```
s = tf('s');

Gp = 100 / (s^2 + 5.5*s + 4.5)
```

```
Gp =

         100
  -----------------
  s^2 + 5.5 s + 4.5

Continuous-time transfer function.
```

```
Gs = 1;
Ga = 0.014;
Gr = 1;
Gd = tf(1);

Da0_max = 1.5e-3;

ap_max = 16e-2;
omegap_max = 0.03;

as_max = 2e-1;
omegas_min = 60;
```

Trovare Kc

```
Kd = 1;
R0 = 1;
Gf = 1 / (Gs * Kd)
```

```
Gf = 1
```

```
er_ss_max = 1.5e-1;
nu_r = 1;
Kp = dcgain(Gp)
```

```
Kp = 22.2222
```

```
Kc_min_mod = abs( Kd^2 * R0 / (Kp * Ga) ) / er_ss_max
```

```
Kc_min_mod = 21.4286
```

```
% Errore su da0.
eda_ss_max = 4.5e-3;
nu_da = 0;
% ~> qualsiasi Kc va bene.
```

Transitorio

```
% dp
edp_ss_max = 2e-3;
M_LF = edp_ss_max / ap_max
```

```
M_LF = 0.0125
```

```
omega_L = omegap_max * 10^( -mag2db(M_LF)/40 )
```

```
omega_L = 0.2683
```

```
eds_ss_max = 8e-4;
M_HF = eds_ss_max * Gs / as_max
```

```
M_HF = 0.0040
```

```
omega_H = omegas_min * 10^( mag2db(M_HF)/40 )
```

```
omega_H = 3.7947
```

```
intervallo_crossover = [omega_L*2 omega_H/2]
```

```
intervallo_crossover = 1×2
    0.5367    1.8974
```

Specifiche transitorio

```
raise_time = 2;
settling_time = 8; % alpha = 5%
overshoot = 0.12;

% Calcoliamo lo smorzamento minimo.
z = abs( log(overshoot) ) / sqrt( pi^2 + log(overshoot)^2 )
```

```
z = 0.5594
```

```
% Calcola la sovraelongazione massima di T ed S.
Tp_max = 1 / (2*z*sqrt(1 - z^2) )
```

```
Tp_max = 1.0783
```

```
Sp_max = 2*z*sqrt(2 + 4*z^2 + 2*sqrt(1 + 8*z^2)) / (4*z^2 -1 + sqrt(1 + 8*z^2))
```

```
Sp_max = 1.3935
```

```
% Calcolo le frequnze minime risultanti dai vincoli sui tempi.
omega_c_tr_min = 1 / raise_time * ...
    sqrt(sqrt(1 + 4*z^2) - 2*z^2) / sqrt(1 - z^2) * (pi - acos(z))
```

```
omega_c_tr_min = 1.2211
```

```
alpha = 0.05;
omega_c_ts_min = 1 / settling_time * ...
    sqrt(sqrt(1 + 4*z^2) - 2*z^2) * (-1) * log(alpha) / z
```

```
omega_c_ts_min = 0.6260
```

```
intervallo_crossover = ...
    [max([omega_c_ts_min omega_c_tr_min intervallo_crossover(1)]) omega_H/2]
```

```
intervallo_crossover = 1×2
    1.2211    1.8974
```

