AMPLITUD AND PHASE MARGIN DETECTION WITH ON-LINE PID CONTROLLER

Damir Vrančič
Youbin Peng*

August, 1994

*Department of Control Engineering
Free University of Brussels
Belgium
# Table of Contents

1. Introduction ........................................................................................................................................ 2

2. Tuning methods .................................................................................................................................. 7
   2.1 Direct method ............................................................................................................................ 7
   2.2 Successive change of $T_i$ ........................................................................................................... 10
   2.3 Correlation compensation method ............................................................................................ 12

3. Detection of Nyquist curve by relay excitation ................................................................................. 16
   3.1 Amplitude and phase computation ............................................................................................ 16
   3.2 Errors .......................................................................................................................................... 19

4. References ........................................................................................................................................ 21

5. Appendix ........................................................................................................................................... 22
1. Introduction

The goal of this report was to find such controller parameters that amplitude margin ($A_m$) and phase margin ($\phi_m$) of the controlled system will be as desired so that, if process have transfer function $G_P(s)$, we want to find such controller $G_C(s)$ to reach desired points A and B on the Nyquist curve in Fig. 1.

![Nyquist plot of the process ($G_P(s)$) and process with a controller ($G_P(s)G_C(s)$)](image)

*Fig. 1. Nyquist plot of the process ($G_P(s)$) and process with a controller ($G_P(s)G_C(s)$)*

![Bode plot of the process with a controller](image)

*Fig. 2. Bode plot of the process with a controller*
Full line (see Fig. 1) represents process’ Nyquist curve and dashed line represents correction of the curve made by controller. Fig. 2 shows representation of the amplitude and phase margin in Bode plot.

Points A and B (see fig. 1) can be detected by many ways. One of the most practical way to do that is to use the relay feedback method [1]. In this method, relay is connected in closed-loop with a process as shown in Fig. 3.

![Fig. 3. Relay feedback method](image)

Characteristics of the used relay is as follows:

![Fig. 4. Relay characteristics](image)

where \( \varepsilon \) represents hysteresis of the relay and \( d \) is an output value. Describing function of such relay is represented in Fig. 5.

![Fig. 5. Describing function of the relay](image)
where \(-1/N(a)\) is

\[
-\frac{1}{N(a)} = -\frac{\pi}{4d} \sqrt{a^2 + \varepsilon^2} - i \frac{\pi \varepsilon}{4d}
\] (1)

The system in Fig. 3 will oscillate at the point where \(G_p(j\omega)\) intersects the relay describing function. To detect point A (see Fig. 1), relay without hysteresis (\(\varepsilon=0\)) should be used. In that case, imaginary part in (1) will be 0 and \(G_p(j\omega)\) and relay describing function will intersect on real axis as shown in Fig. 6.

**Fig. 6. Detection of the amplitude margin**

P is an intersection point and it represents the point at which system described in Fig. 3 oscillates. With detected point P we can calculate amplitude margin as shown in Fig. 1.

To detect phase margin, we have to change hysteresis of used relay such that describing function will intersect desired phase margin (see Fig. 7).

**Fig. 7. Detection of the phase margin**
From (1) and Fig. 7, we can calculate $\varepsilon$:

$$
\varepsilon = \frac{4d}{\pi} \sin \phi_m
$$

If process phase margin is bigger than desired, $G_p(j\omega)$ and describing function will intersect at point $D_1$ and if process phase margin is smaller than desired, intersection point would be $D_2$ (see Fig. 8).

![Fig. 8. System with too small phase margin, -- system with too big phase margin](image)

From intersection point position we can calculate controller parameters to achieve desired amplitude or phase margin. Then we can change the position of the switch in Fig. 3 from A to B and push the controller into closed-loop configuration. If our goal is to satisfy amplitude and phase margin as well, we have to detect more points on the Nyquist curve, usually by changing relay hysteresis or adding some additional function blocks in line with relay [2, 3].

The idea, presented here, is to use modified relay feedback method as shown in Fig. 9.

By this method, controller is always connected in line with the process. It solves some problems related with classical relay feedback method:

- We doesn’t have to add offset signal at the process input to achieve desired set-point (see Fig. 3)
- In classical method, if the process is low-order, we have to use some additional blocks (like integrator [3]) in line with tested process to achieve oscillation, while in new scheme the controller already does it.
In classical method we can’t take into account some controller specialities like input filter (analog or/and digital), delayed output, sampling time, ... or if we can do so, the computation would be too complex. Usually fine tuning is required afterwards. In presented method, the controller is all the time in line with a process, so when switching from A to B (see Fig. 9), no fine tuning is required.

*Fig. 9. Modified scheme of the relay tuning method*
2. Tuning methods

Here some tuning methods to achieve amplitude and phase margin will be shown. To simplify presentation, we will use PI controller. PID controller can be used as well with some modifications \[1, 2, 3\]. Actual amplitude and phase margin will be detected as described in previous section (Fig. 6 and 7). Supported simulations are simplified in the way we calculated intersection points from Nyquist curve of \( G_p(j\omega)G_C(j\omega) \) obtained in program package MATLAB. Next section (3.) describes how to use tuning methods with a relay.

