Best-fit (non lineare) e Python

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

(Dated: version 3 - FF, 6 novembre 2014)

Eseguire e valutare i risultati di un best-fit, in particolare nel caso in cui la funzione di fit sia *non lineare*, è un'operazione generalmente delicata, la cui completa padronanza si acquisisce solo con l'esperienza, tanta esperienza. Di fatto, ogni problema di best-fit può presentare criticità specifiche legate per esempio alla scelta della funzione, alla caratteristiche del data set, alla valutazione dell'incertezza sperimentale, che rendono difficile generalizzare approcci e metodi. In questa nota si fa riferimento alle misure sulla legge di Ohm in presenza di un generatore reale svolte in una esercitazione pratica del corso che, pur non essendo particolarmente adatta, si presta bene come esempio su cui realizzare diversi best-fit e vedere cosa si può ottenere dalla procedura. Inoltre questa nota ha anche (ma non solo!) lo scopo di rinfrescare le conoscenze di "Python" necessarie per svolgere e presentare correttamente un best-fit numerico. Tutto quanto qui mostrato a proposito della programmazione di Python è la trasposizione più o meno diretta di quanto avete visto lo scorso anno (tutto può essere trovato negli appunti di Luca Baldini e Carmelo Sgrò, lots of kudos to them!); non escludo che in futuro troveremo altre forme in grado di realizzare best-fit con un po' più di flessibilità e interattività, così come sicuramente incontreremo altre tipologie di problemi relativi al best-fit.

I. BEST-FIT

Eseguire un best-fit è una delle operazioni più frequenti nella pratica sperimentale. Fare un best-fit *non* significa solo far passare una curva analitica su un set di dati sperimentali, operazione in genere poco significativa. Fare un best-fit conduce, o può condurre, anche ad altri importanti risultati, tra cui:

1. determinare quantitativamente parametri incogniti 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 casi in cui è disponibile una sufficiente "statistica" per le misure, determinare la significatività dell'interpretazione.

Nella maggior parte delle situazioni sperimentali questi obiettivi possono, o devono, essere perseguiti usando dei metodi *numerici*, cioè fidandosi di procedure di calcolo eseguite da un computer (il cosiddetto "algoritmo", in gergo). Infatti, solo per casi estremamente semplici è possibile determinare le relazioni necessarie attraverso procedure analitiche: un caso accessibile con procedure analitiche riguarda il best-fit a una funzione lineare, per la quale è relativamente facile giungere a relazioni matematiche semplici attraverso la minimizzazione analitica delle funzioni coinvolte. Modelli che invece conducono a funzioni non lineari devono essere trattati numericamente, situazione che impone cura per evitare di trarre conclusioni ingiustificate. In questo rispetto, l'uso di "Python" (con i pacchetti che conoscete) rappresenta una scelta adeguata, dato che i pacchetti implementano un Levenberg-Marquardt Algorithm (LMA), una procedura numerica per la ricerca di minimi locali generalmente efficiente e affidabile, che viene applicata alla somma dei residui (quadrati).

D'altra parte è ovvio che, ogni volta che questo è possibile, cioè, nella pratica, quando la funzione di fit è, o può essere resa, lineare, si deve preferire l'uso delle espressioni analitiche di minimizzazione, per le quali i vari aspetti critici discussi in questa nota non si applicano o danno effetti facilmente prevedibili.

A. Richiami su fit e minimo χ^2

Eseguire un best-fit (detto, in questo caso, 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 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:

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

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 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: $\chi^2 = \Sigma\xi^2$. 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 n e la deviazione standard $\sqrt{2n}$.

Affinché ciò sia verificato, occorre che la variabile aleatoria ξ 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 assunzione. Infatti riteniamo di poter sostituire la varianza dell'(ipotetica) distribuzione delle ξ con Δy_i^2, operazione che, come vedremo un po' meglio nel seguito, non è sempre e necessariamente giustificata.

Per ora notiamo che, se le condizioni che abbiamo posto sono ritenute ragionevoli, si assume di fatto $\xi = (y_i - f(x_i))^2/(\Delta y_i)^2$. A questo punto, il metodo 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). 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, e quindi del parametro usato per valutare la significatività del best-fit.

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, dunque affette da incertezza *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.

3. Aspetto che combina i due punti precedentemente elencati: molto spesso le misure eseguite con gli strumenti tipici di Laboratorio 2 (tester, oscilloscopio) sono corredate da un errore dominato dall'incertezza strumentale di calibrazione, che tipicamente copre l'errore stocastico. Talvolta questo conduce a sovrastimare l'incertezza, con l'ovvia conseguenza di *sottostimare* il χ^2 ottenuto e prevenire la possibilità di trarre conclusioni sulla significatività. Naturalmente, però, è sempre possibile confrontare i χ^2 ottenuti con due modelli diversi.

4. Spesso succede che nella misura anche la variabile che compare nell'asse orizzontale, che qui chiamiamo x_i e definiamo "variabile indipendente" in analogia con la matematica delle funzioni, sia affetta da un'incertezza non trascurabile, essendo essa stessa il risultato di una misura. Considerare solo l'incertezza Δy_i è chiaramente una limitazione concettuale da non sottovalutare. Essa infatti può condurre a *sovrastimare* il χ^2 ottenuto attraverso la sottostima delle incertezze. Un modo (non pulito e non sicuro!) per tenere conto delle incertezze Δx_i può essere quello di basarsi sulla propagazione degli errori. In pratica si determina a priori il contributo sull'incertezza in y_i, $\Delta y_i|_{\Delta x_i}$, 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 porre:

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

Questa espressione ha lo svantaggio di richiedere 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.

II. MODUS OPERANDI

Vediamo ora alcune generalità sul modo di procedere quando si deve realizzare e presentare un best-fit di dati sperimentali. Le osservazioni che farò sono molto banali, però, specie considerando la (cattiva) abitudine di usare degli *scripts* più o meno rimanipolati, che precludono un'attiva interazione con il software, vale la pena di sottolinearle. Naturalmente, quando opportuno farò riferimento al software che impiegherete, cioè Python con il necessario armamentario di librerie, cercando quanto più possibile di seguire le indicazioni di Luca Baldini e Carmelo Sgrò, in modo che possiate ritrovarvi facilmente.

I principali consigli pratici che mi vengono in mente sono i seguenti.