Progetto di Gc

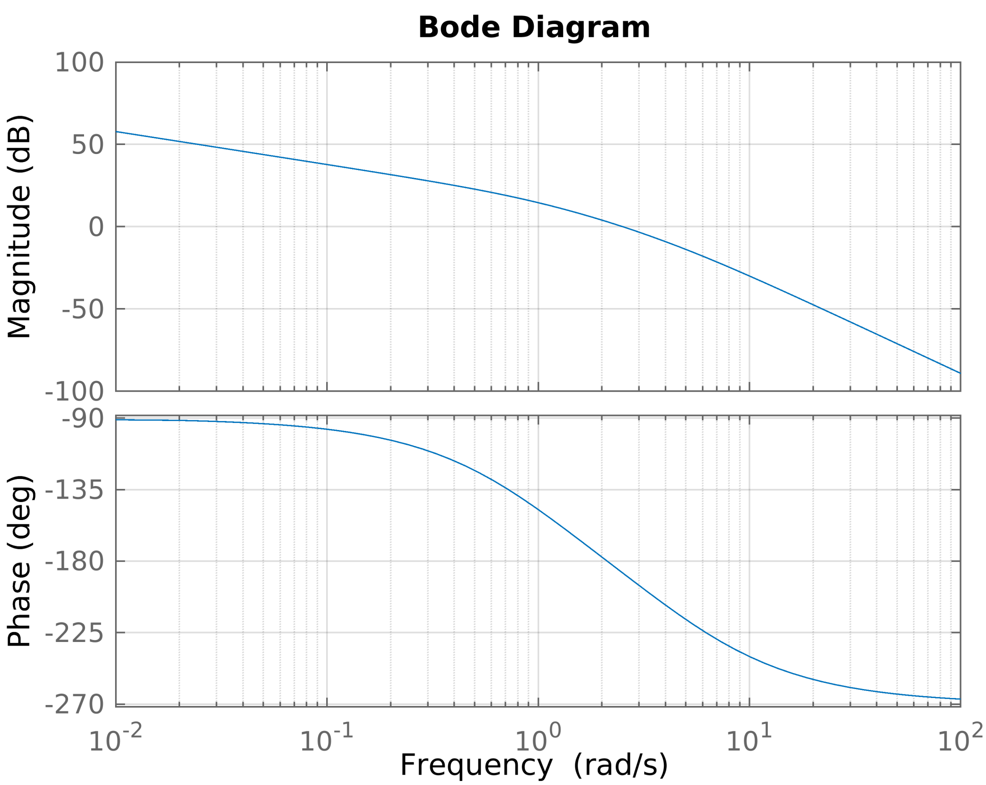
```
% Proviamo con Kc > 0.
Kc = 25;
Gc = Kc / s
```

```
Gc =

  25
  --
  s

Continuous-time transfer function.
```

```
L =  minreal(zpk(Gc * Gf * Gs * Ga * Gp))
```

```
L =

        35
  ---------------
  s (s+4.5) (s+1)

Continuous-time zero/pole/gain model.
```