Basic idea of tuning PI controller is to satisfy both, amplitude and phase margin (\( A_m \) and \( \phi_m \) respectively) by iteratively changing \( K_a \) and \( T_i \). Three different methods of tuning are presented.

2.1 Direct method

Fig. 10 shows typical situation during tuning procedure. Points C and D are actually detected and our goal is to move them toward points A and B respectively.

![Nyquist plot](image)

*Fig. 10. Tuning procedure: moving point C toward A and point D toward B*
To move point C toward A, we can change controller proportional gain $K_p$:

$$K_p = K_p \frac{A_m}{A_a}, \quad (3)$$

where $A_a$ is measured amplitude margin and $A_m$ is a desired one. $K_p$ on the right side is the previous (old) one.

Now, Nyquist curve changes the shape:

![Nyquist curve after changing $K_p$](image)

**Fig. 11. Nyquist curve after changing $K_p$**

Point C moved to point A. But point D can not move directly to point B. With $T_i$ of the controller we can rotate point D from $\phi_a$ to $\phi_m$. But if we would like to move rotated point D directly to B, we would have to change $K_p$ and point C will move out from A. So, in that method we will only rotate point D to angle $\phi_m$ with integral time constant $T_i$ of controller.

Controller transfer function is:

$$|G_c(j\omega)| = K_p \frac{1}{\sqrt{1 + \frac{1}{\omega^2 T_i^2}}} \quad (4)$$
\[
\phi_C(j\omega) = \arctan\left(-\frac{1}{\omega T_i}\right), \quad (5)
\]

where \(\phi_C(\omega)\) represents controller’s phase shift. To rotate point D from angle \(\phi_a\) to \(\phi_m\), \(T_i\) have to be changed:

\[
T_i = \frac{1}{\omega_D \tan \left[\arctan \left(\frac{1}{\omega_D T_i}\right) + \phi_a - \phi_m\right]}, \quad (6)
\]

where \(\omega_D\) represents ultimate frequency at point D and \(T_i\) on the right side of equation represents old value of \(T_i\). In the same time change of \(T_i\) will cause change of absolute gain at point C (4). Intersection point will move out from the point A and procedure have to be repeated.

The result of tuning with presented method is shown in Fig. 12. X axis represents number of iterations, where one iteration means calculating a new pair of \(K_P\) and \(T_i\). Dashed line represents the phase error in degrees. It can be seen that procedure converges, but slowly and practically can’t be used successfully.

\[\text{Fig. 12. Tuning procedure: proportional gain (K}_P\text{), integration constant (T}_i\text{) and difference between actual and desired phase margin (f) for direct method}\]
In all presented simulations we used the same process:

\[
G_P = \frac{1}{(1+s)^2(1+2s)}
\]  

and desired amplitude and phase margin:

\[
A_m = 3 \\
\phi_m = 36^\circ
\]  

Proportional constant (\(K_P\)) at the beginning is 1 and integral constant (\(T_i\)) is 120s.

All used programs (in MATLAB) are printed in appendix.

2.2 Successive change of \(T_i\)

The method is relatively simple. At first we detect point C on Nyquist curve (see Fig. 10) and with \(K_P\) we can move it to point A (3). Then we detect point D. If \(\phi_a > \phi_m\) and there exist no previous \(T_i\) such that \(\phi_a < \phi_m\), then divide \(T_i\) by 2:

\[
T_i = \frac{T_i}{2}
\]  

If \(\phi_a < \phi_m\) and there exist no previous \(T_i\) such that \(\phi_a > \phi_m\), then multiply \(T_i\) by 2:

\[
T_i = 2T_i
\]  

In other case calculate \(T_i\) as:

\[
T_i = \sqrt{T_{i1}T_{i2}}
\]

where \(T_{i1}\) means the last \(T_i\) which caused \(\phi_a < \phi_m\) and \(T_{i2}\) is the last \(T_i\) when \(\phi_a > \phi_m\).

The result of presented method is shown in Fig. 13. We can see, this time algorithm converges faster than previous one, but still slow. It also needs some time to find appropriate range of \(T_i\) (from \(\approx 10^2\) s to about \(10^0\) s).
Fig. 13. Tuning procedure: proportional gain ($K_p$), integration constant ($T_i$) and difference between actual and desired phase margin ($f$) for successive change method

That leads us to modify the algorithm. The first step is to find point C and move it toward point A (see Fig. 10 and eq. 3), then (with new $K_p$) we find point D and calculate new $T_i$ using equation 6 in section 2.1. If new (calculated) $T_i$ is more than two times bigger or more than two times smaller than old one, new $T_i$ would be as calculated (6). Otherwise procedure would be the same as detected by equations 9 and 10.

To speed up the optimisation method when error of phase margin changes the sign, (11) have to be changed:

$$T_i = T_{i1} \left( \frac{T_{i2}}{T_{i1}} \right)^{\frac{\phi_1 - \phi_a}{\phi_1 - \phi_2}}$$

(12)

where $T_{i1}$ represents last $T_i$ for which $\phi_a > \phi_m$, $T_{i2}$ represents last $T_i$ for which $\phi_a < \phi_m$ and $\phi_1$ and $\phi_2$ represents $\phi_a$ obtained when using $T_{i1}$ and $T_{i2}$ respectively.

