Silvio Navaretti (Fedhman)

MATLAB: DIAGRAMMI DI BODE E METODO DEI NODI

13 December 2015

Ognitanto capita di dover risolvere un circuito col metodo dei nodi. Si può far lavorare uno Spice qualunque, ma tanto per cambiare facciamo fare a MATLAB i conti. Forse un "vantaggio" sta nel rimanere nel mondo "analitico".
En passant un rapido esempio di come rappresentare i diagrammi di Bode con MATLAB, utilizzando delle variabili simboliche.
Attenzione: i codici sono piuttosto pesanti, ho inserito i miei tempi di esecuzione per dare un'idea. Se qualcuno ne esegue di simili, potrebbe commentare con i suoi tempi di esecuzione ed i dati della sua macchina?

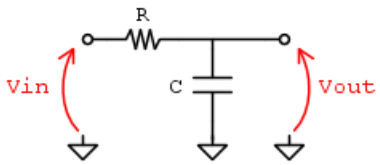
Diagrammi di Bode

Funzioni utili

1i è il modo consigliato dalla MathWorks per scrivere i, l'unità immaginaria.
syms: crea rapidamente delle variabili simboliche. I nomi delle variabli vanno separati da uno spazio. Alle variabili simboliche può essere in seguito sostituito un numero.
matlabFunction(funzione_simbolica) permette di trasformare una funzione simbolica in una "MATLAB function", con cui MATLAB fa i conti più rapidamente rispetto ad una symfun simbolica.

plotyy plotta due funzioni sullo stesso grafico, con assi delle ordinate diversi. Molto utile per i diagrammi di Bode. Su questa funzione ho scritto un altro articolo d'esempio, *Disegnare grafici a doppia scala con MATLAB*.
Vettori_uniti = [vettore1, vettore2, vettore3, ...] concatena due o più vettori sulla stessa riga. Utile per creare assi delle ascisse a step non omogeneo.
abs restituisce il modulo di un numero.
phase restituisce la fase, *in radianti*, di un numero complesso.
radtodeg prende un numero in radianti e lo restituisce in gradi.

Esempio, filtro passa basso del primo ordine



Funzione di trasferimento: $H(jw) = \frac{1}{1 + jwRC} = \frac{1}{1 + j(2\pi f)RC}$

Codice MATLAB

Tempo di esecuzione col workspace pulito: 2.2 s su un vecchio Intel Core 2 Quad da 2.5 GHz.

```
%Crea le due variabili simboliche, istanzia s
syms s f;
s = 1i*(2*pi*f); %s = jw, w = 2*pi*f

R = 10e3; %10 kOhm
C = 1e-6; %1 uF

%Funzione di trasferimento
Htrasf = 1/(1 + s*C*R);

%Per comodità due funzioni differenti per modulo e fase
H_modulo = matlabFunction(20*log10(abs(Htrasf)));
H_fase = matlabFunction(radtodeg(phase(Htrasf)));

%Vettore delle frequenze, ascisse del diagramma di Bode
freq = [1 : 0.1 : 100, 101 : 1 : 1e4];

%hold on per visualizzare la griglia.
figure();
hold on;
grid on;

%Grafico con due assi delle ordinate
[ax, a, p] = plotyy(freq, H_modulo(freq), freq, H_fase(freq));

%Le operazioni di calcolo del minimo e del massimo sono lunghe (in
```

```
%questo caso), conviene farlo una volta sola.
modulo_min = double(min(H_modulo(freq)));
modulo_max = double(max(H_modulo(freq)));
fase_min = double(min(H_fase(freq)));
fase_max = double(max(H_fase(freq)));

%Titolo e nomi degli assi
title(['Bode']);
xlabel('Frequenza [Hz]');
ylabel(ax(1),'Ampiezza [dB]');
ylabel(ax(2),'Fase [°]');

%Imposta i limiti degli assi.
ax(1).XLim = [min(freq), max(freq)];
ax(2).XLim = [min(freq), max(freq)];
ax(1).YLim = [modulo_min, modulo_max];
ax(2).YLim = [fase_min, fase_max];

%Imposta la scala logaritmica delle ascisse
set(ax(1),'xscale','log');
set(ax(2),'xscale','log');

%Calcola due vettori, uno per lo step delle ordinate1 (ampiezza)
%e l'altro per lo step delle ordinate2 (fase).
Ytick_ampiezza = modulo_min : (modulo_max -modulo_min)/10 : modulo_max;
Ytick_fase = fase_min : (fase_max -fase_min)/20 : fase_max;

%Imposta gli step dei due assi delle ordinate.
ax(1).YTick = Ytick_ampiezza;
ax(2).YTick = Ytick_fase;
```

Output del codice

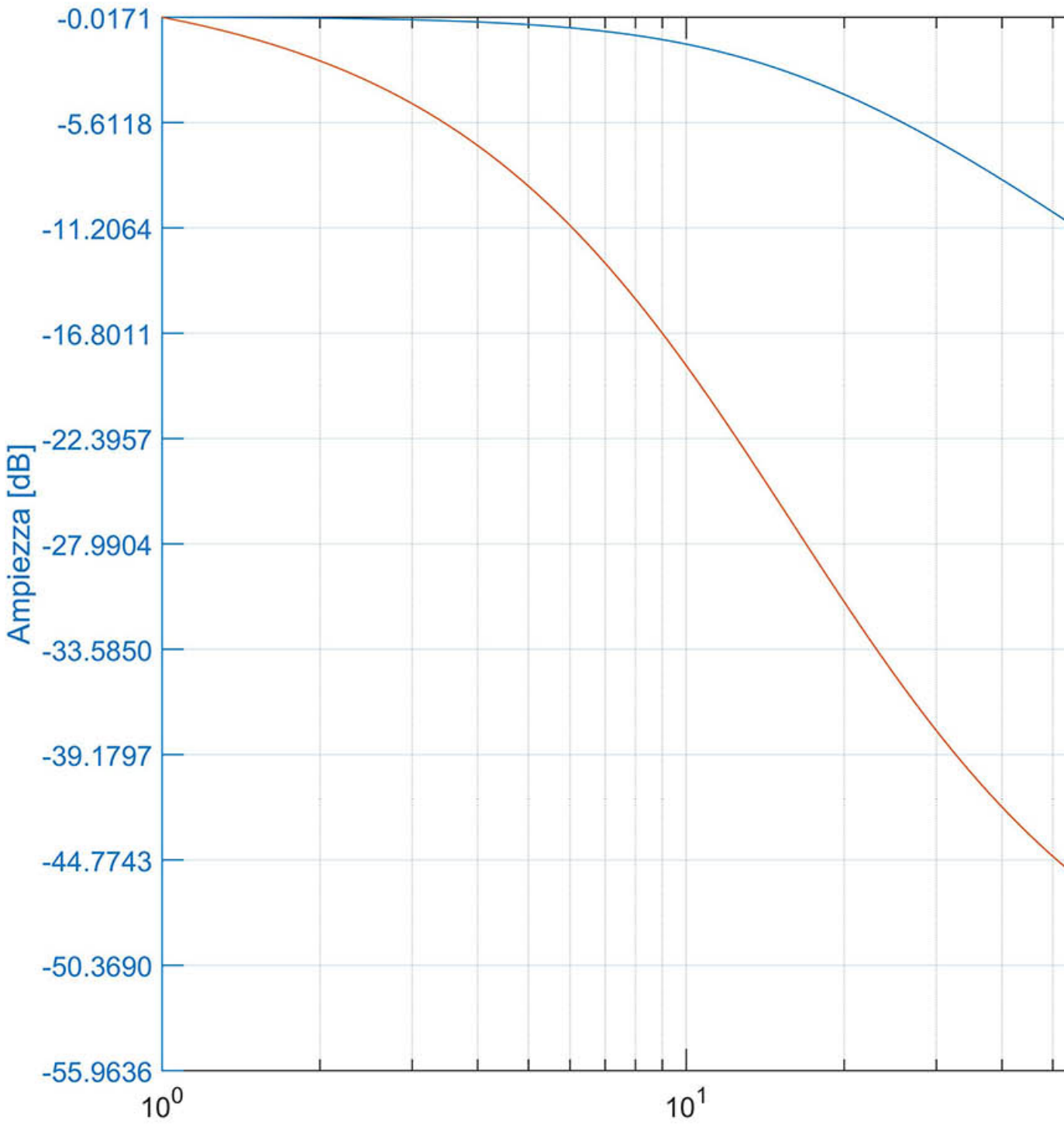


Diagramma di Bode del passa basso. Cliccare per ingrandire l'immagine.

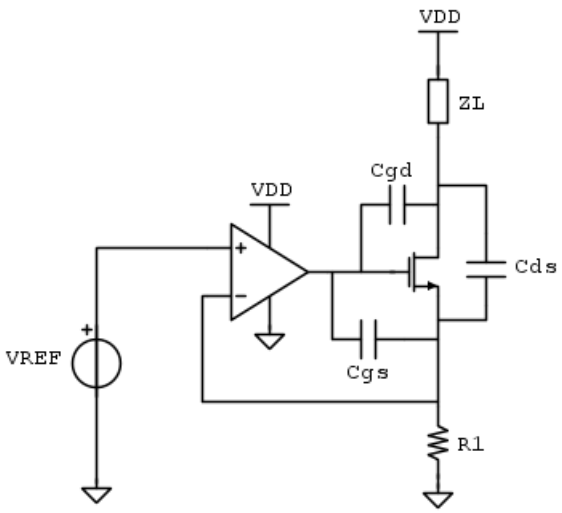
Ulteriori commenti al codice:

Alla riga 16 il vettore delle frequenze è settato in modo da rappresentare adeguatamente il diagramma di Bode in esame, cercando di mantenere ragionevole il tempo di esecuzione. Modificarlo a piacere per restringere/allargare l'asse delle ascisse.
Alle righe 50 e 51 sono stati settati gli step degli assi delle ordinate. Modificare quel 10 (riga 50) e quel 20 (riga 51) per variare gli step.

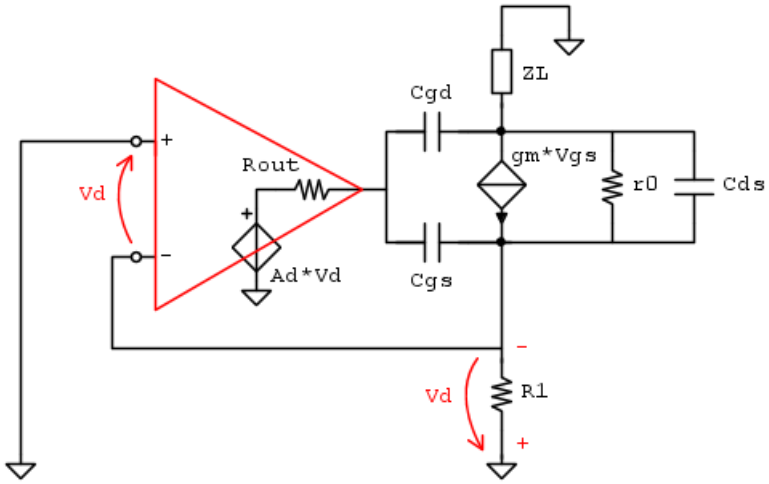
Metodo dei nodi

Nota: diverse calcolatrici scientifiche non programmabili risolvono sistemi lineari 3x3 senza fiatare (tipo la mia Casio fx-991ES da 15€).

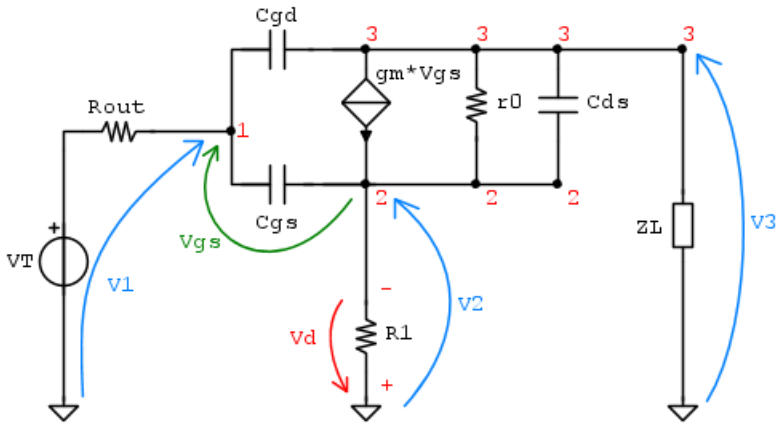
Per analizzare la stabilità del seguente circuito:



Si vuole rappresentare il diagramma di Bode del rapporto di ritorno T, calcolato col metodo di Rosenstark, con la V_{REF} al riferimento e scegliendo di sostituire il pilotato dell'op amp con un generatore di test.
$T = \frac{-A_d \cdot v_d}{V_T}$, A_d è il guadagno ad anello aperto dell'op amp.



I tre condensatori sono tre capacità parassite del MOS.
Forse ci sono altri metodi più furbi, ma ci si arma di pazienza e si identificano i nodi:



$Z_{DS} = r0 \,//\, C_{ds}$

$$1: \quad V_1\left(-\frac{1}{r_{out}} + sC_{gs} + sC_{gd}\right) + V_2\left(-sC_{gs}\right) + V_3\left(-sC_{gd}\right) = -\frac{V_t}{r_{out}}$$

$$2: \quad V_1\left(-sC_{gs}\right) + V_2\left(sC_{gs} + \frac{1}{R_1} + \frac{1}{Z_{DS}}\right) + V_3\left(-\frac{1}{Z_{DS}}\right) = -g_m V_{GS}$$

$$3: \quad V_1\left(-sC_{gd}\right) + V_2\left(\frac{1}{Z_{DS}}\right) + V_3\left(\frac{1}{Z_L} + \frac{1}{Z_{DS}} + sC_{gd}\right) = g_m V_{GS}$$

Sostituendo $V_{GS} = V_1 - V_2$:

$$
\begin{aligned}
1: &\quad V_1\left(-\frac{1}{r_{out}}+sC_{gs}+sC_{gd}\right)+V_2\left(-sC_{gs}\right)+V_3\left(-sC_{gd}\right)=-\frac{V_T}{r_{out}}\\
2: &\quad V_1\left(-sC_{gs}+g_m\right)+V_2\left(sC_{gs}+\frac{1}{R_1}+\frac{1}{Z_{DS}}-g_m\right)+V_3\left(-\frac{1}{Z_{DS}}\right)=0\\
3: &\quad V_1\left(-sC_{gd}-g_m\right)+V_2\left(\frac{1}{Z_{DS}}+g_m\right)+V_3\left(\frac{1}{Z_L}+\frac{1}{Z_{DS}}+sC_{gd}\right)=0
\end{aligned}
$$

Per calcolare T serve $v_d = -V_2$.

Il comando di MATLAB che risolve sistemi lineari è \
Per evitare di commettere (troppi) errori nel scrivere la matrice simbolica su MATLAB si dà un nome ad ogni coefficiente, in questo modo:

$$
\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}=\begin{bmatrix} N1 \\ N2 \\ N3 \end{bmatrix}
$$

Ho rimandato l'analisi del circuito al prossimo weekend, per cui per provare il codice ho tirato più o meno a caso i parametri.

Codice MATLAB

Tempo di esecuzione col workspace pulito: 11.3 s su un vecchio Intel Core 2 Quad da 2.5 GHz.

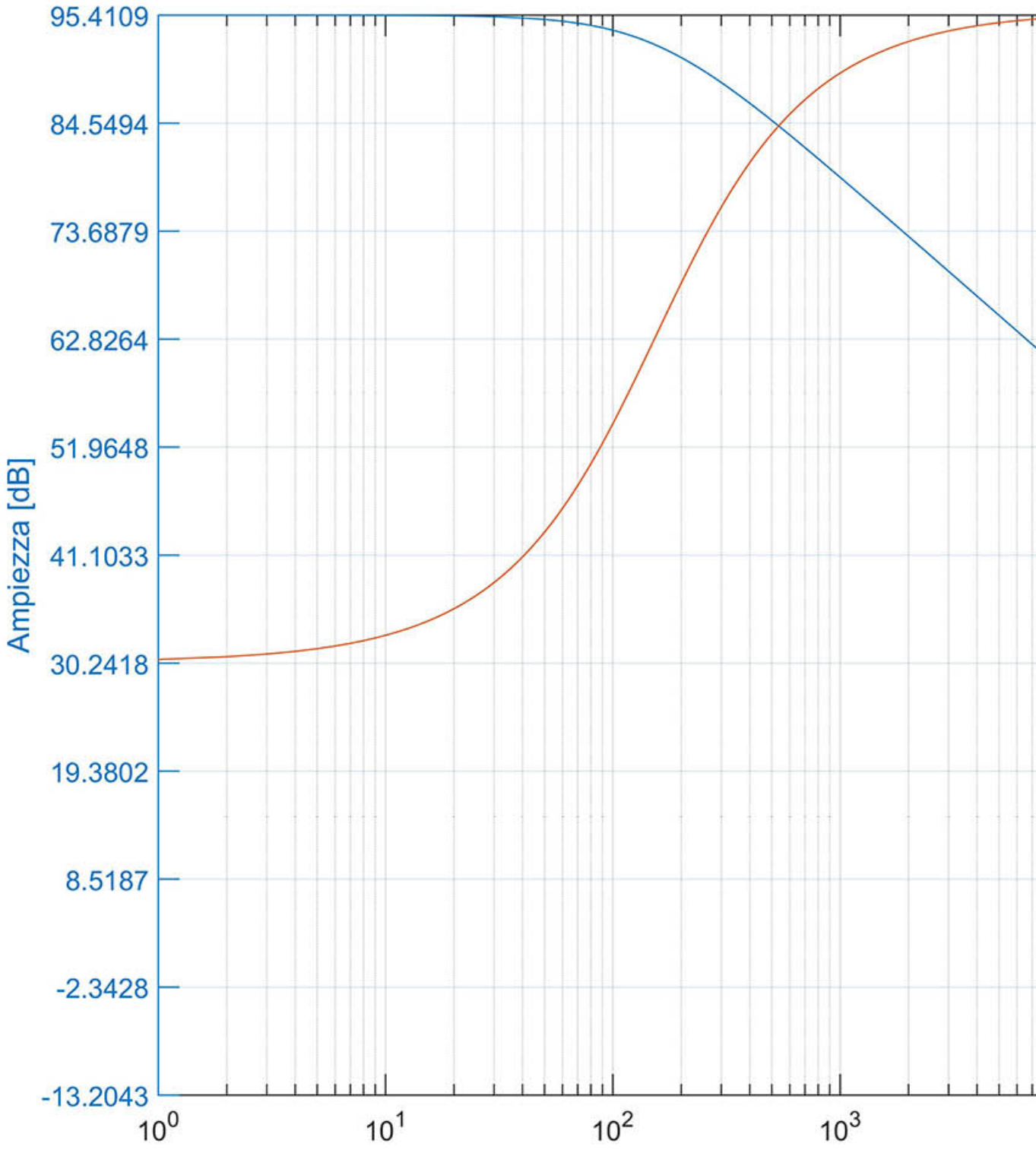
```
%Crea le variabili simboliche.
syms A B C D E F G H I N1 N2 N3 Vt s f;


%Definisce s.
s = 1i*(2*pi)*f;


%Definisce (per ora a caso) i parametri del circuito.
Ad = 1e6;
rout = 10e6;
Cgs = 50e-12;
Cgd = 50e-12;
Cds = 50e-12;
r0 = 10e3;
R1 = 56;
Zds = 1/(1/r0 +s*Cds);
gm = 10e-4;
ZL = 2;
```

```
%Definisce le variabili simboliche della matrice delle conduttanze.
A = -1/rout +s*(Cgs +Cgd);
B = -s*Cgs;
C = -s*Cgd;D = -s*Cgs +gm;
E = s*Cgs +1/R1 +1/Zds -gm;
F = -1/Zds;
G = -s*Cgd -gm;
H = -1/Zds +gm;
I = +s*Cgd +1/ZL +1/Zds;


%Definisce le variabili simboliche del vettore colonna delle correnti.
N1 = -Vt/rout;
N2 = 0;
N3 = 0;


%Crea la matrice delle conduttanze. Si potevano inserire direttamente
%le conduttanze, ma è molto più facile commettere errori.
Conduttanze = [A, B, C; D, E, F; G, H, I];


%Crea il vettore colonna (attenzione ai ;) delle correnti.
Correnti = [N1; N2; N3];


%Calcola le soluzioni e T.
soluzioni = Conduttanze\Correnti;
T = -Ad.*soluzioni(2)./Vt;


%Definisce per comodità due funzioni
H_modulo = matlabFunction(20*log10(abs(T)));
H_fase = matlabFunction(radtodeg(phase(T)));


%Definisce il vettore riga delle frequenze, ascissa di Bode.
freq = [1 : 1 : 1e3, 1e3 : 20 : 1e5, 1e5 : 80 : 1e8, 1e8 : 320 : 1e9];


%hold on per visualizzare la griglia.
```

```
figure();
hold on; grid on;
%Grafico con due assi delle ordinate
[ax, a, p] = plotyy(freq, H_modulo(freq), freq, H_fase(freq));


%Le operazioni di calcolo del minimo e del massimo sono lunghe (in
%questo caso), conviene farlo una volta sola.
modulo_min = double(min(H_modulo(freq)));
modulo_max = double(max(H_modulo(freq)));
fase_min = double(min(H_fase(freq)));
fase_max = double(max(H_fase(freq)));


%Titolo e nomi degli assi.
title(['Bode']);
xlabel('Frequenza [Hz]');
ylabel(ax(1),'Ampiezza [dB]');
ylabel(ax(2),'Fase [°]');


%Imposta i limiti degli assi.
ax(1).XLim = [min(freq), max(freq)];
ax(2).XLim = [min(freq), max(freq)];
ax(1).YLim = [modulo_min, modulo_max];
ax(2).YLim = [fase_min, fase_max];


%Imposta la scala logaritmica delle ascisse.
set(ax(1),'xscale','log');
set(ax(2),'xscale','log');


%Calcola due vettori, uno per lo step delle ordinate1 (ampiezza)
%e l'altro per lo step delle ordinate2 (fase).
Ytick_ampiezza = modulo_min : (modulo_max -modulo_min)/10 : modulo_max;
Ytick_fase = fase_min : (fase_max -fase_min)/20 : fase_max;


%Imposta gli step dei due assi delle ordinate.
ax(1).YTick = Ytick_ampiezza;
ax(2).YTick = Ytick_fase;
```

Output del codice

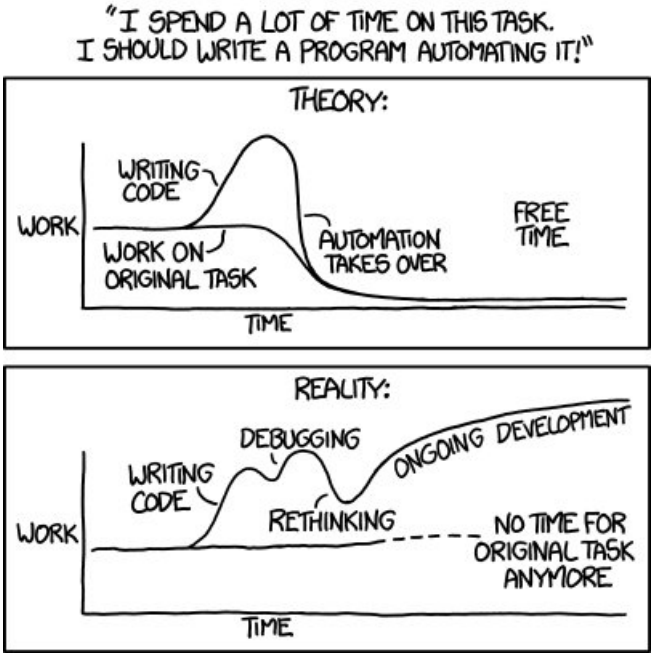
I parametri del circuito sono stati tirati a caso - ho rimandato l'analisi, distratto dal codice


95.4109
84.5494
73.6879
62.8264
51.9648
41.1033
30.2418
19.3802
8.5187
-2.3428
-13.2043
Ampiezza [dB]
10^0
10^1
10^2
10^3

Output del codice - cliccare per ingrandire.

Commento finale

Sono passate settimane e ancora non ho finito di analizzare il circuito.



https://xkcd.com/1319/

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Fedhman:diagrammi-di-bode-e-metodo-dei-nodi-con-matlab"