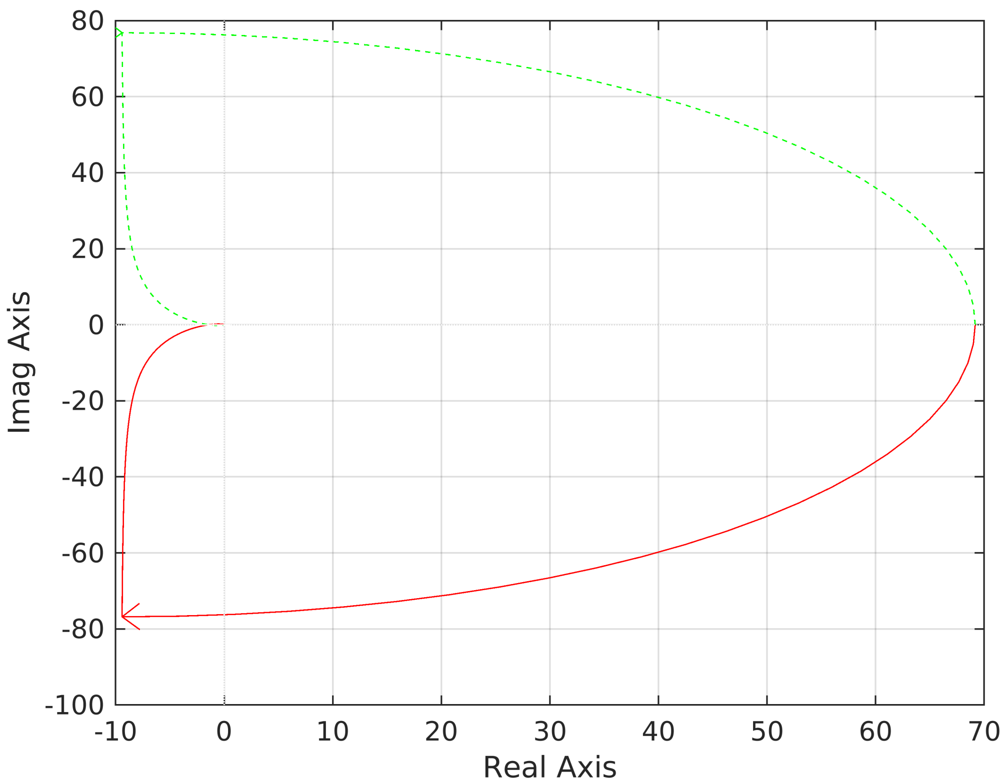
```
bodeplot(L)
grid on
hold off
```




```
[a, b] = tfdata(L, 'v');
nyquist1(a,b)
```

```
p = 3×1
         0
   -4.5000
   -1.0000
P = 0
P = number of Open loop poles in rhp
N = -1
N =  number of anti-clockwise encirclements
Z = 1
Z = number of closed loop poles in rhp
```

```
grid on
hold off
```




```
% N = 2, il segno di Kc non va cambiato.

% Scelgo una omegac_des come media dell'intervallo
omegac_des = mean(intervallo_crossover)
```

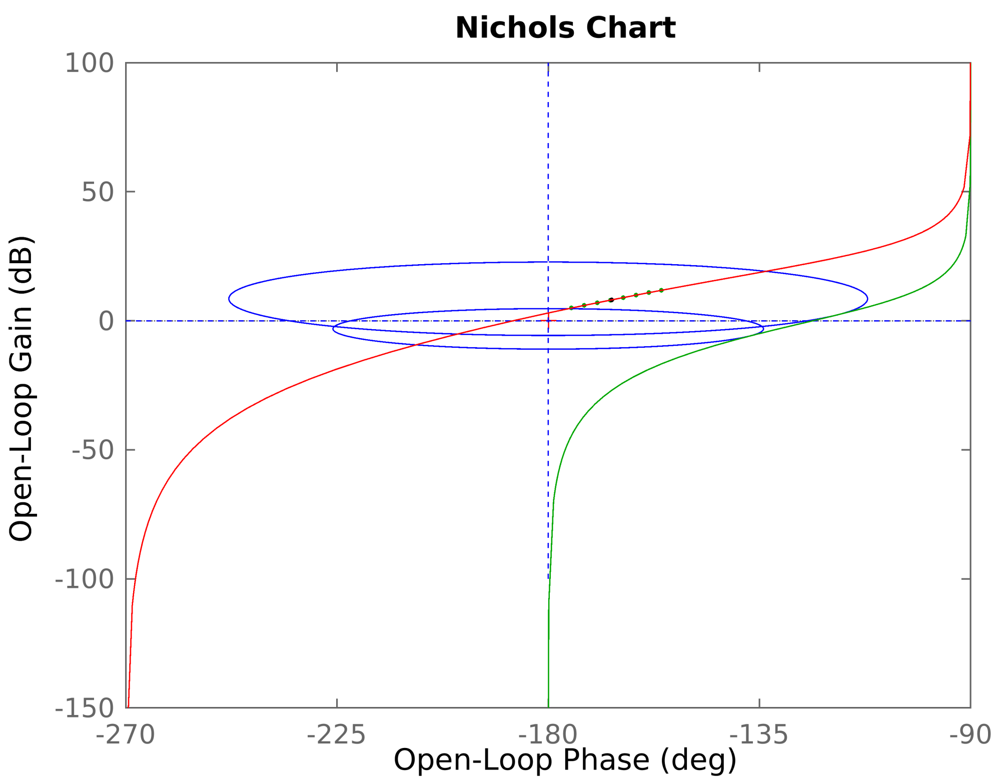
```
omegac_des = 1.5592
```

```
% Scrivo la funzione prototipo.
omegan = omegac_des / sqrt(sqrt(1 + 4*z^2) - 2*z^2);
prot = tf(1, [1/omegan^2 2*z/omegan 1])
```

```
prot =

             1
  -------------------------
  0.3598 s^2 + 0.6711 s + 1
```

```
Continuous-time transfer function.
```

```
Lprot = prot/(1 - prot);

% Adesso diamo un'occhiata al diagramma di Nichols ed alle regioni
% proibite.
myngridst(Tp_max, Sp_max)
nichols(Lprot, 'green')
nichols(L, 'red')
nichols(L, '.g', {intervallo_crossover(1) intervallo_crossover(2)})
nichols(L, '.black', omegac_des)
hold off
```