Results using such improved method is shown in Fig. 14. Improved method converges faster.
Fig. 14. Tuning procedure: proportional gain (\(K_P\)), integration constant (\(T_i\)) and difference between actual and desired phase margin (\(f\)) for improved successive change method

2.3 Correlation compensation method

The method is based on correlation between amplitude and phase margin. Change of amplitude margin when changing phase margin can be measured and vice versa. Then we can predict \(T_i\) for which both, amplitude and phase margin, will be fulfilled.

At first we determine \(K_P\) as shown in section 2.1 (see Fig. 10 and eq. 3). We could have the situation as shown with solid line in Fig. 15.

If we rotate point D for the angle \(\Delta \phi\), Nyquist curve intersects describing function at point D instead of B (dashed line). Actually, the curve rotates the angle \(\Delta \phi_1\):

\[
\Delta \phi_1 = k_1 \Delta \phi
\]

(13)

where \(k_1\) is a gain factor between \(\Delta \phi\) and \(\Delta \phi_1\):

\[
k_1 = \frac{\Delta \phi_1}{\Delta \phi}
\]

(14)
In Fig. 15 we can also see that change of $T_i$ also causes change of amplitude margin. It changes from $A_m$ to $A_1$:

$$A_1 = k_2 A_m \Delta \phi_1$$

(15)

where $k_2$ represents correlation factor from phase to amplitude margin:

$$k_2 = \frac{A}{A_m \Delta \phi_1}$$

(16)

To correct amplitude margin, we have to calculate $K_p$ again (see Fig. 10 and eq. 3). This correction changes phase margin (see Fig. 16).
Fig. 16. Correction of amplitude margin changes phase margin (lower Figure is magnified part of upper Nyquist diagram)

Intersection point with describing function changes from D₁ to D₂. Phase margin increases for the angle \( \Delta \phi_2 \):

\[
\Delta \phi_2 = k_3 \Delta \phi_1 \left( 1 - \frac{A}{A_m} \right)
\]  

(17)

where \( k_3 \) is a correlation factor from amplitude to phase margin:

\[
k_3 = \frac{\Delta \phi_1}{\Delta \phi_2} \left( 1 - \frac{A}{A_m} \right)^{-1}
\]  

(18)
If we want to correct the phase margin exactly from D to B, angles $\Delta \phi_1$ and $\Delta \phi_2$ have to be such that

$$
\Delta \phi_1 - \Delta \phi_2 = \phi_{DOB}
$$

(19)

where $\phi_{DOB}$ represents angle DOB (in Fig. 15 marked as $\Delta \phi$). From (13) to (18) we can calculate such $\Delta \phi$ which will satisfy (19):

$$
\Delta \phi = \frac{k_3 - 1 \pm \sqrt{(k_3 - 1)^2 + 4k_1k_3\phi_{DOB}}}{2k_1k_2k_3}
$$

(20)

From (6), if we substitute $\phi_a - \phi_m = \Delta \phi$, we can calculate and change $T_i$ and start again new iteration with determining $K_P$.

Results of described algorithm are shown in Fig. 17. We can see the method is the fastest one. Drawback of such method is, when we are close to desired amplitude and phase margin, $\Delta \phi$ and related $\Delta \phi_1$ and $\Delta \phi_2$ become small and factors $k_1$, $k_2$ and $k_3$ (14, 16 and 18) become inaccurate. Then we should stop this method and continue with e.g. successive change of $T_i$ method (chapter 2.2).

Fig. 17. Tuning procedure: proportional gain ($K_P$), integration constant ($T_i$) and difference between actual and desired phase margin ($f$) for the correlation compensation method
3. Detection of Nyquist curve by relay excitation

In previous chapter, some methods of tuning PI controller according to detected Nyquist points were discussed. Here, a procedure how to detect points on Nyquist curve by relay method (chapter 1) is presented.

3.1 Amplitude and phase computation

Limit cycle (oscillation) appears at the point where transfer function \( G_P(j\omega)G_C(j\omega) \) crosses describing function of relay \((-1/N(\alpha))\). From relay and process output (see Fig. 9) we can calculate amplitude and phase of our system \((G_P(j\omega)G_C(j\omega))\). Fig. 18 shows typical time response.

\[
\text{Fig. 18. Time response of relay output (} u_i(t) \text{) and process output (} y(t) \text{)}
\]

System input is therefore square wave signal which consists of main harmonic component at frequency \( \omega_0 = 2\pi/T_p \) and other higher harmonic components:

\[
u_i(t) = \frac{4d}{\pi} \sum_{n=0}^{\infty} \frac{1}{2n+1} \sin((2n+1)\omega_0 t) = \frac{4d}{\pi} \left[ \sin\omega_0 t + \frac{1}{3} \sin3\omega_0 t + \ldots \right]
\]

