Esercizio di "simulazione" dell'esperienza sulla resistenza interna del generatore

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

(Dated: version 2 - FF, 25 ottobre 2014)

Questa nota riporta il testo e la soluzione di un esercizio che richiede l'uso del software che avete conosciuto a Laboratorio 1. Esercizio e soluzione sono un po' particolari, e rappresentano in maniera molto semplificata una situazione tipica nella fisica. Lo scopo di questa nota, che riporta una *possibile* soluzione, quella trovata da me, è di costituire una guida per quello che dovete, o dovreste, fare da soli.

I. INTRODUZIONE

L'esercizio che vogliamo svolgere è banalissimo. Abbiamo un semplicissimo circuito di cui possiamo ipotizzare un altrettanto semplicissimo modello. Questo modello contiene degli aspetti che in linea di principio sono incogniti e vogliamo vedere quanto e in quali condizioni questi aspetti influiscano sulla "risposta" del modello. Come succede molto spesso nella pratica di laboratorio, ci aspettiamo che la risposta sia rappresentabile con un grafico e ci chiediamo che forma avrà questo grafico, cioè che andamento avrà la funzione graficata, a seconda di uno o più parametri incogniti.

Useremo la legge di Ohm, che ha un carattere sicuramente deterministico, nel senso che il nostro modello dovrà fornire una risposta determinata a una situazione altrettanto determinata (quasi tutta la fisica generale è deterministica!). Inoltre la matematica che implementa il modello è analitica. Dunque la descrizione matematica del modello è accurata e il termine "simulazione" che ho usato nel titolo va sicuramente messo tra virgolette perché, in genere, quando si fa una "vera" simulazione si tiene conto di aspetti stocastici: non è questo il caso.

Per fare i grafici userò "Python", ovvero l'insieme di pacchetti in ambiente Python che avete imparato a conoscere nel corso di Laboratorio 1. Io Python non l'ho mai usato prima d'ora, quindi è sicuro che la mia soluzione presenti molti punti di miglioramento.

II. RESISTENZA INTERNA DEL GENERATORE DI D.D.P. E SUA MANIFESTAZIONE

Quella che "simuliamo" è un'esperienza del corso pratico, quella che consiste nel montare il circuito di Fig. 1 e misurare la corrente che vi fluisce in funzione della scelta del resistore R. Il generatore di d.d.p. che avete in laboratorio è ovviamente reale, di conseguenza assolve alla sua missione, quella di produrre una differenza di potenziale V_0, nei limiti delle sue possibilità. Infatti, per ragioni legate alla capacità di erogare potenza, non potete chiedere al generatore di mandare nel circuito una corrente di intensità arbitrariamente grande mantenendo inalterata la d.d.p. prodotta.

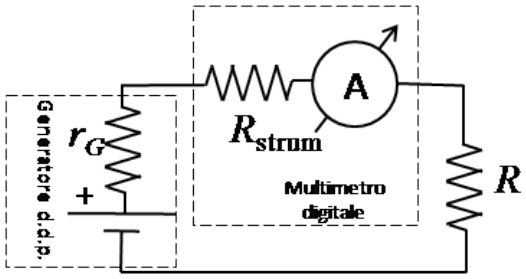

Figura 1. Circuito considerato per la "simulazione" discussa nel testo.

Come abbiamo discusso a lezione, un possibile modello di generatore di d.d.p. reale consiste nella serie di un generatore di d.d.p. *ideale* e di una resistenza interna r_G, come indicata in Fig. 1. A questo modello si dà talvolta il nome di *modello di Thevenin*. Il valore V_0 della d.d.p. erogata dal generatore ideale si può misurare in condizioni di *circuito aperto*, cioè supponendo un carico nullo, ovvero immaginando che il generatore sia chiamato a erogare una corrente *trascurabile*. Come sapete, questa situazione si realizza piuttosto bene, dal punto di vista sperimentale, collegando al generatore un voltmetro con elevatissima resistenza interna, come per esempio è il multimetro digitale (configurato come voltmetro). Dunque V_0 si può facilmente conoscere.

La resistenza interna, che si chiama anche *resistenza di Thevenin*, può essere determinata con uno specifico metodo sperimentale. Nell'esperimento che vogliamo "simulare" essa è considerata incognita, pur appartenendo a un intervallo che possiamo scegliere basandoci sulla ragionevolezza.

Per il momento, e fino a nuovo ordine, immaginiamo che l'amperometro montato nel circuito di Fig. 1 sia *ideale*, cioè che la sua resistenza interna R_{strum} sia del tutto trascurabile. In queste condizioni, la legge di Ohm ci dice che la corrente misurata ha intensità

$$I = \frac{V_0}{R + r_G} \,, \tag{1}$$

dove R è la resistenza del resistore che funge da *carico* (resistivo!) del generatore.

Nell'esperienza pratica R può essere variata discretamente in un vasto intervallo, dato che potete usare tutti

i resistori del banco. In questo esercizio vogliamo prevedere come si modifica la "forma" della *funzione* $I(R)$ a seconda del valore di r_G, partendo dal valore nullo che significa, in pratica, ipotizzare di avere un generatore di d.d.p. ideale.

Ora tutti sapete che la funzione ha il grafico di un ramo di iperbole. Le conoscenze di matematica che sicuramente avete vi permettono anche di stabilire *qualitativamente* come il grafico si modifichi per $r_G \neq 0$, in particolare di prevedere che la modifica sarà più marcata quando $r_G \gtrsim R$. Qualcuno potrebbe anche divertirsi a scrivere sviluppi della funzione di Eq. 1 e da lì dedurre un po' meglio gli andamenti richiesti.

Nella pratica, però, se l'obiettivo è quello di stabilire in modo un po' meno qualitativo la risposta del modello, cioè l'andamento previsto sulla base del modello al variare del parametro r_G, può essere molto utile *disegnare grafici* $I(R)$ per diversi r_G, cioè disegnare una famiglia di funzioni per scelte ragionevoli dei valori che entrano nella funzione stessa. Questo è il banalissimo scopo della nostra "simulazione".

Visto che sperimentalmente faremo variare R su un vasto range, comprendente diverse decadi, e visto che vogliamo evidenziare il meglio possibile le differenze nell'ambito della famiglia di curve che faremo disegnare al software, sceglieremo una *rappresentazione bilogaritmica* per il grafico $I(R)$. Sapete benissimo che, se $r_G = 0$, la rappresentazione bilogaritmica è una retta con coefficiente angolare -1 (nella scala opportuna). La deformazione della retta indicherà subito all'occhio la rilevanza dell'effetto $r_G \neq 0$ nel nostro modello.

Per usare valori realistici, porremo:

$$V_0 = 5 \text{ V} \tag{2}$$

$$R = 10 \text{ ohm} - 1 \text{ Mohm} \tag{3}$$

$$r_G = 0, 5, 10, 15, 20, 25, 30 \text{ ohm} , \tag{4}$$

dove la scelta dei valori di r_G è dettata dall'esperienza (mia, e presto vostra). Notate che, come in ogni "simulazione", i valori sono scelti con incertezza nulla, per cui le cifre significative non sono un problema. Di conseguenza i risultati "simulati" si assumono qui come privi di incertezza.

III. "SIMULAZIONE"

L'esercizio consiste nel produrre una famiglia di grafici $I(R)$ secondo Eq. 1 per diversi valori del parametro r_G. Le librerie `pylab` e `numpy` di Python si prestano benissimo allo scopo. Prima di vedere e commentare lo script utilizzato, faccio qualche premessa.

Un modo efficace per calcolare, e quindi graficare, una funzione nel software consiste nel far valutare la funzione stessa su un array di valori della variabile indipendente, che in questo caso è R. Dunque in prima battuta bisogna costruire un array di R. Visto che il tempo macchina è gratuito e che mi piacerebbe che la mia funzione fosse graficata come una linea continua, senza "seghettamenti", vorrei avere un array costituito da molti valori, qualche centinaio. Naturalmente questo non è quello che si verifica nell'esperimento, dove avete a disposizione un numero finito di R_j. Però in questa "simulazione" posso far finta di avere tantissimi resistori diversi.

Tenendo conto che il grafico avrà una rappresentazione bilogaritmica, e anche del fatto che nella realtà sperimentale R verrà fatto variare su tante decadi, sarà bene che l'array sia costituito da valori *logaritmicamente* equispaziati fra loro. In `numpy` tale array si può creare con il comando `numpy.logspace`, che ha diversi argomenti, alcuni opzionali. Occhio: a differenza del comando, a voi noto, `numpy.linspace`, in cui l'intervallo è delimitato dai valori che volete che l'array abbia come minimo e massimo, qui l'intervallo è delimitato dall'"ordine di grandezza" (a base 10) dei valori che volete far comparire nel vostro array. Se non è molto chiaro, lo capirete meglio guardando i commenti allo script.

Altra informazione tecnica, che credo già vi sia nota. Per scegliere la rappresentazione bilogaritmica occorre premettere al comando di plot le istruzioni `pylab.xscale(log)` e `pylab.yscale(log)` (evidentemente ognuna funziona su un asse). Tutto il resto sono bellurie che servono per rendere accettabile il grafico, e che già avete sicuramente visto lo scorso anno.

Allora, lo script che serve per creare una sola curva, per il valore specifico $r_G = 20$ ohm, è il seguente (trovate questo script sotto il nome `simul_res_int_unico.py` nella mia pagina web).

```
import pylab
import numpy

# constant values
V0=5
rG=20

# create the array R containing 500 values
# evenly spaced, in the logarithmic space,
# between 1e1 and 1e6
R = numpy.logspace(1,6,100)

# evaluate the function
I=V0/(R+rG)

# set the log log scale
pylab.xscale('log')
pylab.yscale('log')

# bellurie che dovete ricordare!
pylab.rc('font',size=16)
pylab.xlabel('R  [ohm]')
pylab.ylabel('I  [A]')

# draw the plot
pylab.plot(R,I,color='black')
```

```
# save the plot (in my own directory!)
pylab.savefig('D:\blahblah\plot3.pdf')

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

Il risultato è mostrato in Fig. 2: si vede chiaramente come l'andamento rettilineo atteso nel modello che trascura la resistenza interna del generatore è non riprodotto. La discrepanza è più marcata nell'estremo basso dei valori di R considerati, per cui d'ora in avanti ridurremo il range della rappresentazione, cosa che, come sapete, si ottiene con i comandi `pylab.xlim()` e `pylab.ylim()` per i due assi.

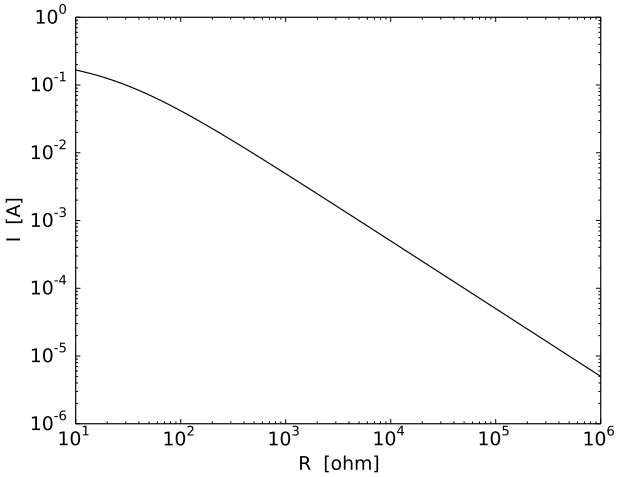

Figura 2. Output della "simulazione" costruita per il valore $r_G = 20$ ohm.

È chiaro che ora diventa interessante vedere come le varie curve cambino al variare di r_G. Dato che sono pigro e non ho avuto voglia di ripetere tante volte lo stesso script per ottenere grafici diversi, ho cercato un modo per avere tutti i grafici che mi servono con un solo script. Vista la mia ignoranza, non sono affatto sicuro che questo sia il modo migliore. Quello che ho fatto è stato inserire un ciclo di loop, per meglio dire un'istruzione `for`, all'interno dello script per generare i grafici che mi servivano e averli come curve sovrapposte, disegnate con diversi colori, sullo stesso plot. Nello script ci sono diverse cose interessanti. Quella che mi sembra più notevole è la semplicissima sintassi del ciclo `for`, che si ottiene scrivendo le istruzioni che si vogliono ripetute con un "indent" (una tabulazione) all'inizio. Ciò mi sembra geniale (anche se rischioso)! Inoltre ho faticato un po' per generare automaticamente la legenda del grafico, nella quale sono indicati i valori di r_G (qui scritta `rG`) corrispondenti alle varie curve, che si differenziano per il colore. Il risultato della fatica è riportato nello script assieme a pochi commenti. Provate a capire quello che ho fatto.

Lo script, che trovate nella mia pagina web con il nome `simul_res_int_multiplo.py`, è questo:

```
import pylab
import numpy

# constant values
V0=5


# create the array R containing 500 values
# evenly spaced, in the logarithmic space,
# between 1e1 and 1e6
R = numpy.logspace(1,6,100)

# initialize the rG and the counter values
rG = 0.
colorcounter = 0

# create the string array for the colors of the
# plotted lines
colore=['black','red','green','orange','blue','gray','vio

# initialize the string array needed for the legend
legenda = [('rG = 0')]

# realize the for cycle (NOTE THE INDENT OF THE
# SECTION TO BE REPEATED IN THE LOOP!)
for colorcounter in range(0,7):

    # evaluate the function
    I=V0/(R+rG)

    # set the log log scale
    pylab.xscale('log')
    pylab.yscale('log')

    # bellurie che dovete ricordare!
    pylab.rc('font',size=16)
    pylab.xlabel('R  [ohm]')
    pylab.ylabel('I  [A]')

    # draw the plot
    pylab.plot(R,I,color=colore[colorcounter])

    # modify the axes range to better show the behavior
    pylab.xlim(1e1, 1e3)
    pylab.ylim(3e-3,3e-1)

    # draw the legenda
    pylab.legend(legenda, prop={'size':12})

    # increment the rG value
    rG += 5.

    # update the legenda (append the current rG value)
    legenda.append('rG = ' + str(rG) + ' ohm')

# save the plot (in my own directory!)
```

```
# (NOTE THAT THERE IS NO INDENT SINCE THE FOR
# CYCLE IS NOW COMPLETED)
pylab.savefig('D:\blahblah\plot5.pdf')

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

Il risultato, che vi invito caldamente a ripetere e migliorare, è riportato in Fig. 3. Notate che, avendo scelto per i due assi lo stesso intervallo di variazione (3 decadi), la retta attesa nel caso di modello con resistenza interna nulla dovrebbe essere parallela alla "diagonale" del grafico. In effetti la curva corrispondente a $r_G = 0$ assomiglia a una retta con questa pendenza. Le altre, invece, assomigliano sempre meno a una retta all'aumentare di r_G. Vedrete poi nell'esperimento se questa simulazione è accurata.

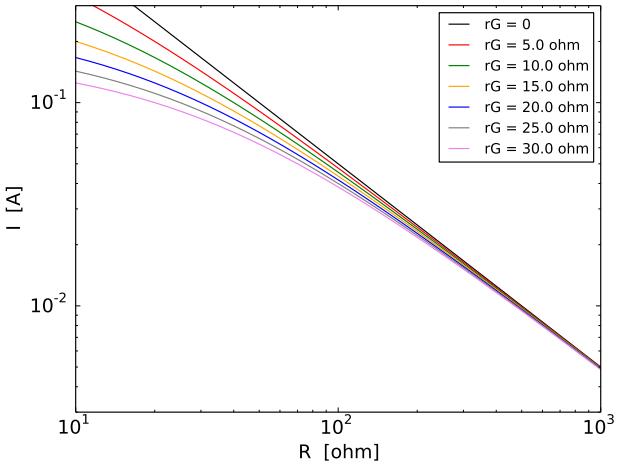

Figura 3. Output della "simulazione" costruita per diversi valori di r_G, come in legenda.

A. Resistenza interna dell'amperometro

Come sapete, in quanto strumento reale anche l'amperometro si discosta dal comportamento ideale, che prevederebbe una resistenza interna nulla. Esso infatti può essere immaginato come un amperometro ideale, con resistenza interna nulla, in serie a un resistore R_{strum}, come mostrato in Fig. 1. Ovviamente anche la presenza di questo elemento reale può condurre a differenze nel comportamento della curva $I(R)$.

Leggendo i manuali si vede come il costruttore del multimetro, invece che fornire il valore della resistenza interna dello strumento configurato come amperometro, abbia preferito dare esplicitamente il valore della "caduta di potenziale" dovuta all'inserzione dello strumento, che è indicata come $\Delta V_{strum} = 200$ mV, *indipendentemente dalla scala*. Ovviamente, se siete interessati a conoscere la resistenza interna, che è invece dipendente dalla scala, è sufficiente che applichiate la legge di Ohm ($R_{strum} = \Delta V_{strum}/I$).

Possiamo banalmente tenere conto di tale caduta di tensione sottraendola al termine V_0 contenuto in Eq. 1. Stavolta non sto a riportare lo script di Python, che ha solo piccole modifiche rispetto a quelli già presentati, ma mostro direttamente in Fig. 4 il risultato come confronto tra la situazione senza e con la correzione (in entrambi i casi, $r_G = 20$ ohm). Si vede come l'effetto sia quello di traslare, pressoché rigidamente, la curva. Dunque esso non comporta effetti eclatanti sulla "forma" del grafico.

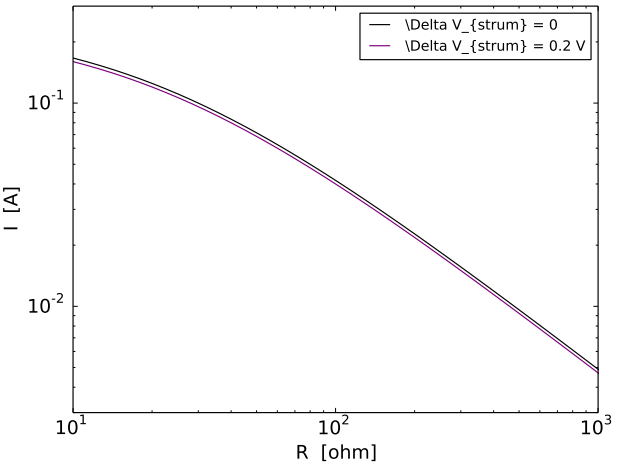

Figura 4. Output della "simulazione" costruita per il valore $r_G = 20$ ohm trascurando o considerando la caduta di potenziale prodotta dall'inserzione in serie dell'amperometro reale (vedi legenda).

IV. NON C'ENTRA NULLA QUI, MA LO METTO QUI!

Visto che ho Python aperto, aggiungo anche un'altra sorta di "simulazione" che non c'entra nulla con il resto. Il problema, qui, è quello di determinare la propagazione dell'errore (massimo) nella valutazione del *rapporto di partizione* in tensione visto nella prima esperienza pratica.

Non ripeto i dettagli sperimentali, che qui non sono rilevanti, ma mi concentro sul fatto che tale rapporto era stabilito valere $\alpha = R_1/(R_1 + R_2)$, con R_1, R_2 valori *misurati* di due resistenze. In quanto misurati, tali valori avevano incertezze, rispettivamente ΔR_1 e ΔR_2. Il problema è quello di determinare, attraverso le regole di propagazione dell'errore (massimo), l'incertezza $\Delta\alpha$.

Come sapete, questo si può ottenere usando la "regola delle derivate parziali":

$$\Delta\alpha = |\frac{\partial\alpha}{\partial R_1}|\Delta R_1 + |\frac{\partial\alpha}{\partial R_2}|\Delta R_2 \qquad (5)$$

$$= \frac{R_1\Delta R_2 + R_2\Delta R_1}{(R_1+R_2)^2} \ , \tag{6}$$

dove l'ultimo passaggio è dato a meno di possibili errori.

L'errore relativo (massimo) su α, che qui chiamo per comodità y, è allora

$$y = \frac{\Delta\alpha}{\alpha} = \frac{R_1\Delta R_2 + R_2\Delta R_1}{R_1(R_1+R_2)} \ , \tag{7}$$

come si può facilmente determinare.

Per motivi che vi saranno chiari più avanti, introduco la grandezza adimensionale $x = R_1/R_2$ e pongo $\Delta y_1 = \Delta R_1/R_1$, $\Delta y_2 = \Delta R_2/R_2$ (le ultime due grandezze sono le incertezze *relative* nella misura delle due resistenze). Salvo errori, trovo con un po' di algebra:

$$y = \frac{\Delta y_1 + \Delta y_2}{1+x} \ . \tag{8}$$

Supponiamo ora che le due resistenze vengano misurate con lo stesso strumento, come avete fatto voi con il multimetro digitale. Salvo casi abbastanza sporadici (pensate a quali possono essere tali casi, per esempio se le due resistenze hanno valori molto diversi), l'incertezza relativa delle due misure sarà la stessa, dato che essa sarà dominata dall'errore strumentale (prevalentemente di calibrazione), cioè $\Delta y_1 = \Delta y_2 = \Delta y$. In queste condizioni si ha

$$z = \frac{y}{\Delta y} = \frac{2}{1+x} \ , \tag{9}$$

valore che rappresenta di quanto aumenta, o diminuisce, l'errore relativo su α rispetto all'errore relativo Δy sulle misure originarie.

Ne ho fatto un grafico (stavolta in carta non logaritmica) usando questo semplicissimo script (lo trovate come `simul_err_part.py` nella mia pagina web):

```
import pylab
import numpy

# create a 500 element x array equispaced
x=numpy.linspace(.1,10,500)

z=2/(1+x)

# make minor ticks to appear on both axes
pylab.minorticks_on()

pylab.xlabel('x');pylab.ylabel('z');pylab.rc('font',size=16)
pylab.plot(x,z,color='black')

pylab.savefig('D:\blahblah\plot7.pdf')
pylab.show()
```

Il risultato, mostrato in Fig. 5, è parecchio interessante. Si vede ad esempio come z tenda a 2 per $x \to 0$. Poiché $x = R_1/R_2$, questo significa $R_2 >> R_1$, cioè una delle resistenze del partitore è molto più grande dell'altra; di conseguenza $\alpha \to R_1/R_2$. In queste condizioni l'errore relativo del rapporto tende alla somma degli errori relativi delle misure di partenza, cioè $z \to 2$ (abbiamo assunto che gli errori relativi delle misure di partenza fossero uguali tra loro). Dall'altra parte, z tende a 0 per $x \to \infty$, cioè per $R_2 << R_1$. In queste condizioni $\alpha \to 1$, per cui dal punto di vista matematico è comprensibile che $\Delta\alpha$ tenda a zero (non molto fisico, però...).

La situazione incontrata è tipica della propagazione degli errori applicata a espressioni in cui la stessa grandezza incerta compare a numeratore e denominatore (R_1, nel nostro caso). Si dice talvolta che c'è *correlazione* nell'incertezza al numeratore e al denominatore; se si trascura la correlazione, si sovrastima l'incertezza nel risultato. In ogni caso, spesso sovrastimare l'incertezza è lecito (non è lecito sottostimarla), per cui la stima grossolana $z \approx 2$, che a rigore è corretta solo se numeratore e denominatore hanno la stessa incertezza relativa, può essere accettabile, almeno in condizioni di partitori di tensione realistici.

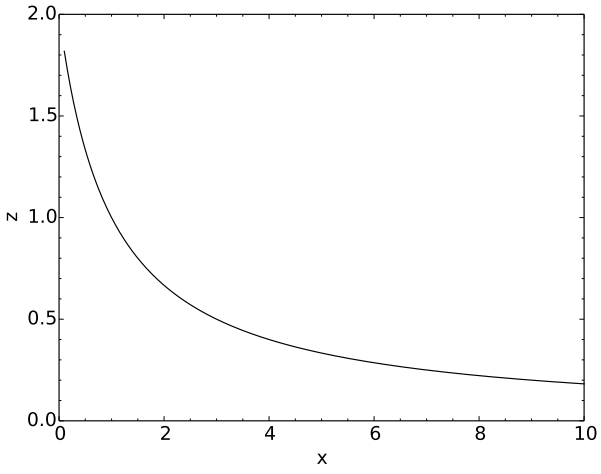

Figura 5. Grafico della funzione scritta in Eq. 9.