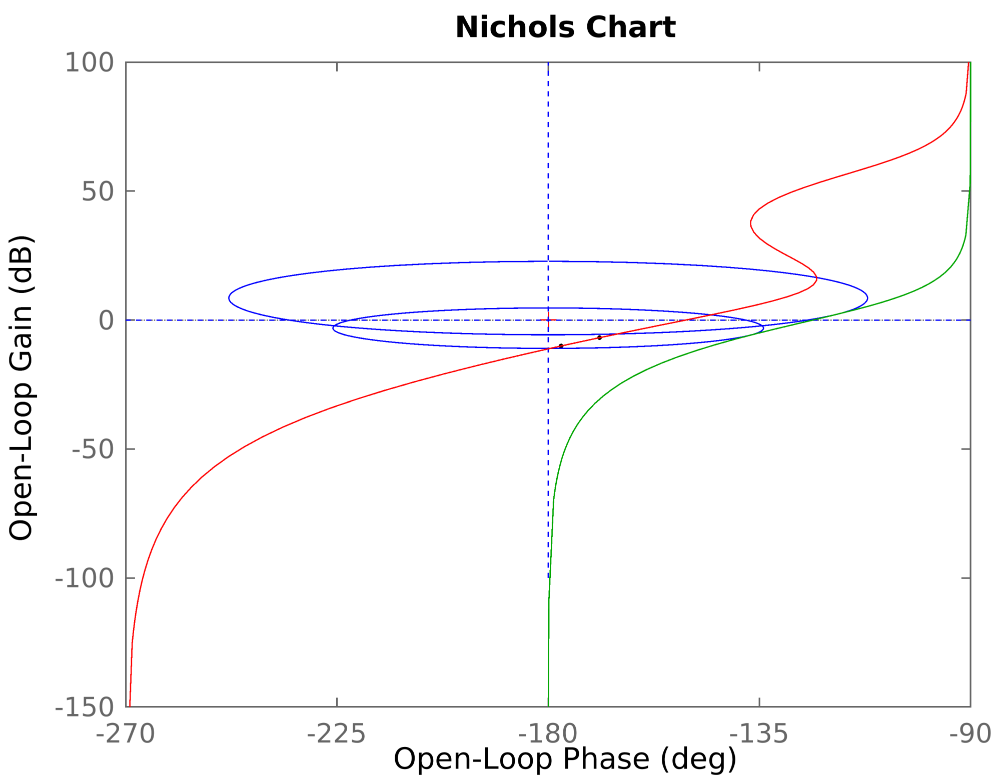

```
% Aggiungiamo una rete lag per far scendere il modulo.
% La faccio scendere del doppio: il motivo e' che il diagramma e' troppo
% interno alle regioni proibite, dunque applichero una spostamento di fase
% ed un aumento del modulo con una rete lead.
Gc = minreal(zpk(lag(omegac_des/100, 5.6) * Kc / s))
```

```
Gc =

  4.4643 (s+0.08732)
  ------------------
    s (s+0.01559)

Continuous-time zero/pole/gain model.
```

```
L = Gc * Gp * Ga * Gf * Gs
```

```
L =

       6.25 (s+0.08732)
  ----------------------------
  s (s+0.01559) (s+1) (s+4.5)

Continuous-time zero/pole/gain model.
```

```
myngridst(Tp_max, Sp_max)
nichols(Lprot, 'green')
nichols(L, 'red')
nichols(L, '.black', [omegac_des intervallo_crossover(2)])
hold off
```