So, process output \((y)\) contains response on all harmonic components of the input signal. To detect the first (main) harmonic \((n=0)\), we have to use “filter” which is in fact Fourier transformation of \(y\) at the main frequency \((\omega_0)\):
\[\begin{align*}
A_{re} &= \frac{2}{T_P} \int_0^{T_P} y(t) \sin(\omega_0 t) dt \\
A_{im} &= \frac{2}{T_P} \int_0^{T_P} y(t) \cos(\omega_0 t) dt \\
A &= \sqrt{A_{re}^2 + A_{im}^2} \\
\phi &= \arctan \frac{A_{im}}{A_{re}}
\end{align*}\] (22a-b-c-d)

where \(A_{re}\) and \(A_{im}\) are real and imaginary component of amplitude respectively, \(A\) is an amplitude of the first harmonic and \(\phi\) is a phase shift of the first harmonic signal. To compute a gain of \(G_p(j\omega_0)G_c(j\omega_0)\), we have to divide \(A\) (22c) with the amplitude of the input signal. Amplitude and phase became:

\[\begin{align*}
A_a &= \frac{4d}{\pi A} \\
\phi_a &= \phi
\end{align*}\] (23a-b)

where in (22d) we used two quadrant function \(\text{atan}\). If we use function \(\text{atan}\) in all 4 quadrants (e.g. function \(\text{atan2}\) in MATLAB), we have to change (23b) into:

\[\phi_a = \pi + \phi\] (24)

Algorithm for detection amplitude and phase is digital, so we changed equations 22a and 22b into next form:

\[\begin{align*}
A_{re} &= \frac{T_s}{T_P} \sum_{k=0}^{n-1} \left[ y(k) \sin(kT_s\omega_0) + y(k+1) \sin\left((k+1)T_s\omega_0\right) \right] \\
A_{im} &= \frac{T_s}{T_P} \sum_{k=0}^{n-1} \left[ y(k) \cos(kT_s\omega_0) + y(k+1) \cos\left((k+1)T_s\omega_0\right) \right]
\end{align*}\] (25a-b)

\(T_s\) is sampling time, \(y(k)\) means k-th sample of \(y\) and \(y(n)\) represents the last sample \((y(T_P))\).

Actual phase margin can be calculated directly from detected amplitude margin if relay hysteresis is set as in (2):

\[\phi_a = \arcsin\left(A_a \sin \phi_m\right)\] (26)
To improve accuracy of the calculated phase margin, we can calculate it as a mean value of (22d) and (26):

$$\phi_a = \frac{\arctan \left( \frac{A_{im}}{A_{re}} \right) + \arcsin(A_a \sin \phi_m)}{2}$$

(27)

Fig. 20 shows tuning procedure when using relay excitation instead of Nyquist curve. We used improved successive change method (chapter 2.2). We can see, comparing with Fig. 14, the relay excitation method converges slower. The reason lays in errors when detecting characteristic Nyquist points.
3.2 Errors

Some error can appear when calculating $A_a$ and $\phi_a$ (23a and 27) because of time discretisation. If detected period from $t=0$ to $t=T_p$ consists of $n$ equidistant sampling intervals, then phase margin can not be detected more accurate than:

$$\Delta \phi = \pm\frac{360^\circ}{n}$$  \hspace{1cm} (28)

Period of the oscillation ($T_p$) can also be inaccurate in a range:

$$\Delta T_p = \pm2\frac{T_p}{n}$$  \hspace{1cm} (29)

what leads to inaccuracy of amplitude margin:
\[ \Delta A_a = \frac{A_a}{n} \] (30)

when \( n \) is relatively big.

Noise in the system can also have strong influence on result, specially when hysteresis of used relay is small or equal to zero. In that case we could add a filter at the relay input and leave some hysteresis. Then, from the first (main) harmonic and higher harmonics, we can find an approximation of the position of the point C (see fig. 10).
4. References


5. Appendix

Program in MATLAB for tuning with direct method (section 2.1)

```matlab
imag = sqrt(-1);
w = logspace (-2,1,200)+0.001;
fi=0:0.05*pi:2*pi;
re = cos(fi);
im = sin (fi);
points1 = re+imag*im;
fml = pi+fm;
rez = []; Kptmp = Kp; Titmp = Ti;

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
hold on;
plot(points);
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm1)+imag*sin(fm1) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);

w=w';
i=1;
while (im(i+1) < -sin(fm))
i = i+1;
end
rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(-rex1));

for l=1:12,
rez = [rez; Kp Ti (f02-fm)*180/pi];
[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
hold on;
plot(points);
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm1)+imag*sin(fm1) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);
i = 1;
while (im(i+1) < 0)
i = i+1;
end
rex2 = re(i)+(re(i+1)-re(i))*(im(i))/(im(i)-im(i+1));
w01 = w(i)+(w(i+1)-w(i))*(im(i))/(im(i)-im(i+1));
A01 = -rex2;
Kp = Kp/(Am*A01);
```
[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm)+imag*sin(fm) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);

w=w';
i = 1;
while (im(i+1) < -sin(fm))
  i = i+1;
end
rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(-rex1));
w02 = w(i)+(w(i+1)-w(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
A02 = sqrt(rex1^2 + (sin(fm))^2);