- Per prima cosa, conviene *sempre* partire dal grafico, cioè costruire un grafico con i *soli* dati sperimentali. Normalmente i dati sperimentali sono

raccolti, assieme alle incertezze, in un file di testo opportunamente formattato (il cancelletto serve per commenti, il tab si usa come spaziatore tra diverse colonne, etc.). È possibile fare errori nella trascrizione dei dati sul file, per cui conviene sempre guardare l'"aspetto" dei dati (con un grafico) prima di lanciare l'algoritmo di best-fit.

- Osservazione davvero ovvia, ma molto importante: il grafico deve essere costruito in modo decente. Questo richiede che nome degli assi, unità di misura, scale, etc. siano correttamente scelti, specificati e configurati. Inoltre, a prescindere dal fatto che le usiate o meno per la procedura di best-fit, le barre di errore *devono sempre comparire tutte* nel grafico. Se avete delle incertezze di misura per la variabile dipendente, queste devono essere *sempre* riportate!

- La procedura numerica di best-fit richiede di stabilire dei *valori iniziali* dei parametri della funzione di fit da cui l'algoritmo possa partire nel suo processo iterativo di minimizzazione. Sapete che la corretta convergenza dell'algoritmo dipende dalla bontà con cui sono scelti tali valori iniziali. Quasi sempre è possibile, basandosi su modelli e considerazioni di ragionevolezza, stabilire dei valori iniziali opportuni, simili a quelli che vi aspettate abbiano i parametri determinati dal best-fit. Spendete il tempo necessario per valutare bene i valori iniziali. La loro scelta, infatti, può determinare il risultato del best-fit, come è comprensibile considerando ad esempio che l'algoritmo di minimizzazione è in grado di individuare solo *minimi locali* su un intervallo necessariamente limitato di variazione dei parametri.

- Qualora abbiate dei dubbi sui valori iniziali, il consiglio è quello di confrontare preliminarmente il grafico dei dati sperimentali con quello della funzione di fit calcolata per la vostra scelta ipotetica dei valori iniziali. La curva della funzione di fit dovrebbe essere "non troppo" differente dai dati sperimentali, o almeno dovrebbe riprodurne l'andamento generale. Questo è piuttosto semplice da fare e può essere molto produttivo per evitare di prendere cantonate.

- Purtroppo l'algoritmo di Python "agisce in silenzio", cioè di default non riporta sulla console nulla che possa indicare se l'algoritmo stesso è arrivato correttamente a convergenza, né dà indicazioni sul valore attribuito ai parametri di best-fit (quelli della funzione risultante dalla procedura di best-fit). Il valore dei parametri corredato dalla stima dell'incertezza e il valore del χ^2 risultante devono essere "estratti" dal software provvedendo opportuni, e non sempre chiarissimi, comandi nello script.

- Come avete già capito, determinare i parametri di best-fit e indicare il χ^2 ottenuto sono *parte integrante e imprescindibile* della procedura. I valori che ottenete vanno *sempre* riportati nelle schede di laboratorio. Tra l'altro, essi sono indispensabili per valutare la bontà del best-fit, che non può essere stabilita solo a occhio, guardando come la curva di best-fit si sovrappone ai dati.

III. L'ESEMPIO

L'esperienza presa ad esempio è semplicissima e abbastanza adatta, pur se non perfetta, per gli scopi di rinfrescare le vostre conoscenze su come fare e presentare un best-fit con Python. Come potrete accorgervi più avanti, in alcuni casi quello che faremo ha motivazioni un po' fittizie e artificiose, però ugualmente l'esempio è utile dal punto di vista didattico.

Nell'esperienza si ha un generatore di d.d.p. *reale* (l'alimentatore che avete già trovato sul banco) che produce una d.d.p. V_0 (misurata a circuito aperto). Quindi si collegano tanti resistori, di resistenza R_j (alcune delle resistenze sul banco) e si misura con il multimetro digitale configurato da amperometro l'intensità I_j della corrente che fluisce nel resistore collegato al circuito.

Come sapete, la legge di Ohm applicata al circuito stabilisce

$$I_j = \frac{V_0 - \Delta V_{strum}}{R_j + r_G} \ , \tag{4}$$

dove r_G rappresenta la resistenza interna del generatore (incognita!) e ΔV_{strum} la caduta di potenziale prodotta dall'inserzione del multimetro configurato come amperometro nel circuito. Il valore nominale dichiarato dal costruttore è, per il multimetro digitale, $\Delta V_{strum} = 200$ mV *quando la grandezza misurata arriva al fondo scala della portata prescelta.* Di conseguenza, tale caduta di potenziale dipende dalla misura che state eseguendo, cioè, nell'esperimento, dalla scelta di R_j. Pertanto ΔV_{strum} *non* è costante (se lo fosse, l'effetto visibile sarebbe pressoché solo quello di shiftare la curva che rappresenta la funzione).

Una forma alternativa dell'Eq. 4 può essere scritta esplicitando il valore R_{strum} della resistenza interna dello strumento, che è invece costante *per un dato fondo scala*:

$$I_j = \frac{V_0}{R_j + r_G + R_{strum}} = \frac{V_0}{R_j + r_{int}}; , \tag{5}$$

dove r_{int} rappresenta la somma delle resistenze interne considerate nel modello.

Se volete, possiamo dare una finalità all'esperienza: in sostanza, essa prevede di verificare che il modello che abbiamo costruito, in cui abbiamo considerato le resistenze interne, è valido, o perlomeno preferibile a quello in cui le resistenze interne sono trascurate (legge di Ohm pura e cruda). Inoltre l'esperienza può essere considerata come un metodo alternativo per determinare quantitativamente (con errore e tutto) le resistenze interne r_{int}. Per quanto riguarda la resistenza interna del generatore,

essa può essere valutata facendo riferimento al modello di Thevenin. Come sapete, in questo ambito si richiede di applicare un carico resistivo noto al generatore reale, e quindi estrarre la resistenza interna r_G dalla misura della corrente che fluisce nel carico. Rispetto a questo approccio, la misura indiretta attraverso il best-fit (nel senso chiarito sopra) ha dei potenziali vantaggi. Infatti essa permette di combinare il risultato di diverse misure, eseguite su diverse R_j, per cui potrebbe condurre a una maggiore accuratezza di misura: vedremo nel seguito se questo è il caso.

