidente come la procedura numerica porti a una diversa valutazione delle incertezze sui parametri, che in questo caso risulta minore con l'uso dell'algoritmo numerico.

Su questa discrepanza, che ha un significato un po' nascosto e sottile, torneremo nella sezione V.

A. La matrice di covarianza

Ci occupiamo qui di ricordare cosa è contenuto nella cosiddetta matrice di covarianza. Infatti, come già annunciato, l'uscita dell'algoritmo di minimizzazione comprende una matrice C (la `covm` dello script), che nel caso di fit a due parametri ha dimensioni 2×2, detta matrice di covarianza (o degli errori). Questa matrice è simmetrica e gli elementi sulla diagonale, C_{ii}, rappresentano il quadrato delle incertezze che il software attribuisce ai parametri del best-fit. Gli elementi fuori diagonale, $C_{ij} = C_{ji}$, sono invece rappresentativi della cosiddetta *covarianza* tra i parametri. Questi valori danno una misura di quanto *la variazione di un parametro influenzi la variazione dell'altro*.

Nel caso, semplice, di una retta che non passa per l'origine, come quello che stiamo considerando, è facilissimo rendersi conto che le variazioni della pendenza (parametro b) e dell'intercetta (parametro a, si intende intercetta con l'asse verticale del grafico) sono legate fra loro. Se provate a fare un best-fit "a mano", cioè provate a disegnare una retta che passi, per quanto possibile, attraverso le barre di errore di tutti i dati, potete rendervi facilmente conto che l'obiettivo può essere raggiunto per diverse (idealmente infinite) scelte di pendenza e intercetta. In particolare, per una retta come qui considerato, all'aumentare della pendenza dovrete diminuire l'intercetta, e viceversa. Questo esempio illustra il significato di covarianza: la variazione di un parametro può essere "compensata" dalla variazione dell'altro. Se, come in questo esempio, l'aumento di un parametro è compensato dalla diminuzione dell'altro, la covarianza è negativa, altrimenti essa è positiva. Una covarianza nulla, un caso praticamente irrealizzabile nei best-fit di interesse fisico, indica che i parametri sono completamente svincolati tra loro, cioè che la variazione dell'uno non è affatto correlata con quella dell'altro.

Per quantificare la covarianza si crea una quantità, detta *covarianza normalizzata*, o *coefficiente di correlazione*, che varia tra -1 e 1. Essa è data da

$$c_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}}\ . \tag{25}$$

Convenzionalmente, i parametri si dicono fortemente correlati (o anticorrelati, in caso di segno negativo) se $|c_{ij}| \gtrsim 0.8$ e totalmente correlati (o anticorrelati) se $|c_{ij}| \approx 1$ [7]. Nell'esempio considerato in questa nota i due parametri di fit sono scarsamente (anti)correlati fra loro. Parametri totalmente correlati o quasi completamente scorrelati sono sospetti, e possono indurre a chiedersi se il modello impiegato è quello realmente più adatto a descrivere le osservazioni sperimentali.

La covarianza, *nella sua forma normalizzata*, è una grandezza che, almeno per gli scopi didattici del nostro corso, deve essere *sempre* inclusa nei risultati di ogni best-fit a più di un parametro, a meno di diverse richieste. Molto spesso, infatti, i risultati di un best-fit vengono impiegati per fare delle "previsioni". Per esempio, nel caso che stiamo considerando, potrebbe essere richiesto di determinare l'intensità di corrente I' prevista per un certo valore $\Delta V'$ della differenza di potenziale. Come vedremo nel seguito, una previsione accurata richiede di tenere in debito conto della covarianza.

B. Richiami sulla covarianza

Immaginiamo di avere una funzione $f(\alpha, \beta)$ che dipende da due variabili α, β [8] e supponiamo che queste variabili corrispondano a due grandezze misurabili. La loro misura fornisce i valori α_i, β_i, con i che corre da 1 a N numero totale delle misure. In seguito alle misure, si possono determinare i valori medi $\overline{\alpha}$ e $\overline{\beta}$ e le rispettive varianze (sperimentali), definite come

$$\sigma_\alpha^2 = \frac{1}{N-1}\sum_i (\alpha_i - \overline{\alpha})^2 \tag{26}$$

$$\sigma_\beta^2 = \frac{1}{N-1}\sum_i (\beta_i - \overline{\beta})^2 \ . \tag{27}$$

La varianza della *grandezza* $f_i = f(\alpha_i, \beta_i)$ sarà definita come

$$\sigma_f^2 = \frac{1}{N-1}\sum_i (f(\alpha_i, \beta_i) - \overline{f})^2 \ , \tag{28}$$

con $\overline{f} = f(\overline{\alpha}, \overline{\beta})$. Sviluppiamo al primo ordine di Taylor la *funzione* $f(\alpha_i, \beta_i)$ attorno a $\overline{f}$:

$$f(\alpha_i, \beta_i) \simeq \overline{f} + (\alpha_i - \overline{\alpha})\frac{\partial f}{\partial \alpha} + (\beta_i - \overline{\beta})\frac{\partial f}{\partial \beta} \ . \tag{29}$$