df = f02-fm;
Ti = (w02*tan(atan(1/(w02*Ti))+df))^(-1)
end
Kp = Kptmp;
Ti = Titmp;

subplot(211);
semilogy([0:11],rez(:,2))
grid
title('Ti [s]')

subplot(212)
plot([0:11],rez(:,1),[0:11],rez(:,3),'--')
grid
axis([0,12,-10,10])
title('__ Kp, -- f [deg]')
xlabel('Iterations')
Program in MATLAB for tuning with successive change method (section 2.2)

```matlab
imag = sqrt(-1);
w = logspace (-2,1,200)+0.001;
fi=0:0.05*pi:2*pi;
re = cos(fi);
im = sin (fi);
points1 = re+imag*im;
fm1 = pi+fm;
d = 0.1;
rez = [];
more = 0;
less = 0;
Kptmp = Kp;
Titmp = Ti;

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm1)+imag*sin(fm1) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);
w=w';
i=1;
while (im(i+1) < -sin(fm))
i = i+1;
end
rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(-rex1));
for l=1:12,
[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
i = 1;
while (im(i+1) < 0)
i = i+1;
end
rex2 = re(i)+(re(i+1)-re(i))*(im(i))/(im(i)-im(i+1));
w01 = w(i)+(w(i+1)-w(i))*(im(i))/(im(i)-im(i+1));
A01 = -rex2;
Kp = Kp/(Am*A01);

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm1)+imag*sin(fm1) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);
```
\[ w = w' \]
\[ i = 1; \]
\[ \text{while } (\im(i+1) < -\sin(fm)) \]
\[ \quad i = i+1; \]
\[ \text{end} \]
\[ \text{rex1} = r(e(i)+r(e(i+1)-r(e(i)))*(\sin(fm)+i(m(i))/(i(m(i)-i(m(i+1))))\];
\[ f02 = \text{atan}(\sin(fm)/(-\text{rex1})) \];
\[ w02 = w(i)+(w(i+1)-w(i))*\sin(fm)\}/\text{im(i)-(im(i)+1))\];
\[ \text{A02} = \sqrt{\text{rex1}^2 + (\sin(fm))^2}; \]
\[ \text{if } (f02 < fm) \]
\[ \quad \text{less} = Ti; \]
\[ \quad \text{if } (\text{more} == 0) \]
\[ \quad \quad Ti = 2*Ti; \]
\[ \quad \text{else} \]
\[ \quad \quad \text{Ti} = \sqrt{Ti*\text{more}}; \]
\[ \quad \text{end} \]
\[ \text{else} \]
\[ \quad \text{more} = Ti; \]
\[ \quad \text{if } (\text{less} == 0) \]
\[ \quad \quad Ti = Ti/2; \]
\[ \quad \text{else} \]
\[ \quad \quad \text{Ti} = \sqrt{Ti*\text{less}}; \]
\[ \quad \text{end} \]
\[ \text{end} \]
\[ \text{end} \]
\[ Kp = Kt\text{mp}; \]
\[ Ti = Tit\text{mp}; \]
\[ \text{subplot}(211); \]
\[ \text{semilogy}([0:11],rez(:,2)) \]
\[ \text{grid} \]
\[ \text{title}('Ti [s]') \]
\[ \text{subplot}(212) \]
\[ \text{plot}([0:11],rez(:,1),[0:11],rez(:,3),'--') \]
\[ \text{grid} \]
\[ \text{axis}([0,12,-10,10]) \]
\[ \text{title}('_Kp, -- f [deg]') \]
\[ \text{xlabel}('Iterations') \]
Program in MATLAB for tuning with *improved successive change method* (section 2.2)

```matlab
imag = sqrt(-1);
w = logspace (-2,1,200)+0.001;
fi=0:0.05*pi:2*pi;
re = cos(fi);
im = sin (fi);
points1 = re+imag*im;
fm1 = pi+fm;
d = 0.1;
rez = [];
more = 0;
less = 0;
moref = 0;
lessf = 0;
Kptmp = Kp;
Titmp = Ti;

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
polt([-1/Am+1.5*imag -1/Am-1.5*imag]);
polt([cos(fm1)+imag*sin(fm1) 0]);
polt([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);
w=w';
i=1;
while (im(i+1) < -sin(fm))
    i = i+1;
end
rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(-rex1));
for l=1:12,
    rez = [rez; Kp Ti (f02-fm)*180/pi];
    [re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
    points = re+imag*im;
    i = 1;
    while (im(i+1) < 0)
        i = i+1;
    end
    rex2 = re(i)+(re(i+1)-re(i))*(im(i))/(im(i)-im(i+1));
w01 = w(i)+(w(i+1)-w(i))*im(i)/(im(i)-im(i+1));
A01 = -rex2;
Kp = Kp/(Am*A01);

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
polt([-1/Am+1.5*imag -1/Am-1.5*imag]);
polt([cos(fm)+imag*sin(fm) 0]);
polt([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
```

pause (1);

w=w';
i = 1;
while (im(i+1) < -sin(fm))
i = i+1;
end
rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(rex1));
w02 = w(i)+(w(i+1)-w(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
A02 = sqrt(rex1^2 + (sin(fm))^2);