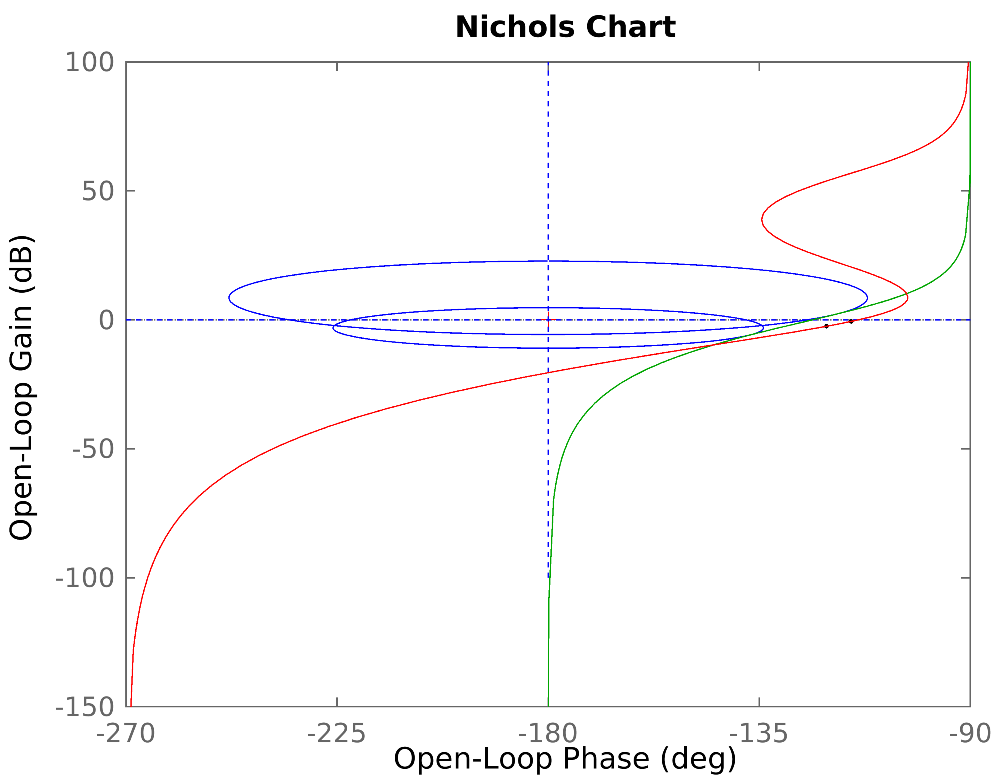

```
% Ora aggiungo una rete LEAD.
% La fase deve guadagnare 46deg, per intersecare il punto dove si trova la
% funzione prot, ed il suo modulo deve aumentare di 8.12dB.
z = omegac_des/1.8; m = 14;
Gc = minreal(zpk(lead(z, m) * Gc))
```

```
Gc =

  62.5 (s+0.8662) (s+0.08732)
  ----------------------------
    s (s+12.13) (s+0.01559)

Continuous-time zero/pole/gain model.
```

```
L = Gc * Gp * Ga * Gf * Gs
```