Infine, potete anche provare a supporre R_{strum} nota sulla base del dato *nominale* dichiarato dal costruttore. Infatti nel corso dell'esperimento sapete quale fondo scala state utilizzando per ciascuna delle misure fatte e quindi potete stabilire qual è il valore di R_{strum}. Questo valore può essere sommato a quello di R_j, in modo da lasciare come unico parametro incognito r_G.

Questa scelta è corretta, tuttavia essa non viene consigliata perché, in buona sostanza, non potete attribuire un'incertezza al valore di R_{strum}, che ritenete noto a partire da dati nominali. Così facendo rischiate di sottostimare l'incertezza nel valore sommato $R_j + R_{strum}$, cosa che non è (quasi) mai da farsi.

Possiamo in ogni caso affermare di avere individuato due finalità che corrispondono ai requisiti della procedura di best-fit come li abbiamo elencati prima: (i) potremo usare due modelli, con e senza resistenze interne, e confrontare i relativi best-fit, cioè i relativi valori del χ^2 ottenuto, oppure giocare con diverse espressioni della funzione di fit e vedere quale funziona meglio; (ii) potremo ottenere informazioni quantitative sul parametro di fit r_{int}, da confrontare con le aspettative (ovvero, per esempio, con la misura diretta fatta con il metodo di Thevenin). Per questioni di chiarezza, nello svolgimento dell'esempio non seguirò precisamente questo ordine, ma alla fine vedrete che tutto sarà consistentemente illustrato.

A. I dati e il loro grafico

Nell'esperimento ho iniziato misurando la d.d.p. prodotta dal generatore a circuito aperto, cioè collegando il multimetro digitale configurato da voltmetro che, grazie all'elevata resistenza interna, consente di trascurare la corrente che fluisce nello strumento. Ho trovato $V_0 = (4.94 \pm 0.02)$ V (ho supposto un'incertezza strumentale $\pm 0.5\%$ e verificato che questa è maggiore di ± 1 digit). Quindi ho inizialmente misurato i valori delle resistenze R_j con il multimetro digitale configurato come ohmetro e mi sono segnato i risultati assieme alla loro incertezza. Infine ho collegato uno per volta i resistori al generatore, in qualche caso usando combinazioni di resistori in serie e parallelo, interponendo il multimetro digitale configurato come amperometro, e ho misurato l'intensità di corrente I_j, valutandone l'errore con l'incertezza strumentale. Ho infine costruito con un text editor un file di testo con quattro colonne che riportano, nell'ordine, $R_j, I_j, \Delta R_j, \Delta I_j$.

Dato che è *sempre* poco consigliabile dare in pasto a un algoritmo valori molto piccoli o molto grandi, che è invece la situazione che si verifica in alcuni casi, ho deciso di *riportare le resistenze e le intensità di corrente* (e le loro incertezze) su scale di misura più confortevoli. Ho allora scelto di esprimere le resistenze in kohm e le intensità di corrente in mA. Inoltre nella scelta dei resistori, sapendo cosa volevo ottenere, ho privilegiato quelli di valore più basso (nella speranza di non bruciare il fusibile dell'alimentatore ho cercato di essere rapido nel fare le misure!), lasciando perdere quelli di valore molto elevato.

Per prima cosa ho graficato i dati usando questo script, che non metto in rete perché poco utile:

```
# the following lines are needed to import the content of different
# software packets (indeed, only the first one is strictly needed for the
# purpose of the present script, the following ones will be needed in the
# next implementations)
import pylab
import numpy
from scipy.optimize import curve_fit

# this is the command which extract the data from the columns in the text file
# the "unpack" instruction is required, don't forget it
# note that the file is stored in some directory of my own computer
R,I,dR,dI=pylab.loadtxt('D:/blahblah/data.txt',unpack=True)

# lines needed to increase the font size (to improve readibility), to put the
# correct labels on the axes, to make minor ticks to appear, respectively
pylab.rc('font',size=16)
pylab.xlabel('R  [kohm]')
pylab.ylabel('I  [mA]')
pylab.minorticks_on()
```

```
# modify the axis range to improve readibility
pylab.xlim(-10,350)
pylab.ylim(-10,130)

# draw the plot with errorbars: NOTE THE UGLY SYNTAX FOR INDICATING ERRORS
# IN THE AXES (THE VERTICAL ERROR BAR COLUMN COMES BEFORE THE HORIZONTAL ONE!)
# the linestyle command says that no line is needed, the marker is a circle,
# the color is black
pylab.errorbar(R,I,dI,dR,linestyle = '', color = 'black', marker = 'o')

# save the plot as a pdf file somewhere (in my own directories!)
pylab.savefig('D:/blahblah/fig1a.pdf')

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

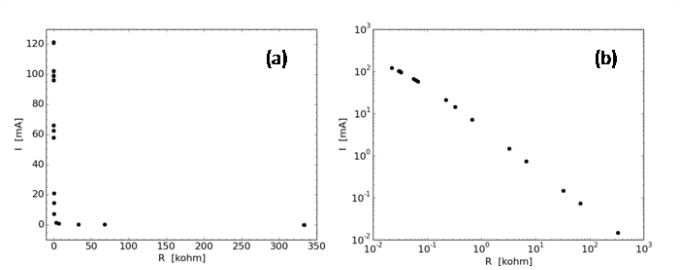

Figura 1. Grafico dei dati sperimentali in rappresentazione lineare (a) e bilogaritmica (b).

L'output grafico dello script è mostrato in Fig. 1(a). Lo script è commentato (in inglese maccheronico), tuttavia sottolineo di nuovo la strana sintassi necessaria per produrre le barre di errore lungo i due assi, che ci sono anche se si vedono poco a causa del basso valore assoluto dell'incertezza strumentale. La successione di argomenti è infatti: valore del dato sull'asse orizzontale, valore su quello verticale, errore sull'asse verticale, errore sull'asse orizzontale.

Il grafico prodotto è decisamente molto poco leggibile e la presentazione deve essere migliorata. Il motivo è la presenza di dati che coprono diversi ordini di grandezza della misura. In queste condizioni è sicuramente preferibile usare la *rappresentazione bilogaritmica*, che si ottiene premettendo alla produzione del plot le istruzioni `pylab.xscale('log');pylab.yscale('log')` (il punto e virgola fa da separatore per i comandi, che così possono andare su un'unica linea). In queste condizioni scegliere a mano il range degli assi non è più utile e ci si può servire della scelta (automatica) di default. Ugualmente è superfluo richiedere il disegno delle minor ticks, che è automatico quando la scala è logaritmica. Il risultato, molto più carino, è mostrato in Fig. 1(b) e lo script è disponibile nella mia pagina web con il nome `fit_ohm_display_log.py`.

B. Funzione di fit e valori iniziali

A questo punto dobbiamo scegliere il modello che vogliamo usare per il best-fit dei dati sperimentali. Come già affermato, è in linea di principio possibile usare diversi modelli (con o senza resistenze interne, con la caduta di potenziale o la resistenza interna dell'amperometro). Per cominciare, decido di usare la funzione di Eq. 5 e di porre la *sola* resistenza interna r_{int} (in realtà, ricordo, somma di due resistenze interne, una delle quali presumibilmente non costante a causa dell'uso di diverse scale di misura) come *parametro del fit*. In altre parole, il fit ha *un solo parametro libero*. Sottolineo che lasciare un solo parametro libero in un fit può non essere la scelta più corretta e noto come questo implichi di porre come parametro fisso V_0 (che scelgo pari al valore misurato, ovviamente *senza incertezza*). Tuttavia questa scelta è utile per fini didattici, dato che in questo modo la procedura di fit è tecnicamente più semplice.

Come prima cosa decido i valori iniziali (*il* valore iniziale) ponendo $r_{int} = 0.1$ kohm (ricordate che le resistenze vanno espresse in kohm a causa della scelta delle unità di misura fatta prima!). Questa scelta del valore iniziale non ha profonde motivazioni. Vedremo nell'esperienza che la resistenza interna dell'alimentatore, modellato secondo Thévenin, è di poche decine di ohm, essendo per altro quasi completamente dominata dalla resistenza del fusibile montato in serie all'alimentatore stesso. Infatti, come esperienza e Diego insegnano, montando un fusibile con corrente di fusione più alta, realizzato con un filo più spesso e quindi con resistenza più bassa, la resistenza di Thévenin si abbassa sensibilmente.

Stabilito tale valore indicativo per r_{int}, preparo uno script che contiene la definizione della funzione e ne fa il grafico sovrapposto a quello dei dati sperimentali.

Lo script, cha anche stavolta non posto nella mia pagina web perché poco rilevante, è questo:

```
import pylab
import numpy
from scipy.optimize import curve_fit

# data load
R,I,dR,dI=pylab.loadtxt('D:/blahblah/data.txt',unpack=True)

# set the constant parameter(s)
V0 = 4.94
rint = 0.1

# define the fit function (NOTE THE INDENT!)
def fit_function(R, rint):
    return  (V0/(R+rint))

# bellurie
pylab.rc('font',size=16)
pylab.xlabel('R  [kohm]')
pylab.ylabel('I  [mA]')
pylab.xscale('log'); pylab.yscale('log')

# prepare the independent variable values where the fit function
# has to be evaluated (a 100 point logarithmic equispaced array
# from 10^-2 to 10^3 kohm)
func_grid = numpy.logspace(-2, 3, 100)

# plot the function with a continuous black line
# NOTE THE ARGUMENTS OF fit_function!
pylab.plot(func_grid, fit_function(func_grid, rint), color = 'black')

# put a legend to show the used rint value (fontsize = 12)
# NOTE THE WAY THE VALUE OF rint IS AUTOMATICALLY INCLUDED IN THE TEXT
pylab.legend([('rint = ' + str(rint) +' kohm')], prop={'size':12})

# data plot (NOTE THE MARKER IS A DOT)
pylab.errorbar(R,I,dI,dR,linestyle = '', color = 'black', marker = '.')

# save the graph somewhere
pylab.savefig('D:/blahblah/fig2a.png')

# show on screen
pylab.show()
```

Lo script è commentato (ho accorciato i commenti delle parti già descritte nello script precedente), però qualcosa la voglio sottolineare. Notate come in esso sia stata definita una funzione (chiamata `fit_function`) che ha due argomenti, rispettivamente la variabile indipendente R e r_{int} (chiamato qui `rint` per brevità), mentre V_0 (qui `V0`) viene considerato una costante. Questo non è indispensabile, ma è utile in previsione degli sviluppi futuri. Inoltre osservate come, per graficare la funzione, ho creato un array (o lista, in realtà) di punti per la variabile dipendente: ho scelto un array di 100 punti equispaziati logaritmicamente nell'intervallo $(10^{-2} - 10^3$ (e sono kohm, se vogliamo dare un'unità di misura). Infine osservate come, per il gusto di cambiare, abbia usato un marker a punto, invece che a cerchio, per i dati sperimentali: questo permette di vedere un po' meglio le barre di errore. Tutto il resto sono bellurie di vario genere.

Il risultato è mostrato in Fig. 2(a): si vede come ci sia già un discreto accordo con i dati sperimentali, per cui le condizioni iniziali possono essere considerate ragionevoli per avviare la procedura di best-fit. Ora la loro scelta è stata fatta sulla base della ragionevolezza, o dell'esperienza. Sicuramente quello che sto per fare non ha basi di ragionevolezza, ma può comunque essere utile per capire il ruolo che le condizioni iniziali hanno nel best-fit. Impazzendo, avrei potuto porre come con-

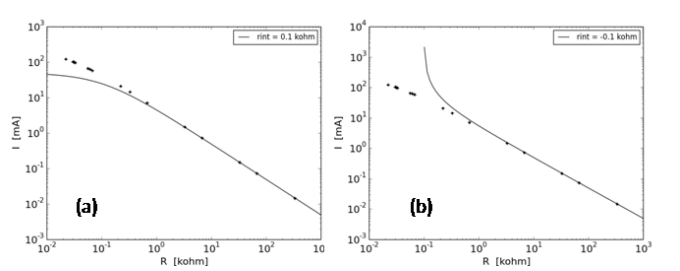

Figura 2. Grafico dei dati sperimentali con sovrapposti le funzioni calcolate per le scelte dei valori iniziali $r_{int} = 0.1$ kohm (a) e $r_{int} = -0.1$ kohm (b).

dizioni iniziale $r_{int} = -0.1$ kohm (il software non sa che le resistenze sono sempre positive, e io avrei potuto dimenticarlo!). Avrei ottenuto il grafico di Fig. 2(b), che mostra chiaramente disaccordo (c'è una divergenza nella funzione!). Usando questi valori iniziali ci sarebbero potuti essere problemi nella convergenza dell'algoritmo di minimizzazione.

Ora, a commento finale di questa sezione, posso tranquillamente affermare che la procedura di graficare i dati e la funzione calcolata con i valori iniziali è molto probabilmente inutile nell'esempio che stiamo trattando e potete sicuramente saltarla nella pratica. Tuttavia averla presentata per esteso mi fa sentire a posto con la coscienza, visto che in futuro potrebbero sicuramente presentarsi situazioni in cui la scelta dei valori iniziali è critica.

C. Il best-fit

Finalmente possiamo passare alla vera e propria esecuzione del best-fit. Per farlo, dobbiamo attenerci strettamente alle istruzioni del software, che in prima battuta suonano parecchio poco chiare. Almeno per questo primo esempio, manterrò per quanto possibile lo stile a cui siete abituati (modo elegante per dire che copio parte dello script da Baldini e Sgrò!). Tra le altre operazioni e comandi che bisogna includere nello script occorre:

- creare un array, o lista (qui chiamato `initial_values`) che contiene i valori iniziali dei parametri di fit; nel caso che stiamo esaminando, l'array ha lunghezza unitaria, contenendo solamente il valore iniziale dell'unico parametro r_{int};
- richiamare la procedura, o *routine*, di minimizzazione attraverso il comando `curve_fit` (affinché esso sia riconosciuto occorre premettere l'istruzione `from scipy.optimize import curve_fit` che abbiamo messo in testa a tutti gli script di questa nota). Gli argomenti della chiamata, cioè gli oggetti che si trovano fra parentesi dopo il comando di chiamata della routine, comprendono nell'ordine: il nome della funzione (`fit_function`), la variabile indipendente (`R`), i dati (`I`), i valori iniziali (`initial_values`), la "deviazione standard" del χ^2, cioè, per noi in questo esempio, l'incertezza ΔI_i (`dI`).
- La routine restituisce due array, che in questo esempio si chiamano `pars` e `covm`. La sintassi prevede di porre i nomi dei due array davanti al comando di chiamata, separati da una virgola e seguiti da un "=". La natura di questi array, come sarà chiaro in seguito, è diversa: il primo è un vettore (array unidimensionale), il secondo è una matrice (array bidimensionale), ma in questo primo esempio, dove operiamo con un solo parametro, possiamo disinteressarci di ciò.

In buona sostanza, nello script dopo la definizione della funzione vanno inserite queste istruzioni:

```
# set the array of initial value(s)
initial_values = (0.1)

# call the minimization procedure (NOTE THE ARGUMENTS)
pars, covm = curve_fit(fit_function, R, I, initial_values, dI)
```

Come già affermato, i risultati dell'algoritmo sono riportati negli array qui chiamati `pars` e `covm`. Il primo array contiene i valori dei parametri ottenuti dalla procedura di best-fit, il secondo riporta la cosiddetta *matrice di covarianza* di cui tratteremo in seguito un po' più nel dettaglio. Per ora ci basti sapere che gli elementi diagonali (solo uno, nel caso di singolo parametro!) di questa matrice rappresentano la *varianza* da attribuire alla valutazione tramite best-fit dei parametri. Più nello specifico, la radice quadrata di tale varianza corrisponde all'incertezza sui parametri determinati dal best-fit *entro una deviazione standard*, che è il modo convenzionale per definire l'errore sulla misura indiretta dei parametri eseguita tramite best-fit. Non voglio ora approfondire questo aspetto, ma sottolineo che tutto questo, a rigore, è valido solo se sono verificate le condizioni che abbiamo posto a monte dell'uso del best-fit, in particolare quelle che ci hanno permesso di usare Δy_i nella definizione del χ^2 [Eq. 2].

Queste informazioni, che fanno parte costitutiva della procedura di best-fit, vanno estratte e debitamente annotate, per esempio nelle schede di Laboratorio. Affinché compaiano come messaggio sulla console di Python è necessario aggiungere delle istruzioni specifiche, che purtroppo andranno riscritte quando dovrete modificare il best-fit (per tenere conto di altri parametri o di altre funzioni). Inoltre occorre calcolare il valore del χ^2 ottenuto al termine della procedura di minimizzazione, che è bene venga mostrato assieme al numero di gradi di libertà del best-fit (ndof, cioè number of degrees of freedom), che, come sapete, è definito come differenza tra il numero di dati considerati e quello dei parametri della funzione di fit.

A tutto questo provvedono le seguenti istruzioni, che, pur essendo abbastanza criptiche, non commento in dettaglio:

```
# extract the estimated standard deviation of the parameter(s)
delta_rint = numpy.sqrt(covm.diagonal())

# calculate the resulting chisquare and the number of dof
chisq = (((I - fit_function(R, pars))/dI)**2).sum()
ndof = len(R) - len(pars)

# print the results on the console (with bellurie)
print('r_int = %f +- %f' % (pars, delta_rint))
print('Chisquare/ndof = %f/%d' % (chisq, ndof))
```

L'intero script lo trovate nella mia pagina web sotto il nome `fit_ohm_one_parameter.py`. L'uscita che si legge in console è costituita per i miei dati dai risultati:

$$r_{int} = (0.018 \pm 0.0002) \text{ ohm} \quad (6)$$
$$\chi^2 = 9.9 \quad (7)$$
$$\text{ndof} = 14 \,, \quad (8)$$

dove ho usato un numero congruo di cifre significative (per favore, non riportate cifre poco significative nelle vostre schede!). Un test preliminare di affidabilità della procedura di best-fit consiste nel valutare il χ^2 ridotto (χ^2/ndof), che è minore di uno come ci piace che sia. Per vostrà curiosità, consultando le "tabelle del chi-quadro" si vede che la significatività del best-fit è attorno a 0.75, cioè ripetendo tante volte la procedura e/o le misure, nel 75% circa dei casi troverei valori "peggiori". Sulla base di quanto abbiamo stabilito all'inizio, questa affermazione è sicuramente da prendere con le molle, dato che non abbiamo garanzie sul corretto siginificato statistico delle nostre misure (per esempio, l'incertezza dei dati è strumentale!).

Poi possiamo verificare che il valore ottenuto per r_{int} è effettivamente "ragionevole". A questo proposito ricordo che, nel modello, $r_{int} = r_G + R_{strum}$, dove la resistenza interna dell'amperometro dipende dalla scala impiegata. Per le misure a più bassa resistenza, che sono le più "interessanti" per l'esperienza svolta, sono stati selezionati i fondo scala di 200 mA e 20 mA (scelti in modo da massimizzare il numero di cifre significative della misura). A tali scale corrispondono resistenze interne dello strumento rispettivamente di 1 ohm e 10 ohm. La resistenza interna del generatore può essere misurata con il metodo di Thèvenin. Io l'ho fatto per il mio esperimento, ottenendo $r_{G,Th} = (12 \pm 3)$ ohm. Tutto sommato, il valore ottenuto dal best-fit (effettivamente molto più accurato di quello tirato fuori con il metodo di Thevenin, stando all'incertezza ottenuta) mi sembra ragionevole. Infine il grafico con i dati e la funzione di fit sovrapposta è mostrato in Fig. 3, e l'accordo, a occhio, è proprio niente male.

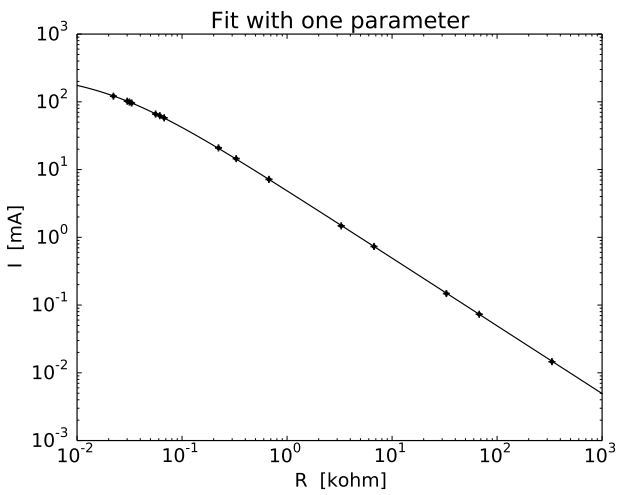

Figura 3. Grafico dei dati sperimentali con sovrapposta la funzione di fit determinata usando un solo parametro libero, come discusso nel testo.

D. Grafico dei residui

Qualche volta può essere utile verificare puntualmente lo scostamento tra la funzione di best-fit e i dati speri-

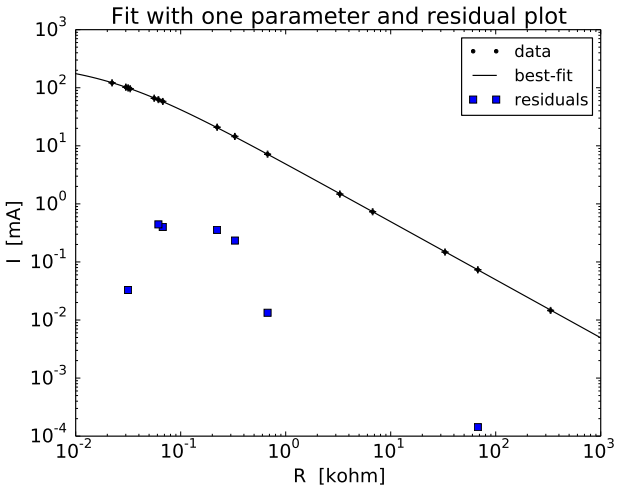

Figura 4. Grafico analogo al precedente con aggiunti i valori dei residui definiti nel testo.

mentali costruendo il cosiddetto *grafico dei residui*, che riporta per ogni valore misurato della variabile indipendente (la R_j) lo scarto tra previsione del best-fit e dato. Questo si può ottenere facilmente aggiungendo qualche istruzione allo script. Dato che qui non voglio entrare nella discussione del metodo dei subplots, che tratteremo eventualmente in altra sede, mi limito a sovrapporre il grafico dei residui a quello dei dati e del best-fit. Per farlo è sufficiente aggiungere allo script un'istruzione del tipo `pylab.plot(R,I-fit_function(R,pars),'bs')`: la trovate, assieme a qualche ulteriore e minimale belluria, nel file postato nella mia pagina web (di default, commento le linee che servono al plot dei residui).

In ogni caso, il risultato è mostrato in Fig. 4: si vede come la discrepanza maggiore in termini assoluti si abbia per i dati acquisiti a bassi valori di resistenze. Ora, questi dati sono quelli in cui il valore dell'intensità di corrente è maggiore e anche quelli in cui la deviazione rispetto alla legge di Ohm pura e cruda è più rilevante. Entrambe queste circostanze possono spiegare qualitativamente perché per essi i residui sono più grandi.

IV. FIT A DUE PARAMETRI

In linea di massima potremmo già essere più che soddisfatti di quanto trovato. Tuttavia, visto il carattere didattico di questa nota, proviamo a eseguire un fit con più di un solo parametro, nella speranza di migliorare ulteriormente l'accordo con i dati. Chiaramente i due parametri da considerare non potranno in alcun modo essere r_G e R_{strum} come compaiono in Eq. 5: queste due grandezze sono sommate l'un l'altra, e nessun metodo riuscirà mai a distinguere gli effetti della variazione dell'una o dell'altra. In altre parole, i due parametri r_G e R_{strum} sono *completamente correlati tra loro*.

Possiamo piuttosto considerare la funzione di Eq. 4 e tenere come parametri liberi di fit r_G e ΔV_{strum}. Dal punto di vista fisico la scelta *non* è molto giustificata. Sappiamo infatti che ΔV_{strum} *non* è costante nelle nostre misure, dato che dipende dalla lettura attuale dell'intensità di corrente I. Tuttavia introdurre questo nuovo parametro equivale di fatto a lasciare libero il valore di V_0 (V_0 e ΔV_{strum} sono sommati algebricamente al numeratore della funzione), che prima avevamo fissato senza tenere conto dell'incertezza di misura. E poi tutto questo serve per provare!

Lo script richiede più di qualche modifica rispetto a prima. In primo luogo va cambiata la definizione della funzione e provvisti i valori iniziali, che stavolta sono due e metto in forma di lista di Python (uso le parentesi quadre e cambio pure il nome, per economia, in `init`). Poi per passare i parametri alla routine di minimizzazione e per poterli alla fine estrarre uso i "puntatori" della lista, per esempio `pars[0]` e `pars[1]` che rappresentano i valori al primo e al secondo posto della lista `pars` (state attenti agli indici: la prima posizione è generalmente la 0). Quindi, dato che ormai siete grandi (non più al primo anno!) e in grado di capire quello che fate voi e che fa il software, uso una modalità semplificata per scrivere sulla console le uscite della procedura di best-fit.

Lo script, che potete trovare nella mia pagina web sotto il nome `fit_ohm_two_param.py`, è riportato qui di seguito *per le sole parti di interesse* (a questo punto avete capito bene cosa serve prima e dopo!):

```
# set the constant parameter(s)
V0 = 4.94

# define the fit function (NOTE THE INDENT!)
def fit_function(R, rG, DeltaV):
    return  ((V0-DeltaV)/(R+rG))

# set the array of initial value(s) (IN THE FORM OF A LIST)
initial_values = [0.1,0.2]

# call the minimization procedure (NOTE THE ARGUMENTS)
```

```
pars, covm = curve_fit(fit_function, R, I, initial_values, dI)

# calculate the resulting chisquare and the number of dof
chisq = (((I - fit_function(R, pars[0], pars[1]))/dI)**2).sum()
ndof = len(R) - len(pars)

# print the results on the console in an "expert" mode
print('Chisquare/ndof = %f/%d' % (chisq, ndof))
print('Fit parameters: ', pars)
print('Uncertainty on fit parameters: ', numpy.sqrt(covm.diagonal()))
print('Covariance matrix: ',covm)

# determine the "correlation" and print it on the console
# (NOTE THE SYNTAX)
print('Correlation: ',covm[0,1]/(numpy.sqrt(covm[0,0])*numpy.sqrt(covm[1,1])))
```

Un aspetto molto importante riguarda l'estrazione e la scrittura sulla console della *matrice di covarianza*, di cui parleremo in dettaglio nella sezione seguente. Qui mi limito a mostrarvi il grafico prodotto dallo script (Fig. 5), che continua a mostrare un accordo decisamente buono, e commentare sui risultati, che sono:

$$r_G = (0.018 \pm 0.0003)\ \text{ohm} \tag{9}$$

$$\Delta V_{strum} = (0.007 \pm 0.019)\ \text{V} \tag{10}$$

$$\chi^2 = 9.8 \tag{11}$$

$$\text{ndof} = 13\ . \tag{12}$$

Come si vede, la resistenza interna determinata risulta molto simile (identica dentro l'errore) a quella trovata nel fit a un singolo parametro. Infatti il parametro di fit restituito ΔV_{strum} è molto piccolo, cioè molto minore rispetto a V_0, per cui esso modifica in modo trascurabile l'esito del best-fit. Inoltre l'incertezza associata a questo parametro risulta grande rispetto al valore del parametro (è maggiore di questo), che può anche essere interpretato come indice di una scarsa "rilevanza" del parametro stesso nel modello. Infine, il χ^2 ridotto ottenuto è leggermente maggiore che nel fit a un singolo parametro, da cui la conclusione che, probabilmente, abbiamo fatto una fatica inutile nel provare il fit a due parametri (sicuramente tutt'altro che inutile dal punto di vista dell'esercizio!).

I risultati ottenuti non devono stupirvi troppo. Sappiamo infatti che il ruolo della resistenza interna dello strumento è limitato, essendo essa generalmente piccola rispetto alle altre resistenze in gioco. Inoltre sappiamo che il modello non è corretto, dato che supponiamo che la presenza dello strumento produca un effetto costante su tutte le misure, affermazione che sappiamo essere sbagliata.

In questa nota non voglio presentare un'ulteriore prova, concettualmente significativa ma praticamente poco interessante. Tale prova consiste nell'usare come modello la legge di Ohm pura e cruda, senza resistenze interne, lasciando come parametro libero del fit (è necessario averne almeno uno!) la d.d.p. V_0. Su questo tipo di analisi torneremo altrove in una nota che sarà resa disponibile in futuro. Tuttavia ho provato anche questa soluzione, ottenendo, come atteso, un pessimo accordo fra dati e best-fit. Il χ^2 trovato in questo caso è addirittura superiore a 1000!

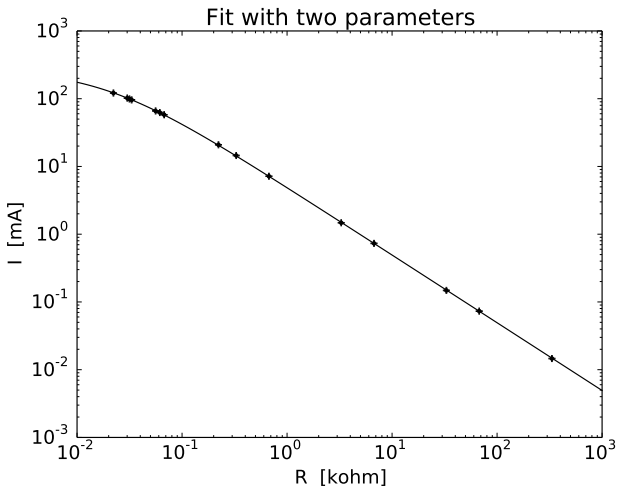

Figura 5. Grafico dei dati sperimentali con sovrapposta la funzione di fit determinata usando due parametri liberi, come discusso nel testo.

V. MATRICE DI COVARIANZA

Sicuramente l'esercizio che abbiamo svolto è molto importante per rendersi conto della presenza di una caratteristica del best-fit che, probabilmente, vi era ignota. Come già affermato, l'uscita della routine di minimizzazione consiste in un vettore di parametri e in una matrice, detta di covarianza. Questa matrice è sempre quadrata ed è di dimensioni pari al numero di parametri impiegati: dunque essa è senz'altro irrilevante quando il fit ha un solo parametro libero, ma molto importante quando i parametri sono più di uno.

Per mantenere un approccio pratico al problema, evitiamo considerazioni matematiche e limitiamoci a osservare che:

- gli elementi diagonali rappresentano la varianza dei parametri valutati tramite best-fit. Infatti l'istruzione un po' criptica `numpy.sqrt(covm.diagonal())` che trovate negli script quando si desidera visualizzare l'incertezza dei parametri indica proprio un'operazione consistente nell'estrarre gli elementi diagonali della matrice e di farne la radice quadrata, che è appunto la deviazione standard (l'incertezza) sul valore dei parametri di fit.

- Gli elementi fuori diagonale (la matrice è simmetrica, dunque per due parametri si ha in realtà un solo elemento fuori diagonale) rappresentano la covarianza non normalizzata dei parametri.

Il nome di questa grandezza contiene il suo significato. La covarianza (fra, ad esempio, due parametri di best-fit) dà una misura di quanto la variazione di un parametro può influire sull'altro. Quindi essa permette di stabilire, almeno qualitativamente, quanto i parametri siano "indipendenti" o "dipendenti" tra loro.

Nell'esempio trattato nella sezione precedente, l'elemento fuori diagonale della matrice di covarianza risultava $\approx -4 \times 10^{-6}$. Il segno negativo indica, in pratica, che se uno dei due parametri fosse stato aumentato l'altro avrebbe dovuto diminuire. Se ricordiamo come è scritta la funzione, questo è ovvio: uno si trova al numeratore, l'altro al denominatore di una frazione (il segno negativo che nel modello abbiamo posto davanti a ΔV_{strum} non conta). Il valore determinato numericamente non ha un diretto significato. Sicuramente, però, maggiore esso è e più grande è il grado di "correlazione" tra i parametri.

Normalmente una qualche indicazione quantitativa si può ottenere usando il valore *normalizzato* della covarianza, che qui chiamiamo anche *correlazione*. La normalizzazione si fa dividendo per il prodotto delle deviazioni standard (le incertezze) sui parametri (questa opzione è integrata nello script mostrato in precedenza, dove le linee corrispondenti sono commentate). Nel caso dell'esempio, la normalizzazione produce una correlazione ≈ -0.7: un valore unitario rappresenta una completa correlazione, il valore ottenuto rappresenta un elevato grado di correlazione.

L'analisi della correlazione ha diverse conseguenze pratiche. Per il momento limitiamoci a osservare che, se più parametri sono *molto* correlati tra loro (per esempio, correlazione maggiore di 0.7-0.8, in valore assoluto), il modello prescelto potrebbe essere non molto significativo, poiché il ruolo dei vari parametri non può essere considerato separatamente. Inoltre le incertezze sui parametri di best-fit aumentano all'aumentare della correlazione, come abbiamo verificato anche nel nostro esempio. Infatti per parametri correlati si ha che l'incertezza nella determinazione di uno si riflette (si propaga) nell'incertezza sulla determinazione dell'altro, o degli altri. Avremo probabilmente modo di tornare su questo aspetto eseguendo altri best-fit in futuro.

VI. INCERTEZZE SULLA VARIABILE INDIPENDENTE

Torniamo in questa sezione sulla questione dell'incertezza sulla variabile indipendente (Δx_i, ovvero ΔR_j nel contesto del nostro esempio). Nell'esperimento le resistenze sono state misurate con il multimetro digitale, dunque esse hanno un'incertezza che non è trascurabile, ma anzi è paragonabile o addirittura superiore, in termini relativi, a quella sulla misura delle intensità di corrente. Di conseguenza, aver considerato nel best-fit solo le incertezze Δy_i (ΔI_j, nel contesto dell'esempio) è a rigore non giustificato e i risultati ottenuti potrebbero essere non del tutto corretti, specie per la valutazione del χ^2 che potrebbe risultare sovrastimato.

Abbiamo già stabilito un modo possibile (non bello, non sicuro!) per tenere conto dell'incertezza sulla variabile indipendente. Esso si basa sulla propagazione dell'errore, che qui applichiamo usando le regoline a voi note. Per semplicità, e per evitare accanimenti matematici, facciamo riferimento al modello che porta a un fit a un solo parametro, r_{int}. Possiamo facilmente determinare che il contributo dell'incertezza ΔR sulla misura di I è atteso essere pari a $V_0 \Delta R/(R + r_{int})$. Questo contributo va aggiunto in quadratura all'incertezza ΔI. In altre parole, il termine che sta a dividere i residui quadrati in Eq. 2 diventa

$$\sqrt{(\Delta I)^2 + (\frac{V_0}{(R + r_{int})^2} \Delta R)^2} \, . \qquad (13)$$

Riprendiamo lo script usato per il fit a un solo parametro e infiliamoci questa nuova definizione di incertezza "complessiva", che dovrà essere passata alla routine di minimizzazione. A questo scopo, modifico la chiamata in `pars, covm = curve_fit(fit_function, R, I, initial_values, numpy.sqrt(dI**2+((V0/(R+initial_values)**2)*dR)**2))`, dove dovete porre attenzione a come l'incertezza, il quinto argomento della chiamata, è stata definita. Come fatto notare prima, introducendo ΔR in questa forma si è costretti a fare delle assunzioni a priori sui valori dei parametri (del parametro) di fit, che in questo caso è posto pari al valore iniziale. Inoltre va modificata anche la linea dello script dove viene calcolato il χ^2, che assume la forma seguente: `chisq = (((I - fit_function(R, pars))/(dI+(V0/(R+pars))*dR))**2).sum()`.

Non riporto lo script, né lo metto in rete. Vi dico, però, che l'ho fatto girare ottenendo questi risultati (il grafico non lo mostro, è praticamente indistinguibile da quello di Fig. 3):

$$r_{int} = (0.018 \pm 0.0002) \text{ ohm} \qquad (14)$$

$$\chi^2 = 8.0 \tag{15}$$
$$\text{ndof} = 14\ , \tag{16}$$

dove si nota che effettivamente il valore del χ^2 si è in piccola misura ridotto (il fit è più "significativo"), così come l'incertezza sul parametro r_{int}.

Nella prassi del best-fit è comune trascurare l'effetto delle incertezze sulla variabile indipendente, e probabilmente questo siete abituati a fare, e continuerete a fare, anche voi. Tuttavia, quando queste incertezze sono evidentemente non trascurabili, ovvero quando volete o dovete stimare in modo un po' più accurato i parametri di fit e la loro incertezza, ricordatevi di questa possibilità.