Tix = 1/(w02*tan(atan(1/(w02*Ti))+f02-fm));
if ((Tix/Ti > 2) | (Tix/Ti < 0.5))
    Ti = Tix;
else
    if (f02 < fm)
        less = Ti;
        lessf = f02;
        if (more == 0)
            Ti = 2*Ti;
        else
            a = Ti/more;
            Ti = more*a^((more-fm)/(more-f02));
        end
    else
        more = Ti;
        moref = f02;
        if (less == 0)
            Ti = Ti/2;
        else
            a = less/Ti;
            Ti = Ti*a^((f02-fm)/(f02-lessf));
        end
    end
end

Kp = Kptmp;
Ti = Titmp;
subplot(211);
semilogy([0:11],rez(:,2))
grid
title('Ti [s]')
subplot(212)
plot([0:11],rez(:,1),[0:11],rez(:,3),'--')
grid
axis([0,12,-10,10])
title('___ Kp, -- f [deg]')
xlabel('Iterations')
Program in MATLAB for tuning with correlation compensation method (section 2.3)

```matlab
imag = sqrt(-1);
w = logspace(-2,1,200)+0.001;
fi=0.05*pi:2*pi;
re = cos(fi);
im = sin (fi);
points1 = re+imag*im;
fm1 = pi+fm;
d = 0.1;
rez = [];
Kptmp = Kp;
Titmp = Ti;

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm1)+imag*sin(fm1) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);

w=w';
i=1;
while (im(i+1) < -sin(fm))
    i = i+1;
end
rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(-rex1));
rez = [rez; Kp Titmp (f02-fm)*180/pi];
[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm1)+imag*sin(fm1) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);

i=1;
while (im(i+1) < 0)
    i = i+1;
end
rex2 = re(i)+(re(i+1)-re(i))*(im(i))/(im(i)-im(i+1));
w01 = w(i)+(w(i+1)-w(i))*(im(i))/(im(i)-im(i+1));
A01 = -rex2;
Kp01 = Kp/(Am*A01);
Kp = Kp01;

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
```

plot({cos(fm1)+imag*sin(fm1) 0});
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);

w=w';
i = 1;
while (im(i+1) < -sin(fm))
i = i+1;
end

rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(rex1));
w02 = w(i)+(w(i+1)-w(i))*{sin(fm)+im(i)}/(im(i)-im(i+1));
A02 = sqrt(rex1^2 + (sin(fm))^2);

fx = f02-fm;
df = fx;

Ti = (w02*tan(atan(1/(w02*Ti)))+(f02-fm))^(-1);

for l=1:11,

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([cos(fm1)+imag*sin(fm1) 0]);
plot([-1.5-imag*sin(fm) 1.5-imag*sin(fm)]);
axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);

w=w';
i = 1;
while (im(i+1) < -sin(fm))
i = i+1;
end

rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(rex1));
w02 = w(i)+(w(i+1)-w(i))*{sin(fm)+im(i)}/(im(i)-im(i+1));
A02 = sqrt(rex1^2 + (sin(fm))^2);

fx = f02-fm;
df = fx;

Ti = (w02*tan(atan(1/(w02*Ti)))+(f02-fm))^(-1);
while (im(i+1) < 0)
i = i+1;
end
rex2 = re(i)+(re(i+1)-re(i))*(im(i))/(im(i)-im(i+1));
w01 = w(i)+(w(i+1)-w(i))*(im(i))/(im(i)-im(i+1));
A01 = -rex2;

k2 = A01/(Am*df1);
Kp01 = Kp/(Am*A01);
Kp = Kp01;

[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/Am+1.5*imag -1/Am-1.5*imag]);
plot([-1.5*imag*sin(fm) 1.5*imag*sin(fm)]); axis([-1.5 1.5 -1.5 1.5]);
grid
hold off;
pause (1);
w=w';
i = 1;
while (im(i+1) < -sin(fm))
i = i+1;
end
rex1 = re(i)+(re(i+1)-re(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
f02 = atan(sin(fm)/(-rex1));
w02 = w(i)+(w(i+1)-w(i))*(sin(fm)+im(i))/(im(i)-im(i+1));
A02 = sqrt(rex1^2 + (sin(fm))^2);
rez = [rez; Kp Ti (f02-fm)*180/pi];
fx = f02-fm;
df2 = df-df1-fx;
k3 = df2/df1*(1-A02/Am)^(-1);
dfo1 = 1/(2*k1*k2*k3)*(1+k3+sqrt((1+k3)^2-4*k2*k3*fx))
df02 = 1/(2*k1*k2*k3)*(1+k3-sqrt((1+k3)^2-4*k2*k3*fx))

if (fx > 0)
if (df02 < df01)
df = df01;
else
df = df02;
end
else
if (df02 < df01)
df = df02;
else
df = df01;
end
end

if (abs(df01) == abs(df02))
break;
else
Ti = (w02*tan(atan(1/(w02*Ti))+df))^(1);
end
if (Ti < 0)
Ti = 0.1;
end
end
Kp = Kptmp;
Ti = Titmp;