Introducendo lo sviluppo nell'Eq. 28, si ottiene

$$\sigma_f^2 \simeq \frac{1}{N-1}\sum_i \left((\alpha_i - \overline{\alpha})\frac{\partial f}{\partial \alpha} + (\beta_i - \overline{\beta})\frac{\partial f}{\partial \beta}\right)^2 = \tag{30}$$

$$= \sigma_\alpha^2 \left(\frac{\partial f}{\partial \alpha}\right)^2 + \sigma_\beta^2 \left(\frac{\partial f}{\partial \beta}\right)^2 + 2\sigma_{\alpha\beta}\frac{\partial f}{\partial \alpha}\frac{\partial f}{\partial \beta} \ , \tag{31}$$

dove si definisce *covarianza* (sperimentale) delle grandezze α_i e β_i la

$$\sigma_{\alpha\beta} \equiv \frac{1}{N-1}\sum_i (\alpha_i - \overline{\alpha})(\beta_i - \overline{\beta}) \ . \tag{32}$$

Per semplificare ulteriormente la trattazione e per avvicinarci al caso di nostro interesse, supponiamo $f(\alpha, \beta) = (\alpha + \beta)$. In questo caso si ottiene immediatamente

$$\sigma_f^2 \simeq \sigma_\alpha^2 + \sigma_\beta^2 + 2\sigma_{\alpha\beta} \ . \tag{33}$$

La funzione di best-fit lineare $f = a + bx$ può essere interpretata come somma di due termini, cioè possiamo porre $\alpha = a$ e $\beta = bx$. Allora la varianza (sperimentale) sulla funzione può essere espressa dall'Eq. 33. Possiamo poi interpretare la procedura di best-fit come una sorta di esperimento (l'esperimento consiste nel disegnare una retta che passa per le barre di errore dei dati sperimentali) e la matrice di covarianza come indicativa delle varianze associate a questo esperimento. Questo equivale a porre $\sigma_\alpha^2 = C_{11}$, $\sigma_\beta^2 = C_{22}x^2$, $\sigma_{\alpha\beta} = C_{12}x$ (la presenza di x^2 e x è dovuta alla definizione di $\beta = bx$).

Fatte queste premesse, l'utilità pratica della covarianza è facile da capire. Avendo eseguito il best-fit dei nostri dati e ottenuto il risultato di Eq. 20-24, supponiamo di voler predire il valore I' dell'intensità di corrente che corrisponde a una certa differenza di potenziale $\Delta V'$. Sapendo che l'andamento è lineare secondo la legge $I = \Delta V/R + I_0$, ovvero $y = a + bx$, con a e b determinati dal fit assieme alle loro incertezze Δa e Δb, avremo $I' = \Delta V'/R + I_0$, ovvero $y' = a + bx'$, con ovvio significato dei termini.

Per determinare l'incertezza $\Delta I'$ da attribuire al valore I', ovvero $\Delta y'$ da attribuire a y', potremo utilizzare l'Eq. 33:

$$\Delta y' = \sigma_f = \sqrt{C_{11} + C_{22}x'^2 + 2C_{12}x'} \tag{34}$$

ovvero

$$\Delta I' = \sqrt{(\Delta a)^2 + (\Delta b)^2 \Delta V'^2 + 2c_{12}\Delta a \Delta b \Delta V'} \ , \tag{35}$$

dove abbiamo usato la covarianza normalizzata definita in Eq. 25 [9].

Se per esempio supponiamo $\Delta V' = 1$ V (senza incertezza), usando il risultato di Eq. 20 troviamo $I' = (50 \pm 8)$ mA. Notate che il segno negativo della covarianza "fa diminuire" l'incertezza sulla previsione rispetto a quanto si avrebbe senza considerare la covarianza [10]. Questa circostanza può essere qualitativamente generalizzata. Supponiamo di avere dei dati sperimentali che possono essere fittati con diverse funzioni modello, contenenti un diverso numero di parametri. Per esempio, immaginiamo che i dati possano essere rappresentati da una retta che passa o non passa per l'origine, cioè con funzioni che hanno rispettivamente uno e due parametri liberi (fisicamente, la situazione potrebbe essere quella di un andamento lineare in cui c'è un piccolo offset costante, che viene ritenuto trascurabile a priori se nel fit si usa la retta che passa per l'origine). Nel caso di covarianza negativa, il parametro con lo stesso significato, cioè la pendenza della retta, uscirà dal fit con una incertezza minore se il fit stesso è eseguito con due parametri liberi. Viceversa, se la covarianza è positiva, esso uscirà presumibilmente con una incertezza maggiore.

C. Incertezza Δx_i

Il best-fit del minimo χ^2 condotto sui nostri dati esempio è stato svolto trascurando, di fatto, l'incertezza Δx_i sulle grandezze x_i. Se e quanto questa scelta sia ragionevole dipende, ovviamente, dai dati acquisiti. Per esercizio, possiamo provare a considerare questa incertezza nel best-fit (solo quello numerico per evitare inutili appesantimenti).

Come espresso nell'Eq. 3, una possibile strategia per tenere conto dell'incertezza Δx_i consiste nel propagarne l'effetto sul valore di y_i. Nel caso lineare che stiamo considerando, l'espressione da impiegare è semplicissima. Si ottiene infatti $\sigma_i^2 = (\Delta y_i)^2 + (b\Delta x_i)^2$.

L'implementazione pratica è anche semplicissima: nella riga di script in cui si "definiscono le incertezze" (il commento è `# set the error`) è sufficiente sostituire la riga `sigma = Dy` con `sigma = numpy.sqrt(Dy**2+(bbb*Dx)**2)`, dove `bbb` rappresenta il valore del parametro b del fit. Come già discusso, questo parametro non può essere conosciuto con certezza a priori, dato che la sua valutazione è eseguita proprio dal best-fit. Dunque il consiglio è di eseguire prima un best-fit trascurando Δx_i (come fatto in precedenza) e quindi eseguire un nuovo best-fit considerando anche la propagazione dell'incertezza Δx_i. In genere non c'è alcun bisogno di iterare la procedura, poiché i nuovi valori dei parametri di best-fit restano in buon accordo con quelli determinati trascurando le incertezze Δx_i.

L'esito di questa implementazione dipende ovviamente in modo molto specifico dai dati. Nel caso del nostro esempio essa non conduce a modifiche significative dei risultati espressi in Eq. 20, a parte, come atteso, una ulteriore diminuzione del χ^2 (che diventa $\chi^2 = 1.3$). Se ne conclude che, per l'esempio considerato, le incertezze Δx_i giocano un ruolo effettivamente trascurabile.

D. Residui normalizzati

Un'ulteriore analisi (qualitativa) che è spesso utile compiere a posteriori sul best-fit è quella detta *dei residui*. I residui sono, come noto, le differenze $y_i - f(x_i)$, dove la funzione $f(x_i)$ è calcolata con i parametri ottenuti dal best fit. Nel caso di best-fit del minimo χ^2 possono essere analizzati in particolare i *residui normalizzati* r_i definiti come

$$r_i = \frac{y_i - f(x_i)}{\sigma_i} \ , \tag{36}$$

dove $\sigma_i = \Delta y_i$ nel caso in cui siano trascurabili le incertezze Δx_i (questo è il caso a cui faremo riferimento qui).

Idealmente, cioè per un campione di dati corredato da incertezze *determinate statisticamente* e ben descritto dalla funzione di fit prescelta, r_i dovrebbe essere una variabile aleatoria standard, con una distribuzione Gaussiana a media nulla e varianza unitaria. Dunque un'analisi corretta dei residui (normalizzati) potrebbe essere compiuta costruendo l'istogramma della distribuzione di r_i e valutando se esso è ben descritto da una Gaussiana. Tuttavia questa operazione ha senso solo in presenza di un campione di tante misure (N molto grande), che non è la situazione del nostro esempio. In alternativa, si può costruire un grafico dei residui (normalizzati) r_i in funzione di x_i e valutare "a occhio" se questo grafico presenta degli andamenti evidenti. Può infatti verificarsi che i residui siano distribuiti in modo palesemente disomogeneo. Supponendo una funzione di fit lineare, come in questa nota, se i dati contengono delle non linearità queste possono essere messe in evidenza dall'analisi dei residui. Infatti le discrepanze tra dati e previsioni del fit (lineare) sono attese essere più marcate agli estremi dell'intervallo di dati considerato. Inoltre nella pratica sperimentale succede spesso che i dati vengano acquisiti in modo "sequenziale", uno dopo l'altro. Esistono dei *rumori* periodici, cioè, nel linguaggio di questa nota, degli errori sistematici (definiremo qualitativamente il "rumore" in altra sede), che possono sovrapporsi all'esito della misura. Il carattere sistematico può talvolta essere evidenziato da un andamento ciclico nel grafico dei residui.

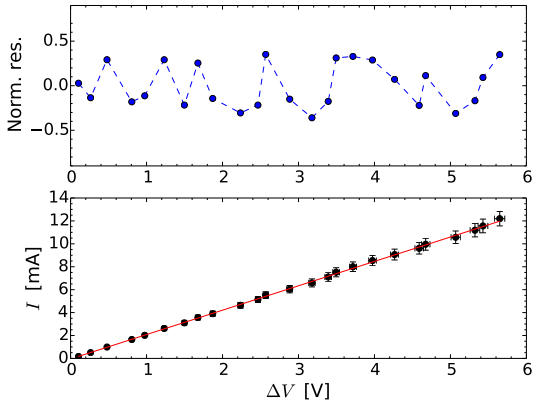

Figura 3. Grafici dei dati con sovrapposto il best-fit numerico (in basso) e dei residui normalizzati (in alto).

La Fig. 3 riporta il grafico dei dati con best-fit numerico (in sostanza analogo a quello di Fig. 2) assieme al grafico dei residui normalizzati. Si vede come, nel caso considerato, non ci siano particolari andamenti, per cui nulla si può concludere da questa analisi. La figura è stata costruita con lo script disponibile in rete con il nome `numer_lin_two_parms_res.py`, che propone anche l'impiego dei subplots, un modo per rappresentare su una stessa figura diversi grafici. Esso è riportato qui di seguito.

```
import pylab
import numpy
from scipy.optimize import curve_fit
x,Dx,y,Dy=pylab.loadtxt('nomefile.txt',unpack=True)

# use subplots to display two plots in one figure
# note the syntax
```

```
pylab.subplot(2,1,2)

pylab.errorbar(x,y,Dy,Dx,linestyle = '', color = 'black', marker = 'o')
pylab.rc('font',size=16)
pylab.xlabel('$\Delta V$  [V]',size=18)
pylab.ylabel('$I$  [mA]', size=18)
pylab.minorticks_on()

init=(0,2)
sigma=Dy
w=1/sigma**2

def ff(x, aa, bb):
    return aa+bb*x

pars,covm=curve_fit(ff,x,y,init,sigma,absolute_sigma=False)

chi2 = ((w*(y-ff(x,pars[0],pars[1]))**2)).sum()
ndof=len(x)-len(init)

print(pars)
print(covm)
print (chi2, ndof)
print('a = ', pars[0], '+/-', numpy.sqrt(covm[0,0]))
print('b = ', pars[1], '+/-', numpy.sqrt(covm[1,1]))
print('norm cov = ', covm[0,1]/(numpy.sqrt(covm[0,0]*covm[1,1])))

xx=numpy.linspace(min(x),max(x),100)
pylab.plot(xx,ff(xx,pars[0],pars[1]), color='red')

# switch to the residual plot
pylab.subplot(2,1,1)

# build the array of the normalized residuals
r = (y-ff(x,pars[0],pars[1]))/sigma

# bellurie
pylab.rc('font',size=16)
pylab.ylabel('Norm. res.',size=18)
pylab.minorticks_on()
# set the vertical range for the norm res
pylab.ylim((-.9,.9))

# plot residuals as a scatter plot
pylab.plot(x,r,linestyle="--",color='blue',marker='o')

pylab.show()
```

V. ABSOLUTE SIGMA

Il titolo criptico di questa sezione, a cui abbiamo fatto cenno in precedenza, nasconde una particolarità della procedura di best-fit di Python: tra le opzioni del comando `curve_fit` ce ne è una che suona piuttosto misteriosa, e rimane tale anche dopo aver consultato la scarna e confusa documentazione disponibile in rete.

L'opzione in questione è `absolute_sigma`, che può avere come valore `False` (*di default*, cioè attiva anche senza porre alcuna specifica nella linea di comando di `curve_fit`) oppure `True` [12]. Potete verificare facilmente come la scelta di questa opzione non modifichi il valore dei parametri risultanti dal best-fit, ma come essa agisca, spesso in maniera non trascurabile, sulla determinazione dell'*incertezza* da attribuire al valore del parametro. In

particolare, confrontando i risultati del best-fit analitico e di quello numerico (per un andamento lineare, come nell'esempio di questa nota), potrete facilmente verificare che l'incertezza sui parametri data da quest'ultimo è in accordo con quella del fit analitico *solo* se l'opzione scelta è `True`, cioè diversa da quella di default.

Nel nostro percorso di "spiegazione" per l'esistenza di questa opzione, partiamo da un'ovvia considerazione: la maggior parte dei metodi di data reduction and analysis discendono dalla circostanza di avere a disposizione campioni di dati in cui l'errore ha un'origine prevalentemente stocastica. Per esempio la propagazione dell'errore e il metodo del minimo χ^2 presuppongono di trattare grandezze affette (principalmente) da errore stocastico. Per il best-fit, questo significa poter pesare i residui quadrati con l'inverso della varianza sperimentale (supposta rappresentativa della varianza della parent distribution), e di poter utilizzare il valore del χ^2 ottenuto dal fit come indicatore della sua affidabilità, o significatività (test del χ^2).

Come già abbiamo sottolineato, la realtà sperimentale tipica delle misure di segnali elettrici è, purtroppo, ben diversa, a causa della pressoché inevitabile preponderanza dell'errore sistematico dovuto alla calibrazione. Di conseguenza, le condizioni nelle quali ci troviamo ad operare, per necessità, o, qualche volta, per praticità, sono le peggiori per l'applicazione rigorosa dei metodi di data reduction and analysis [13]. In particolare, la sovrastima delle incertezze sui dati dovuta alla considerazione dell'errore sistematico comporta, molto spesso, una sorta di sottostima del χ^2 ottenuto dal best-fit, e la conseguente impossibilità di servirsi del test del χ^2 con cognizione di causa.

È facilissimo verificare cosa determina la scelta `absolute_sigma = True` rispetto a quella, di default (dunque valida anche senza specificare l'opzione) `absolute_sigma = False`. Detta Δp_m l'incertezza sul parametro p_m del best-fit (per intenderci, nel caso della retta e usando la nomenclatura precedente si ha $p_1 = a$ e $p_2 = b$), cioè la grandezza ottenuta estraendo la radice quadrata degli elementi diagonali della matrice di covarianza, si ha

$$\Delta p_m|_{\rm False} = \sqrt{\chi^2_{rid}}\Delta p_m|_{\rm True} \, . \tag{37}$$

In altre parole, l'incertezza sui parametri di fit data dall'opzione di default (`False`) è "pesata", o "normalizzata", con la radice quadrata del χ^2_{rid}.

Come illustreremo nel seguito, l'opzione `True` conduce a una valutazione dell'errore sui parametri che è in linea con le aspettative naïf. Supponiamo infatti di voler fittare dai dati a una retta: operando in modo grafico (con matita e righello), possiamo facilmente dedurre che maggiore è l'entità delle barre di errore dei nostri dati, maggiore è l'incertezza con cui possiamo determinare i parametri della retta di fit, cioè intercetta e coefficiente angolare. Questo è in effetti quanto si verifica con la procedura *analitica* di best-fit lineare, come potreste facilmente provare aumentando, o diminuendo, ad arte le incertezze dell'esempio riportato in questa nota.

Partendo dalla supposizione che chi ha scritto la procedura `curve_fit` volesse, con la scelta di default `False`, soddisfare le esigenze di best-fit più comuni, che effettivamente sono quelle in cui l'errore sistematico prevale, possiamo cercare un rationale per la normalizzazione effettuata dalla procedura stessa. È ovvio che, nel caso di sovrastima delle barre di errore, come si ha quando l'errore sistematico prevale (almeno nelle nostre esperienze tipiche), il χ^2_{rid} tende a valere meno di uno, per cui la normalizzazione di Eq. 37 agisce in modo da ridurre l'incertezza sulla valutazione dei parametri. Viceversa, nel caso in cui le barre di errore fossero, per qualche motivo, sottostimate, il χ^2_{rid} tenderebbe a essere ben maggiore di uno, e quindi la normalizzazione agirebbe in senso opposto. Per un best-fit "a regola d'arte", cioè realizzato con una appropriata funzione modello e facendo uso di incertezze di origine rigorosamente stocastica, allora $\chi^2_{rid} \simeq 1$ e la normalizzazione non ha alcun effetto sulla valutazione dell'incertezza sui parametri, cioè l'incertezza sui parametri è la stessa indipendentemente dall'opzione.

A. Asymptotic vs standard

La definizione comunemente accettata per l'incertezza sui parametri risultanti da un best-fit è la seguente [14]: Δp_m costituisce l'intervallo di variazione di p_m che corrisponde a un aumento unitario del χ^2. Questa definizione è in linea con quelle di incertezza per altre grandezze, misurate o calcolate, per cui essa prende qualche volta il nome di *standard error*.

Accanto a questa definizione ne esiste, anche se in forma un po' clandestina, un'altra, a cui qualche volta si fa riferimento con il nome *asymptotic error*. Nel caso di misure, il carattere asintotico significa che questa incertezza sarebbe quella determinata su un campione contenente un numero infinito di elementi. Se l'errore è dominato da fluttuazioni statistiche, allora le due definizioni di incertezza coincidono perfettamente.

Nel caso del best-fit su dati pesati, cioè dotati di barre di errore, l'errore asintotico sui parametri del best-fit può essere inteso come l'incertezza che si avrebbe se il numero di dati fosse molto grande e se la loro incertezza di misura fosse di origine puramente stocastica. In queste condizioni il metodo del minimo χ^2 stabilisce $\chi^2_{rid} \simeq 1$ (il best-fit "a regola d'arte", secondo quanto scritto prima). Allora possiamo rifrasare la definizione di errore asintotico sui parametri del best-fit, dicendo che esso rappresenta l'incertezza che sarebbe attribuita alla valutazione dei parametri *se fosse* $\chi^2_{rid} \simeq 1$.

In sostanza, la procedura `curve_fit` di Python consente di scegliere le due definizioni di errore sui parametri:

`absolute_sigma = False` → asymptotic error

`absolute_sigma = True` → standard error .

Dimostrare che la normalizzazione attraverso $\sqrt{\chi^2_{rid}}$ espressa in Eq. 37 conduce effettivamente a quanto affermato sopra è compito matematicamente complicato. Qualche hint può essere trovato in [15] e, meglio ancora, in [16], dove si illustra un pacchetto di Python, chiamato `kmpfit`, che, basandosi su `scipy.optimize`, permette di eseguire best-fit con un certo numero di funzionalità aggiuntive pre-assemblate.

Ho fatto un rapido controllo per verificare quale sia la definizione di errore sui parametri di best-fit in uso in alcuni software di trattamento dati (nei limiti delle mie possibilità). Usano l'asymptotic error, corrispondente all'opzione `False` di Python: MATLAB (R2015b), gnuplot (5.0.0), Mathematica (6.0, in forma piuttosto complicata); usa lo standard error, corrispondente all'opzione `True` di Python, IgorPro (6.37 e seguenti), un software orientato più a trattare dati di interesse per la fisica che non per l'ingegneria. OriginPro, che non uso, dovrebbe consentire la scelta dei due metodi attraverso opportuna definizione dell'incertezza di misura sui dati da fittare e similmente dovrebbero comportarsi, per quanto ne so, altri software pensati per analisi di dati in fisica.

Il messaggio principale di questa sezione è che la determinazione dell'incertezza sui parametri individuati dal best-fit è "critica", potendo dipendere dalla valutazione delle barre di errore sui dati da fittare, che è a sua volta un'operazione delicata da compiere. Infatti, la situazione "ideale", in cui gli errori di misura sono di origine prevalentemente stocastica, si incontra raramente nella pratica delle nostre esperienze. Eseguire un best-fit richiede attenzione in tutti i passaggi coinvolti e nella valutazione dei risultati: l'incertezza sui valori dei parametri ottenuti dal best-fit deve essere considerata con attenzione per evitare conclusioni erronee ed è altrettanto *necessario* (leggasi obbligatorio) specificare sempre, tra i risultati del best-fit, quale opzione è stata scelta.

Python conosce la situazione e ci viene incontro, proponendo appunto la scelta fra le due opzioni, pur poco chiare e documentate malissimo. Dal punto di vista operativo, le osservazioni che abbiamo svolto suggeriscono che:

1. se gli errori dei dati sperimentali sono di origine prevalentemente stocastica, l'opzione da usare è `True`;

2. se invece l'origine degli errori è poco conosciuta, oppure se si sa che il contributo sistematico è dominante, come si verifica molto spesso nelle nostre esperienze, *è consigliata* l'opzione `False`, quella di default, che consente di evitare marchiane sovrastime (in qualche caso sottostime) dell'incertezza sui valori di parametri di best-fit.

VI. ALTRI POSSIBILI INDICATORI DEL BEST-FIT

In conclusione di questa nota, facciamo un breve cenno ad altri indicatori della qualità del best-fit che qualche volta si usano nella pratica, in particolare con softwares diversi da Python. In termini generali, di questi indicatori non faremo uso, anche perché essi sono raramente citati nell'analisi di esperimenti di fisica (piacciono di più ad altre categorie scientifiche, come biologi, economisti, etc.). Dunque considerate questa sezione come una sorta di piccolo approfondimento per scopi di completezza.

- *Confidence level*: come già affermato, e a voi ben noto dallo scorso anno, assumendo che il best-fit sia "a regola d'arte" (funzione modello appropriata, errore stocastico sui dati, misure scorrelate tra loro e, in definitiva, $\chi^2_{rid} \simeq 1$) , può fare comodo esplicitare il livello di confidenza con cui il best-fit individua il valore dei parametri usando, in pratica, il *test del* χ^2. In questo contesto, l'errore standard definito prima corrisponde a una deviazione standard nella distribuzione del χ^2 (che qui è ovviamente supposta valida) e quindi a un livello di confidenza del 68%; un livello di confidenza del 95% corrisponde a due deviazioni standard, del 99% a tre deviazioni standard.

- *R-squared*: nella nostra pratica di impiego del best-fit siamo abituati a valutare la "bontà" del fit attraverso l'analisi dei residui normalizzati, della covarianza normalizzata (se c'è più di un parametro) e infine del χ^2_{rid}. Poiché normalmente abbiamo ottimi motivi per aspettarci che il χ^2_{rid} sia viziato dalla non corretta identificazione dell'errore sui dati sperimentali, generalmente sovrastimati a causa del contributo sistematico, può in qualche caso essere utile estrarre altri parametri numerici che informino sulla qualità del best-fit, senza essere così fortemente condizionati dalla determinazione delle barre di errore. Di questi parametri ne esistono parecchi (si veda, ad esempio, l'approccio ANOVA [17], che, essendo stato inventato per analizzare le covarianze in popolazioni statistiche, può essere esteso all'esame dei risultati del best-fit). Uno particolarmente semplice da calcolare è l'R-squared (R^2), definito come

$$R^2 = 1 - \frac{\sum_i \left((y_i - f(x_i))^2/\sigma_i^2\right)}{\sum_i \left((y_i - \overline{y})^2/\sigma_i^2\right)} \,, \tag{38}$$

  dove (x_i, y_i) sono le coppie di dati sperimentali da fittare, σ_i rappresenta la deviazione standard usata come peso per il best-fit, $\overline{y}$ è il *valore medio* dei dati acquisiti e le somme si intendono estese su tutto il campione di misure. Questo parametro dà un'indicazione della capacità del fit di riprodurre la variazione dei dati osservati e si può dimostrare [18] che esso, normalmente compreso tra 0 e 1,

tende tanto più all'unità quanto meglio la funzione modello è in grado di interpretare l'osservazione sperimentale. Ciò può essere facilmente compreso in termini qualitativi: per un "buon" fit ci si aspetta che la somma dei residui quadri, che compare, debitamente pesata, al numeratore della frazione, sia molto minore dello scarto quadratico medio dei dati che sta al denominatore, per cui la funzione di Eq. 38 tende in queste condizioni all'unità. Viceversa, essa tende a zero nel caso in cui residui e scarto siano paragonabili tra loro, circostanza che si verifica quando il best-fit non riproduce in modo corretto le osservazioni sperimentali [19].

APPENDICE: FORMULE ANALITICHE PER IL BEST-FIT A UNA RETTA

Questa Appendice è dedicata a una dimostrazione (semplificata, per quanto possibile) delle Eqq. 4-9. Partiamo da una situazione generale, in cui supponiamo di avere una funzione modello generica $f(x)$ che contiene diversi parametri p_m (quelli da tenere liberi nel best-fit). Il problema del best-fit si riduce a determinare il set di parametri p_m che minimizzano la Eq. 2, affermazione che porta a determinare il set di parametri per cui

$$\frac{\partial \left[\sum_i \frac{(y_i - f(x_i))^2}{(\Delta y_i)^2}\right]}{\partial p_m} = 0 \,, \tag{39}$$

dove si intende che viene implementato in qualche modo il controllo che la derivata nulla non corrisponda a un massimo.

Poiché la funzione $f(x)$ dipende dall'*intero* set di parametri, il problema è un buon esempio di ricerca di *minimo condizionato*. In termini generali, il minimo condizionato si può trovare con il metodo dei *moltiplicatori di Lagrange*, che probabilmente studierete in altra sede. Questo metodo conduce facilmente a una matematica abbastanza ostica, incompatibile con gli scopi di questa Appendice. Fortunatamente, se ci si restringe ad esaminare il caso di interesse per questa nota, la retta $f(x) = a + bx$ che ha parametri $p_1 = a$ e $p_2 = b$, la soluzione può essere determinata in maniera sufficientemente rapida anche senza fare uso del metodo generale dei moltiplicatori. Inoltre, al puro scopo di ripulire ulteriormente le formule, possiamo immaginare di partire con un fit dei minimi quadrati, in cui, in sostanza, supponiamo $\Delta y_i = 1$ in Eq. 39 (generalizzeremo poi al fit del minimo χ^2 che è di nostro interesse).

Grazie alla semplice espressione matematica della $f(x)$ in questo caso, possiamo facilmente scrivere le due equazioni di Eq. 39, ottenendo

$$\frac{\partial \left[\sum_i (y_i - a - bx_i)^2\right]}{\partial a} = \tag{40}$$

$$= -2 \sum_i (y_i - a - bx_i) = 0 \,, \tag{41}$$

e

$$\frac{\partial \left[\sum_i (y_i - a - bx_i)^2\right]}{\partial b} = \tag{42}$$

$$= -2 \sum_i x_i (y_i - a - bx_i) = 0 \,. \tag{43}$$

Lavorando di semplice algebra, si ha dalla prima

$$aN + b \sum_i x_i = \sum_i y_i \tag{44}$$

e dalla seconda

$$a \sum_i x_i + b \sum_i x_i^2 = \sum_i x_i y_i \,, \tag{45}$$

dove N è il numero di coppie di dati sperimentali, cioè il valore massimo che può assumere l'indice i delle sommatorie. Nonostante l'apparente complessità, quello appena scritto è un sistema lineare di due equazioni algebriche, che porta alle soluzioni

$$a = \frac{\sum_i x_i^2 \sum_i y_i - \sum_i x_i \sum_i x_i y_i}{N \sum_i x_i^2 - (\sum_i x_i)^2} \tag{46}$$

$$b = \frac{N \sum_i x_i y_i - \sum_i x_i \sum_i y_i}{N \sum_i x_i^2 - (\sum_i x_i)^2} \tag{47}$$

L'estensione al best-fit del minimo χ^2 è piuttosto immediata dal punto di vista concettuale, anche se un po' più complicata come scrittura. Stavolta nelle sommatorie delle Eqq. 40-43 compaiono dei termini $w_i = 1/(\Delta y_i)^2$ a moltiplicare, per cui le Eqq. 44 e 45 diventano

$$a \sum_i w_i + b \sum_i x_i w_i = \sum_i y_i w_i \tag{48}$$

$$a \sum_i x_i w_i + b \sum_i x_i^2 w_i = \sum_i x_i y_i w_i \,. \tag{49}$$

Anche senza ripetere il procedimento nei dettagli, si vede come le soluzioni abbiano la stessa "struttura" di Eqq. 46-47, solo che in tutti i termini delle sommatorie compare a moltiplicare un w_i e che N viene rimpiazzato da $\sum_i w_i$. Quindi, in sostanza, si ottengono le soluzioni espresse nelle Eqq. 4-9 (in quelle espressioni il denominatore delle frazioni è stato indicato con Δ'). Infine, per quanto riguarda la stima sulle incertezze dei parametri, Δa e Δb, esse possono essere determinate, al costo di diverse paginate di passaggi (che qui non si riportano per ovvi motivi), applicando le regole di propagazione dell'errore alle soluzioni appena determinate.

[1] Occorre naturalmente distinguere tra nome delle grandezze e loro unità di misura. Per indicare l'unità di misura si usano diverse convenzioni, per esempio mettendone l'espressione simbolica tra parentesi quadre usando caratteri normali (non corsivi). Se necessario, è possibile usare prefissi moltiplicativi (M, k, m, μ, n, etc.). Nel caso in cui l'unità di misura sia arbitraria, si può usare [arb.un.] (il termine [a.u.] indica le unità atomiche). Naturalmente, se l'unità di misura non c'è (grandezze adimensionali che non hanno unità di misura), essa non va espressa. Lo stesso si deve fare quando la grandezza rappresentata è "normalizzata", cioè ottenuta dal rapporto tra la grandezza (per esempio misurata) e una grandezza di riferimento, che deve avere le stesse dimensioni e unità di misura della prima.

[2] Per motivi oscuri, molto spesso la scelta di default per il character size dei valori sugli assi è troppo piccolo per essere apprezzato in una stampa.

[3] In alcuni casi, a cui faremo cenno in seguito, è invece conveniente eseguire una *linearizzazione dei dati*, cioè una manipolazione matematica dei dati di partenza che permetta di rendere lineare il loro andamento. Questa operazione, che deve essere accompagnata da opportune manipolazioni delle barre di errore, è ben diversa rispetto a quella che prevede l'uso della rappresentazione logaritmica o semi-logaritmica.

[4] L'analisi delle sorgenti di errore e delle incertezze da associare alle misure di grandezze elettriche effettuate con strumenti standard sarà oggetto di approfondite considerazioni più avanti.

[5] I.G. Hughes and T.P.A. Hase, *Measurements and their uncertainties* (Oxford University Press, Oxford, 2013).

[6] Nel *test del* χ^2 si assume generalmente che il *livello di confidenza* sia attorno al 95% quando il valore del χ^2 è compreso tra $\mu_{\chi 2} - 2\sigma_{\chi 2}$ e $\mu_{\chi 2} + 2\sigma_{\chi 2}$, dove $\mu_{\chi 2}$ e $\sigma_{\chi 2}$ sono rispettivamente il valore medio e la deviazione standard della *distribuzione del* χ^2. Infatti questo livello di confidenza corrisponde a uno spread dei dati inferiore a 2 deviazioni standard ("2 sigma") rispetto alla media attesa, che è convenzionalmente preso come indice di una ragionevole attendibilità. Per la distribuzione del χ^2 si ha $\mu_{\chi 2} \simeq N$ e $\sigma_{\chi 2} \simeq \sqrt{2N}$. Introducendo il χ^2_{rid}, si ottiene che il livello di confidenza del 95% corrisponde a $1 - 2\sqrt{2/N} < \chi^2_{rid} < 1 + 2\sqrt{2/N}$.

[7] Un esempio clamoroso di correlazione completa è quello in cui il parametro τ di un decadimento esponenziale viene espresso come prodotto di altri due parametri che non compaiono in altre parti della funzione di best-fit. I due parametri in questione sono ovviamente del tutto (anti)correlati, e il best-fit non può dare risultati affidabili.

[8] La trattazione può essere estesa a funzioni con più di due variabili: qui ci si limita a due per semplicità.

[9] Purtroppo l'esempio considerato è un po' incasinato in termini di parametri di fit e grandezze fisiche ad essi associate. L'espressione di Eq. 34 mostra che è estremamente utile usare direttamente i valori che compaiono nella matrice di covarianza per eseguire la stima discussa nel testo, anche se questi valori non sono immediatamente riferibili a grandezze fisiche.

[10] Che la covarianza negativa implichi una "diminuzione" nell'incertezza della previsione è diretta conseguenza del fatto che la funzione di best-fit è esprimibile come una somma. Espressioni diverse si trovano in altri casi [5], come si può facilmente determinare calcolandone la covarianza.

[11] L'argomento di questa sezione è stato messo in luce grazie alle utili e brillanti osservazioni di due vostri colleghi, FLD e AC, dell'anno 2015/16. A loro va un enorme ringraziamento.

[12] L'opzione è implementata solo a partire da una certa versione di Python, o delle sue librerie. Dunque se incappate in un errore di compilazione relativo a questa opzione, vuol dire semplicemente che essa non è disponibile nella versione di Python che state impiegando. In questo caso, l'opzione è fissa al valore `False`.

[13] Spesso è possibile compiere degli sforzi finalizzati a migliorare questa situazione. Per esempio, come vedremo in futuro, nelle misure automatizzate con Arduino si può in genere prevedere la possibilità di estrarre la deviazione standard sperimentale su un campione ragionevolmente esteso di misure ripetute in automatico. Questo è il punto di partenza per compiere una distinzione tra contributi stocastici e sistematici all'errore. Si potrebbe ad esempio eseguire gli eventuali best-fit per l'analisi dei dati usando solo l'incertezza stocastica e poi tenere conto del contributo sistematico nella conversione dei risultati del best-fit a unità "fisiche". Avremo modo di trattare questo argomento in seguito.

[14] P.R. Bevington and D.K. Robinson, Data reduction and error analysis for the physical sciences (McGraw-Hill, New York, 2002).

[15] P.H. Richter, "Estimating Errors in Least-Squares Fitting", TDA Progress Report 42-122 (1995); `http://ipnpr.jpl.nasa.gov/progress_report/42-122/122E.pdf`

[16] `https://www.astro.rug.nl/software/kapteyn/kmpfittutorial.html`

[17] `https://en.wikipedia.org/wiki/Analysis_of_variance`

[18] `https://en.wikipedia.org/wiki/Coefficient_of_determination`

[19] Il parametro R-squared nasce per l'analisi della significatività di best-fit lineari. La sua estensione a best-fit non lineari non è scontata, specie per alcune tipologie di funzioni modello, ma tuttavia esso è spesso impiegato in maniera estensiva.