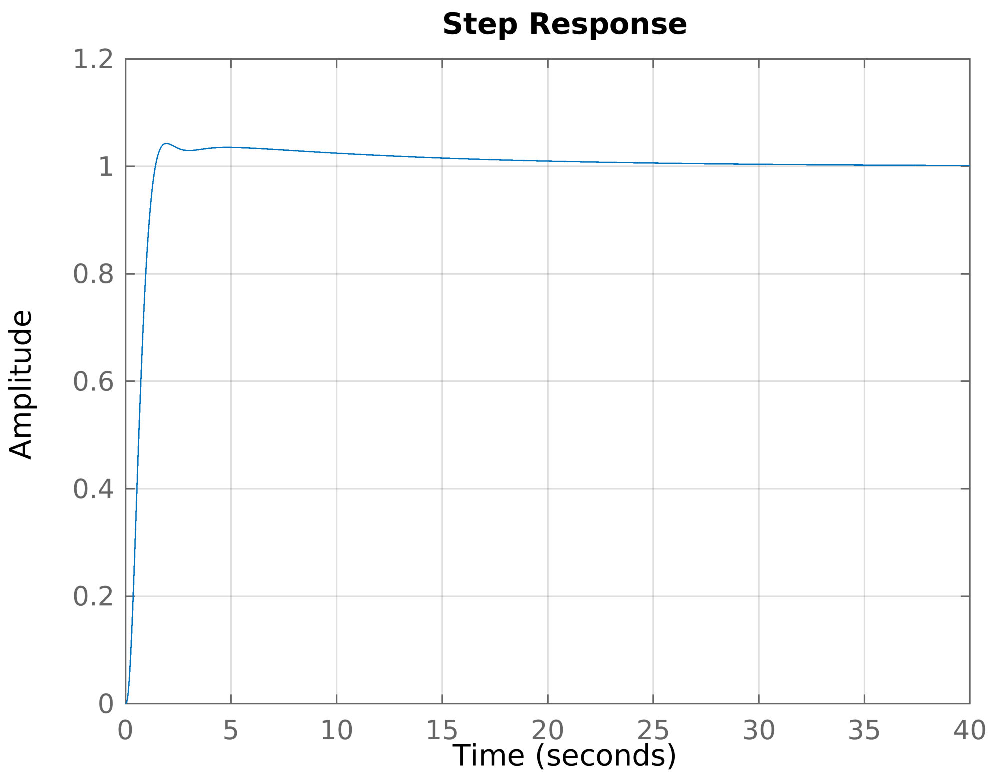
```
L =

       87.5 (s+0.8662) (s+0.08732)
  ------------------------------------
  s (s+12.13) (s+4.5) (s+1) (s+0.01559)

Continuous-time zero/pole/gain model.
```

```
myngridst(Tp_max, Sp_max)
nichols(Lprot, 'green')
nichols(L, 'red')
nichols(L, '.black', [omegac_des intervallo_crossover(2)])
hold off
```



Controllo le specifiche sul dominio del tempo.

```
step(L/(1+L))
% raise_time = 1.43
% settling_time = 1.96
% overshoot = 0.043
grid on
hold off
```



Controllo specifiche sul dominio della frequenza

```
magp = bode(1/(1+L), omegap_max);
ap_max * magp, edp_ss_max, ap_max * magp < edp_ss_max
```

```
ans = 0.0013
edp_ss_max = 0.0020
ans = logical
   1
```

```
mags = bode(L/(1+L) / Gs, omegas_min);
as_max * mags, eds_ss_max, as_max * mags < eds_ss_max
```

```
ans = 7.9196e-05
eds_ss_max = 8.0000e-04
ans = logical
   1
```

```
% Anche le specifiche sulla risposta in frequenza vengono rispettate.
```

```
function [rete] = lead(z, m)
s = tf('s');
rete = (1 + s/z) / (1 + s/(m*z));
end

function [rete] = lag(p, m)
s = tf('s');
```

```
rete = (1 + s/(m*p)) / (1 + s/p);
end
```