subplot(211);
semilogy([0:11],rez(:,2))
grid
title('Ti [s]')
subplot(212)
plot([0:11],rez(:,1),[0:11],rez(:,3),'-')
grid
axis([0,12,-10,10])
title('Kp, -- f [deg]')
xlabel('Iterations')
Program in MATLAB for tuning with relay excitation (section 3.1)

\[
\text{imag} = \sqrt{-1};
\]
\[
w = \logspace(-2,1,100)+0.001;
\]
\[
\text{fi} = 0:0.05\pi:2\pi;
\]
\[
\text{re} = \cos(\text{fi});
\]
\[
\text{im} = \sin(\text{fi});
\]
\[
\text{points1} = \text{re}+\text{imag}^*\text{im};
\]
\[
\text{fm} = \pi+\text{fm};
\]
\[
\text{d} = 0.1;
\]
\[
\text{rez} = [];
\]
\[
\text{more} = 0;
\]
\[
\text{less} = 0;
\]
\[
\text{Kptmp} = \text{Kp};
\]
\[
\text{Ttmp} = \text{Ti};
\]
\[
[\text{re},\text{im}] = \text{nyquist}([\text{Kp}\times\text{Ti} \ \text{Kp}], [2\times\text{Ti} \ 5\times\text{Ti} \ 4\times\text{Ti} \ \text{Ti} \ 0], w);
\]
\[
\text{points} = \text{re}+\text{imag}^*\text{im};
\]
\[
\text{plot}(\text{points});
\]
\[
\text{hold on};
\]
\[
\text{plot}(\text{points1});
\]
\[
\text{plot}([-1/\text{Am}+1.5*\text{imag} -1/\text{Am}-1.5*\text{imag}]);
\]
\[
\text{plot}([\cos(\text{fm})+\text{imag}^*\sin(\text{fm}) 0]);
\]
\[
\text{plot}([-1.5-\text{imag}^*\sin(\text{fm}) 1.5-\text{imag}^*\sin(\text{fm})]);
\]
\[
\text{axis([-1.5 1.5 -1.5 1.5])};
\]
\[
\text{grid}
\]
\[
\text{hold off};
\]
\[
\text{pause}(1);
\]
\[
\text{w} = \text{w'};
\]
\[
i = 1;
\]
\[
\text{while } (\text{im}(i+1) < -\sin(\text{fm}))
\]
\[
i = i+1;
\]
\[
\text{end}
\]
\[
\text{rex1} = \text{re}(i)+(\text{re}(i+1)-\text{re}(i))*(\sin(\text{fm})+\text{im}(i))/(\text{im}(i)-\text{im}(i+1));
\]
\[
\text{f02} = \text{atan}(\sin(\text{fm})/(-\text{rex1}));
\]
\[
\text{for } i = 1:12,
\]
\[
\text{rez} = [\text{rez}; \text{Kp} \times \text{Ti} \ (\text{f02}-\text{fm})*180/\pi];
\]
\[
\text{eps} = 0;
\]
\[
[\text{t},\text{x},\text{y}] = \text{gear('rele',60,[],[1e-3,1e-5,0.01,0,3,0])};
\]
\[
\text{a} = \text{size}(\text{yout},1);
\]
\[
\text{j} = \text{a};
\]
\[
\text{while } ((\text{yout}(j,3) \ < \ 0) \ | \ (\text{yout}(j-1,3) \ > \ 0))
\]
\[
j = j-1;
\]
\[
\text{k} = j;
\]
\[
\text{t2} = \text{yout}(k,1);
\]
\[
\text{j} = k-1;
\]
\[
\text{while } ((\text{yout}(j,3) \ < \ 0) \ | \ (\text{yout}(j-1,3) \ > \ 0))
\]
\[
j = j-1;
\]
\[
\text{l} = j;
\]
\[
\text{t1} = \text{yout}(l,1);
\]
\[
\text{tp} = \text{t2}-\text{t1};
\]
\[
\text{w0} = 2*\pi/\text{tp};
\]
\[
\text{t} = \text{yout}(1:k,1)-\text{t1};
\]
\[
\text{y} = \text{yout}(1:k,2);
\]
\[
\text{rea} = 0;
ima = 0;
for j = 1:k-1,
    rea = rea + 0.5*(y(j)*sin(w0*t(j))+y(j+1)*sin(w0*t(j+1)))*(t(j+1)-t(j));
    ima = ima + 0.5*(y(j)*cos(w0*t(j))+y(j+1)*cos(w0*t(j+1)))*(t(j+1)-t(j));
end
rea=rea*2/tp;
ima=ima*2/tp;
tp1vect(i) = tp;
A0 = sqrt(rea^2+ima^2);
A01vect(i) = A0;
Kp = Kp*4*d/(A0*A0*pi);
Kpvect(i) = Kp;
[re,im] = nyquist([Kp*Ti Kp],[2*Ti 5*Ti 4*Ti Ti 0],w);
points = re+imag*im;
plot(points);
hold on;
plot(points1);
plot([-1/A0+imag -1/A0-imag]);
plot([cos(fm)+imag*sin(fm)*0]);
axis([-1 1 -1 1]);
grid
hold off;
pause (1);

