Introduzione ai circuiti e studio di filtri del primo ordine[a]

Francesco Polleri[1, b] e Mattia Sotgia[1, c]
(Gruppo A1)
[1] *Dipartimento di Fisica,*
Università degli Studi di Genova, I-16146 Genova,
Italia

(Dated: presa dati 19 ottobre 2021, analisi dati e relazione in data 27 ottobre 2021)

Vogliamo costruire e verificare il funzionamento di un filtro passa-basso (*low-pass filter*) o passa-alto (*high-pass filter*) sfruttando i principi fisici che sono dietro al comportamento di un circuito RC posto in tensione alternata.

I. MISURAZIONE DI *R*

Con il multimetro da banco impostato per misure di resistenza in corrente continua colleghiamo i capi dei connettori a "banana" al multimetro e alla base di lavoro, quello nero sul GND, quello rosso sul capo +Vcc. Mettiamo in serie ai pin corrispondenti al GND e a +Vcc la nostra resistenza ed effettuiamo così la misura del suo valore reale.

II. DESCRIZIONE APPARATO SPERIMENTALE

Realizziamo un circuito come quello presente in Figura 1. Utilizziamo l'oscilloscopio sia come generatore di segnale sinusoidale che come rivelatore. Utilizzando uno sdoppiatore BNC a T possiamo dividere il segnale in uscita, e portare così uno dei due impulsi all'ingresso 1 dello strumento, per poter visualizzare e misurare la tensione in entrata (v_{in}) e il suo periodo (T). Il secondo cavo BNC lo portiamo ad uno degli ingressi della base di lavoro. In questo modo abbiamo che tutta la base di lavoro viene messa a massa (*o a terra?*) e che un pin è posto alla tensione v_{in} alternata.

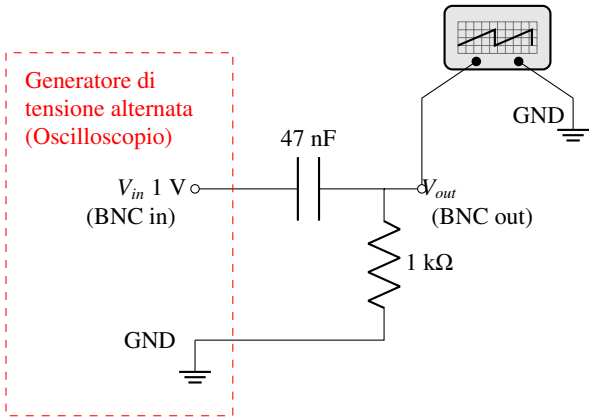

Figura 1 Circuito utilizzato per il filtro passa-alto progettato nell'esperienza, i valori di R e C sono i valori nominali riportati sul componente.

Mettiamo quindi in serie a quest'ultimo il condensatore (C) e la resistenza (R). Il capo libero della resistenza lo poniamo a massa (*o a terra?*).

Colleghiamo infine un pin che sia equipotenziale ad un connettore BNC libero con il nodo di collegamento tra C ed R. Colleghiamo il BNC al secondo input presente sull'oscilloscopio per visualizzare il valore di tensione in uscita (v_{out}).

Per fornire la tensione di input al sistema con il tasto `Wave Gen` visualizziamo l'interfaccia di generazione del segnale, che impostiamo poi su 1V di ampiezza e lasciamo la frequenza libera di essere variata nel corso della presa dati.

Per effettuare le misure di ampiezza dei segnali che leggiamo con l'oscilloscopio, con il tasto `Meas` accedo all'interfaccia per effettuare misure sul segnale in input. Con i tasti sul basso prima imposto la sorgente del segnale che voglio misurare (1 o 2) che tipo di misura voglio effettuare (ampiezza, periodo, ritardo) e poi imposto con il tasto `Settings` i parametri della misura.

Impostiamo quattro misure:

- Misura della tensione che noi forniano al circuito, che leggiamo con la sorgente 1 dell'oscilloscopio. Impostiamo una misura di ampiezza sulla sorgente 1.
- Misura della tensione v_{out}, che leggiamo dal secondo ingresso. Impostiamo quindi una misura di ampiezza sulla sorgente 2.
- Misura del periodo del segnale, che effettuiamo sul segnale v_{in} poichè osserivamo che ha maggiore stabilità rispetto al segnale in uscita al circuito. Impostiamo una misura di Periodo sulla sorgente 1.
- Misura del ritardo del segnale v_{out} rispetto a v_{in}. Impostiamo una misura che legga la differenza temporale tra la salita della sorgente 1 e la salita adiacente del segnale 2.

Lo strumento permette di variare l'intervalli orizzontali e verticali in cui misuriamo il segnale per adattarne al meglio la visualizzazione e fornisce i valori di fondo scala relativi per le tensioni e per i tempi, necessari per ricavare l'errore.

III. METODI SPERIMENTALI

Dato lo strumento sopra descritto, variando il valore della frequenza del segnale generato, partendo da 10Hz e arri-

[a] Esperinza n. 1
[b] s5025011@studenti.unige.it; In presenza in laboratorio per la presa dati
[c] s4942225@studenti.unige.it

Tabella I Dati grezzi (sbagliati/pre-correzione)

v_{in} (mV)		v_{out} (mV)		T (s)		dt (s)	
991	140	4.6	2	0.1	0.02	0.02	0.02
991	140	7.6	2	0.05	0.01	0.011	0.01
991	140	15.2	3	0.02	0.005	0.0046	0.005
991	140	28	4	0.01	0.002	0.0024	0.002
993	140	53	8	0.005	0.001	0.0012	0.001
991	140	125	18	0.002	0.0005	0.00045	0.0005
991	140	243	34	0.001	0.0002	0.000208	0.0002
985	140	445	62	0.0005	0.0001	$8.5e-05$	0.0001
962	140	746	104	0.0002	$5e-05$	$2.1e-05$	$5e-05$
951	140	877	124	0.0001	$2e-05$	$6e-06$	$2e-05$
945	140	913	128	$5e-05$	$1e-05$	$1.6e-06$	$1e-05$
946	140	934	140	$2e-05$	$5e-06$	$2.8e-07$	$5e-06$
946	140	940	140	$1e-05$	$2e-06$	$5e-08$	$2e-06$

vando a 100kHz rilevando tre valori (1-2-5) per ogni decade, annotiamo i valori di v_{in}, v_{out}, T e dt, con i relativi fondo scala.

Riportiamo in Tabella III i valori misurati.

IV. CALCOLI E FIT DATI

Vogliamo ottenere il valore della funzione di trasferimento $|H[\nu]|$, ma poichè si tratta di una funzione complessa preferiamo scriverla in termini del suo modulo e della sua fase. Otteniamo che

$$|H[\nu]| = \frac{v_{in}}{v_{out}}$$

e inoltre

$$|H[\nu]| = \frac{1}{\sqrt{1 + \frac{1}{R^2C^2\omega^2}}}$$

che con $\omega_0 = \frac{1}{RC}$ diventa

$$|H[\nu]| = \frac{1}{\sqrt{1 + \left(\frac{\nu_0}{\nu}\right)^2}} \tag{1}$$

con $\frac{\omega_0}{\omega} = \frac{\nu_0}{\nu}$.

Allo stesso modo per la fase che ottemiamo come

$$\varphi[\nu] = 2\pi\frac{dt}{T}$$

lo possiamo vedere anche come

$$\varphi[\nu] = \arctan\left(\frac{1}{RC\omega}\right)$$

che diventa

$$\varphi[\nu] = \arctan\left(\frac{\nu_0}{\nu}\right) \tag{2}$$

con $\frac{\omega_0}{\omega} = \frac{\nu_0}{\nu}$.

Dal data sheet dello strumeto ricaviamo come calcolare gli errori massimi sui dati misurati che con la regola del 3σ rendiamo statistici. Quindi propaghiamo per ottenere gli errori su $|H[\nu]|$ e $\varphi[\nu]$.

I valori sono riportati in Tabella II.

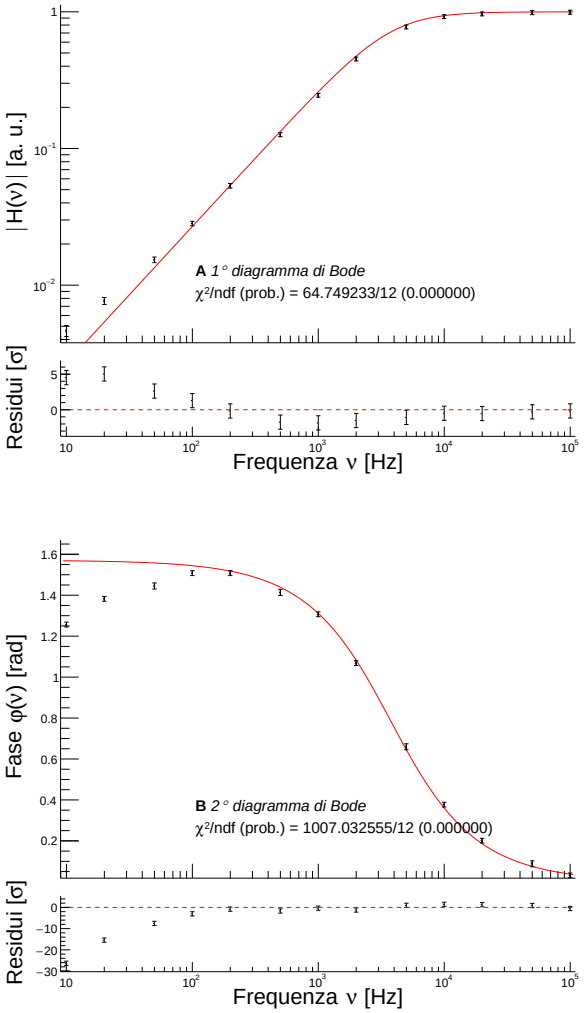

Figura 2 Diagrammi di Bode, grafico preliminare precedente ad una considerazione sui punti a $\nu < 100$Hz.

Dalle tabelle ricaviamo un grafico in scala bilogaritmica della funzione di trasferimento rispetto alla frequenza (ν) e un grafico in scala semilogaritmica sulle frequenze di $\varphi[\nu]$ rispetto a ν. Riportiamo in Figura 2 il grafico prodotto.

Eseguiano un fit dei dati sulla base delle funzioni (1) e (2), e otteniamo quindi il valore di ν_0 che abbiamo impostato co-

Tabella II Valri calcolati (sbagliati/pre-correzione)

Funzione di trasferimento $\lvert H[\nu]\rvert$ [a. u.]		Fase $\varphi[\nu]$ [rad]		Frequenza ν [Hz]	
0.00464178	0.000432655	1.25664	0.0118382	10	0.0184752
0.00766902	0.000454563	1.3823	0.0118859	20	0.0369504
0.015338	0.000720135	1.44513	0.0148892	50	0.11547
0.0282543	0.00105838	1.50796	0.011938	100	0.184752
0.0533736	0.00206972	1.50796	0.011938	200	0.369504
0.126135	0.00411336	1.41372	0.0148732	500	1.1547
0.245207	0.00788168	1.3069	0.0118568	1000	1.84752
0.451777	0.0145359	1.06814	0.0117749	2000	3.69504
0.775468	0.0252639	0.659734	0.0145902	5000	11.547
0.922187	0.0304294	0.376991	0.0116292	10000	18.4752
0.966138	0.0318566	0.201062	0.0116143	20000	36.9504
0.987315	0.0336198	0.0879646	0.0145118	50000	115.47
0.993658	0.0337266	0.0314159	0.0116085	100000	184.752

me parametro nella funzione di fit dal I e dal II diagramma di Bode.

Analizziamo i valori di χ^2/ndfe di prob(χ^2) e osserviamo la compatibilità dei valori trovati. Riportiamo il risultato delle operazioni di calcolo nell'appendice A.

V. CONSIDERAZIONI SU RUMORE A BASSE FREQUENZE

Nell'analisi dei dati attraverso i fit dei due diagrammi di Bode abbiamo escluso i primi tre punti, cioè quelli a frequenze minori di 100Hz. Il motivo di questa esclusione è legato alla presenza di un "rumore" che appunto si manifesta in particolar modo a basse frequenze in quanto la sua ampiezza diventa confrontabile con quella di v_{out}.

Appendice A: Output analisi dati prima della correzione

```
Processing analisi_RC_filter.C...

 **********
    PRIMO DIAGRAMMA DI BODE (AMPIEZZA)
 **********

 FCN=64.7492 FROM MIGRAD    STATUS=CONVERGED      24 CALLS          25 TOTAL
                     EDM=2.78419e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0           3.71715e+03   5.54290e+01   2.19424e-01  -1.34626e-06

** CHI2 / NDF ( PROB. ) 64.7492 / 12 ( 3.03328e-09 )

Frequenza di Taglio da |H(w)|, v = 3717.15 +/- 55.429 Hz

 **********
    SECONDO DIAGRAMMA DI BODE (FASE)
 **********

 FCN=1007.03 FROM MIGRAD    STATUS=CONVERGED      18 CALLS          19 TOTAL
                     EDM=1.07153e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0           3.76629e+03   6.10946e+01   2.27456e-04  -3.14644e-02

** CHI2 / NDF ( PROB. ) 1007.03 / 12 ( 5.76595e-208 )

Frequenza di Taglio da phi(w), v = 3766.29 +/- 61.0946 Hz
** Verifica compatibilita => COMPATIBILE
```

Appendice B: Output analisi dati dopo la correzione

```
Processing analisi_RC_filter.C...

 **********
    PRIMO DIAGRAMMA DI BODE (AMPIEZZA)
 **********

 FCN=9.72567 FROM MIGRAD    STATUS=CONVERGED      22 CALLS          23 TOTAL
                     EDM=5.57532e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0           3.88205e+03   6.75748e+01   1.08018e-01  -4.94157e-06

** CHI2 / NDF ( PROB. ) 9.72567 / 12 ( 0.640013 )

Frequenza di Taglio da |H(w)|, v = 3882.05 +/- 67.5748 Hz

 **********
    SECONDO DIAGRAMMA DI BODE (FASE)
 **********

 FCN=20.0293 FROM MIGRAD    STATUS=CONVERGED      18 CALLS          19 TOTAL
```

```
                    EDM=1.45607e-07    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0           3.78558e+03   6.12585e+01   3.28702e-05  -3.66507e-02

** CHI2 / NDF ( PROB. ) 20.0293 / 9 ( 0.0177321 )

Frequenza di Taglio da phi(w), v = 3785.58 +/- 61.2585 Hz
** Verifica compatibilita => COMPATIBILE
```

Appendice C: Programma di analisi dati

```
#include<vector>
#include<cmath>
#include<iostream>
#include<fstream>
#include<string>

#include<TCanvas.h>
#include<TGraphErrors.h>
#include<TF1.h>
#include<TStyle.h>
#include<TAxis.h>
#include<TMath.h>
#include<TLatex.h>
#include<TLegend.h>

const double title_size = 21;

std::string rawdata = "../dati/presa_dati_2021_10_19_seconda_versione.txt";

void print_mmsg(std::string mmsg){
    std::cout << std::endl
        << " **********" << std::endl
        << "    " << mmsg << std::endl
        << " **********" << std::endl
        << std::endl;
}

void print_stat(TF1* _f){
    std::cout << std::endl
        << "** " << "CHI2 / NDF ( PROB. ) "
        << _f->GetChisquare() << " / " << _f->GetNDF() << " ( " << _f->GetProb() << " )"
        << std::endl << std::endl;
}

std::string compatible(double G1, double errG1,
                       double G2, double errG2){
    double abs_values = abs(G2-G1);
    double err_abs_val = 3*sqrt(pow(errG1, 2) + pow(errG2, 2));
    if(abs_values<err_abs_val){
        return "COMPATIBILE";
    }
    return "NON-COMPATIBILE";
}

void set_TGraphAxis(TGraphErrors* g, std::string ytitle){
    g->SetTitle("");
    g->GetYaxis()->SetTitle(ytitle.c_str());
    g->GetYaxis()->SetTitleOffset(2);
    g->GetYaxis()->SetTitleFont(43);
    g->GetYaxis()->SetTitleSize(title_size);
    g->GetYaxis()->SetLabelFont(43);
    g->GetYaxis()->SetLabelSize(12);
    g->GetYaxis()->CenterTitle();

    g->GetXaxis()->SetTickLength(0.05);
}

void set_ResidualsAxis(TGraphErrors* rg, std::string xtitle, std::string ytitle="Residui [#sigma]"){
    rg->GetXaxis()->SetTitle(xtitle.c_str());
    rg->GetXaxis()->SetTitleOffset(5);
    rg->GetXaxis()->SetTitleFont(43);
    rg->GetXaxis()->SetTitleSize(title_size);

    rg->GetYaxis()->SetTitle(ytitle.c_str());
    rg->GetYaxis()->SetTitleOffset(2);
    rg->GetYaxis()->SetTitleFont(43);
    rg->GetYaxis()->SetTitleSize(title_size);
    rg->GetYaxis()->CenterTitle();

    rg->GetYaxis()->SetLabelFont(43);
    rg->GetYaxis()->SetLabelSize(12);
    rg->GetYaxis()->SetNdivisions(5, 5, 0);
    rg->GetXaxis()->SetLabelFont(43);
    rg->GetXaxis()->SetLabelSize(12);
    rg->GetXaxis()->CenterTitle();

    rg->GetXaxis()->SetTickLength(0.08);
}

double max_to_stat(double value){
    return value/(std::sqrt(3));
}

// funzione calcolo incertezza a partire da fondo scala (per Qualsiasi grandezza)
// tab. VALORI  | Grandezza misurata | errPercent | partitions | fondoscala (range1)
//              | V (tensione)       | 3.5%       | 8          | variabile
//              | T (periodi)        | ?.?%       | ?          | variabile

double get_VRangeErr(double errPercent, int partitions, double range1){
  return errPercent * partitions *  range1;
}
double get_TRangeErr(double range1, double errPercent = 0.0016, int partition = 10){
    return range1 * errPercent * partition;
}

double getH(double vin, double vout){
    return vout / vin;
}

double get_HErr(double Vin, double Vout, double eVin, double eVout){
  return sqrt(pow(eVout / Vin, 2) + pow(eVin * Vout / pow(Vin, 2), 2));
}

double get_phi(double T, double dt){
    return 2 * M_PI * dt / T;
```

```
}

double get_phiErr(double T, double dt, double eT, double edt){
    return 2 * M_PI * sqrt(pow(edt/T, 2) + pow(dt * eT/(pow(T, 2)), 2));
}

void analisi_RC_filter(){

    gStyle->SetFrameLineWidth(0);
    gStyle->SetTextFont(43);
    gStyle->SetLineScalePS(1);

    std::ifstream data(rawdata.c_str());

    std::ofstream out_rawdata("../misc/rawdata_correct.txt"); // carbon copy of original data
    std::ofstream out_cleandata("../misc/cleandata_correct.txt"); // values from rawdata with error
    std::ofstream out_computeddata("../misc/computeddata_correct.txt"); // computed data for final graph

    double Vin, fsVin, Vout, fsVout, T, fsT, dt, fsdt;

    TCanvas* c1 = new TCanvas("c1", "", 600, 1000);
    c1->SetMargin(0.16, 0.06, 0.12, 0.06);
    c1->SetFillStyle(4000);
    c1->Divide(1, 2);

    // Analisi 1mo diagramma di BODE, |H(w)| su w
    c1->cd(1);

    TGraphErrors* H_plot = new TGraphErrors();
    H_plot->SetName("H_plot");
    TF1* H_fit = new TF1("Hf", "1/sqrt(1+(pow([0]/x, 2)))");
    H_fit->SetParameter(0, 3e3);

    TGraphErrors* H_resd = new TGraphErrors();
    TF1* H_res_f = new TF1("H_rf", "0", 10, 1e6);
    H_res_f->SetLineStyle(2);

    TLatex* header = new TLatex();
    header->SetTextFont(43);
    header->SetTextSize(15);

    TPad* Hp1 = new TPad("", "", 0.0, 0.3, 1.0, 1.0);
    TPad* Hp2 = new TPad("", "", 0.0, 0.0, 1.0, 0.295);
    Hp1->SetMargin(0.14, 0.06, 0.0, 0.06);
    Hp1->SetFillStyle(4000);
    Hp1->SetLogx();
    Hp1->SetLogy();
    Hp1->Draw();
    Hp2->SetMargin(0.14, 0.06, 0.4, 1.0);
    Hp2->SetFillStyle(4000);
    Hp2->SetLogx();
    Hp2->Draw();


    // Analisi 2do diagramma di BODE, phi su w
    c1->cd(2);

    TGraphErrors* phi_plot = new TGraphErrors();
    phi_plot->SetName("phi_plot");
    TF1* phi_fit = new TF1("phi_f", "atan([0]/x)");
    phi_fit->SetParameter(0, 3e3);
    phi_fit->SetParLimits(0, 1e3, 1e4);

    TGraphErrors* phi_resd = new TGraphErrors();
    TF1* phi_res_f = new TF1("phi_rf", "0", 10, 1e6);
    phi_res_f->SetLineStyle(2);

    TLatex* phi_header = new TLatex();
    phi_header->SetTextFont(43);
    phi_header->SetTextSize(15);

    TPad* phi_p1 = new TPad("", "", 0.0, 0.3, 1.0, 1.0);
    TPad* phi_p2 = new TPad("", "", 0.0, 0.0, 1.0, 0.295);
    phi_p1->SetMargin(0.14, 0.06, 0.0, 0.06);
    phi_p1->SetFillStyle(4000);
    phi_p1->SetLogx();
    phi_p1->Draw();
    phi_p2->SetMargin(0.14, 0.06, 0.4, 1.0);
    phi_p2->SetFillStyle(4000);
    phi_p2->SetLogx();
    phi_p2->Draw();


    for(int i=0; data >> Vin >> fsVin >> Vout >> fsVout >> T >> fsT >> dt >> fsdt; i++){
        out_rawdata << Vin << " " << fsVin << " " << Vout << " " << fsVout << " " << T << " " << fsT << " " << dt << " " << fsdt << std::endl;
        double eVin, eVout;
        if(fsVin<=0.01){
            eVin = max_to_stat(get_VRangeErr(0.045, 8, fsVin));
        }else{
            eVin = max_to_stat(get_VRangeErr(0.035, 8, fsVin));
        }
        if(1/T<=100){
            // Correzione per punti sotto i 100Hz
            eVout = max_to_stat(get_VRangeErr(0.15, 8, fsVout));
        }else{
            if(fsVout<=0.01){
                eVout = max_to_stat(get_VRangeErr(0.045, 8, fsVout));
            }else{
                eVout = max_to_stat(get_VRangeErr(0.035, 8, fsVout));
            }
        }
        double eT = max_to_stat(get_TRangeErr(fsT));
        double edt = max_to_stat(get_TRangeErr(fsdt));

        out_cleandata << Vin << " " << eVin << " " << Vout << " " << eVout << " " << T << " " << eT << " " << dt << " " << edt << std::endl;

        H_plot->SetPoint(i, 1 / T, Vout / Vin);
        H_plot->SetPointError(i, eT/pow(T, 2), get_HErr(Vin, Vout, eVin, eVout));

        phi_plot->SetPoint(i, 1 / T, 2 * M_PI * dt / T);
        phi_plot->SetPointError(i, eT/pow(T, 2), get_phiErr(T, dt, eT, edt));


        out_computeddata << Vout / Vin << " " << get_HErr(Vin, Vout, eVin, eVout) << " "
                         << 2 * M_PI * dt / T << " " << 2 * M_PI * sqrt(pow(edt/T, 2) + pow(dt * eT/(pow(T, 2)), 2)) << " "
                         << 1 / T << " " << eT/pow(T, 2) << std::endl;
    }
    out_rawdata << "EOF" << std::endl;
    out_cleandata << "EOF" << std::endl;
    out_computeddata << "EOF" << std::endl;
```

```
    // Grafico 1 Bode
    print_mmsg("PRIMO DIAGRAMMA DI BODE (AMPIEZZA)");
    Hp1->cd();
    H_plot->Draw("ap");
    H_plot->Fit("Hf", "", "");

    std::string H_stat="#chi^{2}/ndf (prob.) = "
            +std::to_string(H_fit->GetChisquare())+"/"
            +std::to_string(H_fit->GetNDF())
            +" ("+std::to_string(H_fit->GetProb())+")";

    header->DrawLatexNDC(0.35, 0.15, ("#splitline{#bf{A} #it{1#circ diagramma di Bode}}{" + H_stat + "}").c_str());

    print_stat(H_fit);

    // RESIDUI
    Hp2->cd();

    for(int i=0; i<H_plot->GetN(); i++){
            H_resd->SetPoint(i, H_plot->GetX()[i], (H_plot->GetY()[i] - H_fit->Eval(H_plot->GetX()[i]))/H_plot->GetEY()[i]);
            H_resd->SetPointError(i, 0, 1);
    }
    H_resd->Draw("ap");
    H_res_f->Draw("same");

    double frequenza_taglio_amp = H_fit->GetParameter(0);
    double err_frequenza_taglio_amp = H_fit->GetParError(0);

    std::cout << "Frequenza di Taglio da |H(w)|, v = " << frequenza_taglio_amp << " +/- " << err_frequenza_taglio_amp << " Hz" << std::endl;

    // Grafico 2 Bode
    print_mmsg("SECONDO DIAGRAMMA DI BODE (FASE)");
    phi_p1->cd();
    phi_plot->Draw("ap");
    phi_plot->Fit("phi_f", "", "", 100, 1e6);

    std::string phi_stat="#chi^{2}/ndf (prob.) = "
            +std::to_string(phi_fit->GetChisquare())+"/"
            +std::to_string(phi_fit->GetNDF())
            +" ("+std::to_string(phi_fit->GetProb())+")";

    phi_header->DrawLatexNDC(0.35, 0.15, ("#splitline{#bf{B} #it{2#circ diagramma di Bode}}{" + phi_stat + "}").c_str());

    print_stat(phi_fit);

    // RESIDUI
    phi_p2->cd();

    for(int i=0; i<phi_plot->GetN(); i++){
            phi_resd->SetPoint(i, phi_plot->GetX()[i], (phi_plot->GetY()[i] - phi_fit->Eval(phi_plot->GetX()[i]))/phi_plot->GetEY()[i]);
            phi_resd->SetPointError(i, 0, 1);
    }
    phi_resd->Draw("ap");
    phi_res_f->Draw("same");

    double frequenza_taglio_fase = phi_fit->GetParameter(0);
    double err_frequenza_taglio_fase = phi_fit->GetParError(0);

    std::cout << "Frequenza di Taglio da phi(w), v = " << frequenza_taglio_fase << " +/- " << err_frequenza_taglio_fase << " Hz" << std::endl;

    std::cout << "** Verifica compatibilita => " << compatible(frequenza_taglio_amp, err_frequenza_taglio_amp, frequenza_taglio_fase, err_frequenza_taglio_fase) << std::
          endl;


    set_TGraphAxis(H_plot, "#left|H(#nu)#right| [a. u.]");
    set_ResidualsAxis(H_resd, "Frequenza #nu [Hz]");


    set_TGraphAxis(phi_plot, "Fase #varphi(#nu) [rad]");
    set_ResidualsAxis(phi_resd, "Frequenza #nu [Hz]");

    c1->SaveAs("../fig/RC_bode_corretto.pdf");

    return;
}

#ifndef __CINT__
int main(){
    analisi_RC_filter();
    return 0;
}
#endif
```