eps = 4*d*sin(fm)/pi;
[t,x,y] = gear ('rele',60,[],[1e-3,1e-5,0.01,0,3,0]);
a = size(yout,1);
j = a;
while ((yout(j,3) < 0) | (yout(j-1,3) > 0))
    j = j-1;
end
k = j;
t2 = yout(k,1);
j = k-1;
while ((yout(j,3) < 0) | (yout(j-1,3) > 0))
    j = j-1;
end
l = j;
t1 = yout(l,1);
tp = t2-t1;
tp2vect(i) = tp;
w0 = 2*pi/tp;
t = yout(l:k,1)-t1;
y = yout(l:k,2);
rea = 0;
ima = 0;
for j = 1:k-1,
    rea = rea + 0.5*(y(j)*sin(w0*t(j))+y(j+1)*sin(w0*t(j+1)))*(t(j+1)-t(j));
    ima = ima + 0.5*(y(j)*cos(w0*t(j))+y(j+1)*cos(w0*t(j+1)))*(t(j+1)-t(j));
end
rea=rea*2/tp;
ima=ima*2/tp;
tp1vect(i) = tp;
A0 = sqrt(rea^2+ima^2);
\[ f_2 = \arctan\left(\frac{\text{ima}}{\text{rea}}\right); \]
\[ f_{21} = \arcsin\left(\frac{\sin(f_m)\cdot 4\cdot d/A_0}{\pi}\right); \]
\[ f_2 = \frac{f_2 + f_{21}}{2}; \]
\[ f_{2\text{vect}}(i) = f_2; \]
\[ f_{2\text{nn}} = \frac{\pi}{2} - \arcsin\left(\frac{\sin(f_m)}{\tan(f_2 - \pi)}\right); \]
\[ w_2 = \frac{2\pi}{\text{tp}}; \]
\[ T_{ix} = \frac{1}{(w_2^* \tan(\arctan(1/(w_2^* T_i)) + f_2 - f_m))}; \]
\[ \text{if } ((T_{ix}/T_i > 2) \text{ | } (T_{ix}/T_i < 0.5)) \]
\[ T_i = T_{ix}; \]
\[ \text{else} \]
\[ \text{if } (f_2 < f_m) \]
\[ \text{less} = T_i; \]
\[ \text{lessf} = f_2; \]
\[ \text{if } (\text{more} == 0) \]
\[ T_i = 2\cdot T_i; \]
\[ \text{else} \]
\[ a_1 = T_i/\text{more}; \]
\[ T_i = \text{more} \cdot a_1^\left((\text{more}-f_m)/(\text{more}-f_2)\right); \]
\[ \text{end} \]
\[ \text{else} \]
\[ \text{more} = T_i; \]
\[ \text{moref} = f_2; \]
\[ \text{if } (\text{less} == 0) \]
\[ T_i = T_i/2; \]
\[ \text{else} \]
\[ a_1 = \text{less}/T_i; \]
\[ T_i = T_i \cdot a_1^\left((f_2-f_m)/(f_2-\text{lessf})\right); \]
\[ \text{end} \]
\[ \text{end} \]
\[ [\text{re}, \text{im}] = \text{nyquist}([Kp \cdot T_i \ Kp],[2 \cdot T_i \ 5 \cdot T_i \ 4 \cdot T_i \ 4 \cdot T_i \ 0],w); \]
\[ \text{points} = \text{re} + \text{imag} \cdot \text{im}; \]
\[ \text{plot}(\text{points}); \]
\[ \text{hold on}; \]
\[ \text{plot}(\text{points1}); \]
\[ \text{plot}([-1/\text{Am}+1.5*\text{imag} -1/\text{Am}-1.5*\text{imag}]); \]
\[ \text{plot}([\cos(f_m)+\text{imag} \cdot \sin(f_m) \ 0]); \]
\[ \text{plot}([-1.5-\text{imag} \cdot \sin(f_m) \ 1.5-\text{imag} \cdot \sin(f_m)]); \]
\[ \text{axis}([-1.5 \ 1.5 \ -1.5 \ 1.5]); \]
\[ \text{grid} \]
\[ \text{hold off}; \]
\[ \text{pause}(1); \]
\[ w=w'; \]
\[ i = 1; \]
\[ \text{while } (\text{im}(i+1) < -\sin(f_m)) \]
\[ i = i+1; \]
\[ \text{end} \]
\[ r_{ex1} = \text{re}(i)+(\text{re}(i+1) - \text{re}(i)) \cdot (\sin(f_m) + \text{im}(i))/(\text{im}(i) - \text{im}(i+1)); \]
\[ f_{02} = \arctan(\sin(f_m)/(r_{ex1})); \]
\[ \text{end} \]
\[ K_p = K_{p\text{tmp}}; \]
\[ T_i = T_{i\text{tmp}}; \]
\[ \text{subplot}(211); \]
\[ \text{semilogy}([0:11], \text{rez}((:,2))); \]
\[ \text{grid} \]
where RELE.M represents next scheme in SIMULINK: