TAMA Modecleaner alignment
Version 1.6

G. Heinzel, K. Arai, NAO Mitaka

March 13, 2001

1 Introduction

The effects of mirror misorientations[1] in interferometers with three or more mirrors cannot easily be determined analytically in the general case. Therefore a MATHEMATICA program for numerical 3-D ray tracing was written. It uses geometrical optics to find the axis of the eigenmode for a given combination of reflecting and/or refracting plane and spherical surfaces. A similar program is described and printed in Appendix E3 of MPQ243. The term 'beam' in this text refers to the geometrical axis of a beam, without taking into account the transverse shape or optical phase.

2 The modecleaner cavities

Figure 1 shows a schematic view of the TAMA modecleaner cavity seen from above, together with the coordinate system adopted in this section.

The cavity consists of two flat mirrors ($\mathrm{M_a}$ and $\mathrm{M_b}$) that are separated by a relatively short distance (20 cm), and a curved mirror $\mathrm{M_c}$ with radius of curvature $R = 15\,\mathrm{m}$, which is at a distance $L = 9.738\,\mathrm{m}$ from the (center between the) flat mirrors. The beam enters through $\mathrm{M_a}$ and travels clockwise to $\mathrm{M_b}$, $\mathrm{M_c}$, $\mathrm{M_a}$, etc.

$\mathrm{M_e}$ is the location of the photodetector behind the end mirror (in the following called 'end detector' for short), which is at a distance of 2.08 m behind mirror $\mathrm{M_c}$.

There are four beams of interest leaving the cavity. The main output beam is '$\mathrm{Out_b}$' which goes to the interferometer and the stabilization of which is the main purpose of the modecleaner. There are two beams coming from $\mathrm{M_a}$: the directly reflected input beam and the beam '$\mathrm{Out_a}$' which is a fraction of the cavity eigenmode. The cavity is well aligned to the incoming beam (which we consider fixed), if these two beams are perfectly superimposed. By taking two quadrant diodes with two different lens systems and appropriately demodulating their outputs, we can obtain four independent error signals, similar to the case of a simple two-mirror Fabry-Perot cavity. The longitudinal locking signal is also obtained from these two interfering beams (using the Pound-Drever-Hall scheme).

[1] To avoid confusion, we call a mirror or other component **misoriented** in this section, if its angular orientation differs from its reference orientation. The resulting movement of beams (e.g. cavity eigenmodes) will be called **misalignment**. An interferometer is called **well-aligned** if there are neither misorientations nor misalignments, i.e. components as well as beams are in their reference positions.

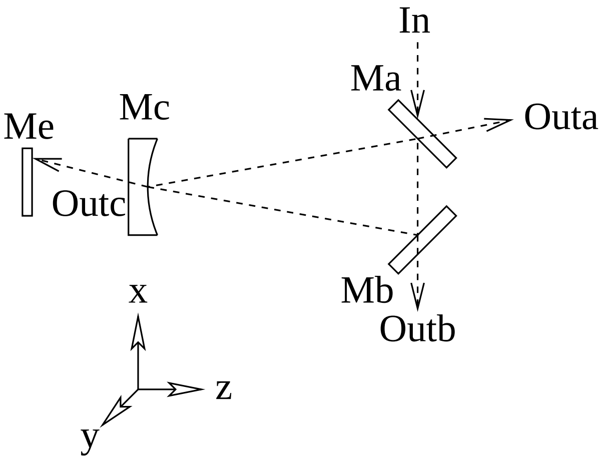


Figure 1: Schematic diagram of the TAMA modecleaner cavity seen from above.

In the three-mirror cavity, there are two additional degrees of freedom. They are linear combinations of movements of all three mirrors, as computed below. We will call them “neutral modes” because they have no influence on either the interference of the incoming beam with the cavity eigenmode nor on the outgoing beam. They can be used to control the spot position on the far mirror $\mathrm{M_c}$. This is described in more detail below.

In the ray-tracing program, we first compute the well-aligned case (i.e., all mirrors are hit in their center) as reference. We call $\mathrm{P_a}$, $\mathrm{P_b}$ and $\mathrm{P_c}$ the points where the axis of the eigenmode intersects the mirrors $\mathrm{M_a}$, $\mathrm{M_b}$ and $\mathrm{M_c}$, respectively. After misorienting one particular mirror by the small angle ε, we recompute the eigenmode axis, compare it with the well-aligned case and divide the difference by ε. The main results are:

- The shifts of the spots $\mathrm{P_a}$, $\mathrm{P_b}$ and $\mathrm{P_c}$.
- The angles γ_a, γ_b and γ_c between the beams ‘$\mathrm{Out_a}$’, ‘$\mathrm{Out_b}$’, ‘$\mathrm{Out_c}$’ and their respective references. For the vertical misalignments which are considered separately, we call these angles δ_a, δ_b and δ_c, respectively. We also compute the angle γ_d (δ_d) between the directly reflected incoming beam and its reference for the case that $\mathrm{M_a}$ is misoriented.
- For ‘$\mathrm{Out_a}$’ in the case of misorienting $\mathrm{M_a}$, we also compute the angle $\gamma'_\mathrm{a} = \gamma_\mathrm{a} - \gamma_\mathrm{d}$, which is the angle between the beam leaving the cavity and the directly reflected beam, because this is the angle between the interfering wavefronts that is detected by the quadrant diode.
- As described above we finally compute the angle θ which describes the ‘character’ of the misalignment at the waist. It is given by $\theta^\mathrm{w} = \arctan(\gamma'_\mathrm{a} z_\mathrm{R}/\Delta z_\mathrm{waist})$.

The waist of the cavity eigenmode is located halfway between the mirrors $\mathrm{M_a}$ and

$\mathrm{M_b}$. Its Rayleigh range is given by

$$z_\mathrm{R} = \sqrt{\frac{L_\mathrm{RT}}{2}\left(R - \frac{L_\mathrm{RT}}{2}\right)} = 7.126\,\mathrm{m}, \tag{1}$$

where $L_\mathrm{RT}/2$ is one half of the round-trip distance:

$$L_\mathrm{RT}/2 = \sqrt{L^2 + d^2/4} + d/2 = 9.8385\,\mathrm{m}. \tag{2}$$

3 Horizontal misalignments:

By ‘horizontal’ misalignments we mean that a mirror is rotated around the vertical axis, i.e. beam spots move horizontally (in the plane of the modecleaner cavity). Angles are counted as positive if a mirror is rotated *clockwise*, if seen from above (as in Figure 1).

We introduce the common- and differential mode motion of mirrors $\mathrm{M_a}$ and $\mathrm{M_b}$ by defining angles α_+ and α_-. Furthermore we introduce the “neutral” mode α_n as follows:

$$\begin{array}{c|c|c|c}
 & \mathrm{M_a} & \mathrm{M_b} & \mathrm{M_c} \\
\hline
\alpha_+ & \alpha_\mathrm{a} = \alpha_+ & \alpha_\mathrm{b} = \alpha_+ & \alpha_\mathrm{c} = 0 \\
\alpha_- & \alpha_\mathrm{a} = \alpha_- & \alpha_\mathrm{b} = -\alpha_- & \alpha_\mathrm{c} = 0 \\
\alpha_\mathrm{n} & \alpha_\mathrm{a} = \alpha_\mathrm{n} & \alpha_\mathrm{b} = \alpha_\mathrm{n} & \alpha_\mathrm{c} = 0.701\alpha_\mathrm{n}
\end{array} \tag{3}$$

$$\alpha_+ = \frac{\alpha_\mathrm{a} + \alpha_\mathrm{b}}{2}, \tag{4}$$

$$\alpha_- = \frac{\alpha_\mathrm{a} - \alpha_\mathrm{b}}{2}, \tag{5}$$

The results of the raytracing program are given in Table 1.

Cause	$\mathrm{P_a}$ Δx	$\mathrm{P_a}$ Δz	waist Δz	$\mathrm{Out_a}$ γ_a	$\mathrm{Out'_a}$ $\gamma'_\mathrm{a} = \gamma_\mathrm{a} - \gamma_\mathrm{d}$	θ^w
α_a	$-9.740\,\mathrm{m}\cdot\alpha_\mathrm{a}$	$9.840\,\mathrm{m}\cdot\alpha_\mathrm{a}$	$9.739\,\mathrm{m}\cdot\alpha_\mathrm{a}$	$0.981\,\alpha_\mathrm{a}$	$-1.019\,\alpha_\mathrm{a}$	$-36.72°$
α_b	$9.538\,\mathrm{m}\cdot\alpha_\mathrm{b}$	$-9.636\,\mathrm{m}\cdot\alpha_\mathrm{b}$	$-9.739\,\mathrm{m}\cdot\alpha_\mathrm{b}$	$-1.019\,\alpha_\mathrm{b}$	$-1.019\,\alpha_\mathrm{b}$	$36.72°$
α_c	$0.288\,\mathrm{m}\cdot\alpha_\mathrm{c}$	$-0.291\,\mathrm{m}\cdot\alpha_\mathrm{c}$	$0.000\,\mathrm{m}\cdot\alpha_\mathrm{c}$	$2.906\,\alpha_\mathrm{c}$	$2.906\,\alpha_\mathrm{c}$	$-90.00°$
α_+	$-0.202\,\mathrm{m}\cdot\alpha_+$	$0.204\,\mathrm{m}\cdot\alpha_+$	$0.000\,\mathrm{m}\cdot\alpha_+$	$-0.039\,\alpha_+$	$-2.039\,\alpha_+$	$90.00°$
α_-	$-19.278\,\mathrm{m}\cdot\alpha_-$	$19.477\,\mathrm{m}\cdot\alpha_-$	$19.477\,\mathrm{m}\cdot\alpha_-$	$2.000\,\alpha_-$	$0.000\,\alpha_-$	$0.00°$
α_n	$0.000\,\mathrm{m}\cdot\alpha_\mathrm{n}$	$0.000\,\mathrm{m}\cdot\alpha_\mathrm{n}$	$0.000\,\mathrm{m}\cdot\alpha_\mathrm{n}$	$2.000\,\alpha_\mathrm{n}$	$0.000\,\alpha_\mathrm{n}$	—

Cause	$\mathrm{P_b}$ Δx	$\mathrm{P_b}$ Δz	$\mathrm{Out_b}$ γ_b	$\mathrm{P_c}$ Δx	$\mathrm{Out_c}$ γ_c	d. refl. γ_d	End Δx
α_a	$9.538\,\mathrm{m}\cdot\alpha_\mathrm{a}$	$9.637\,\mathrm{m}\cdot\alpha_\mathrm{a}$	$1.019\,\alpha_\mathrm{a}$	$-0.291\,\mathrm{m}\cdot\alpha_\mathrm{a}$	$-1.019\,\alpha_\mathrm{a}$	$2\cdot\alpha_\mathrm{a}$	$-2.411\,\mathrm{m}\cdot\alpha_\mathrm{a}$
α_b	$-9.740\,\mathrm{m}\cdot\alpha_\mathrm{b}$	$-9.841\,\mathrm{m}\cdot\alpha_\mathrm{b}$	$1.019\,\alpha_\mathrm{b}$	$-0.291\,\mathrm{m}\cdot\alpha_\mathrm{b}$	$0.981\,\alpha_\mathrm{b}$	$0\cdot\alpha_\mathrm{b}$	$1.749\,\mathrm{m}\cdot\alpha_\mathrm{b}$
α_c	$0.288\,\mathrm{m}\cdot\alpha_\mathrm{c}$	$0.291\,\mathrm{m}\cdot\alpha_\mathrm{c}$	$-2.906\,\alpha_\mathrm{c}$	$28.596\,\mathrm{m}\cdot\alpha_\mathrm{c}$	$2.906\,\alpha_\mathrm{c}$	$0\cdot\alpha_\mathrm{c}$	$34.642\,\mathrm{m}\cdot\alpha_\mathrm{c}$
α_+	$-0.202\,\mathrm{m}\cdot\alpha_+$	$-0.204\,\mathrm{m}\cdot\alpha_+$	$2.039\,\alpha_+$	$-0.581\,\mathrm{m}\cdot\alpha_+$	$-0.039\,\alpha_+$	$2\cdot\alpha_+$	$-0.662\,\mathrm{m}\cdot\alpha_+$
α_-	$19.278\,\mathrm{m}\cdot\alpha_-$	$19.477\,\mathrm{m}\cdot\alpha_-$	$0.000\,\alpha_-$	$0.000\,\mathrm{m}\cdot\alpha_-$	$-2.000\,\alpha_-$	$2\cdot\alpha_-$	$-4.160\,\mathrm{m}\cdot\alpha_-$
α_n	$0.000\,\mathrm{m}\cdot\alpha_\mathrm{n}$	$0.000\,\mathrm{m}\cdot\alpha_\mathrm{n}$	$0.000\,\alpha_\mathrm{n}$	$19.478\,\mathrm{m}\cdot\alpha_\mathrm{n}$	$2.000\,\alpha_\mathrm{n}$	$2\cdot\alpha_\mathrm{n}$	$23.639\,\mathrm{m}\cdot\alpha_\mathrm{n}$

Table 1: Results of the ray-tracing program for horizontal misalignments of the TAMA modecleaner.

4 Vertical misalignments:

In the modecleaners, the horizontal and vertical axes are *not* similar. The results of the raytracing program for vertical misalignments are given in Table 2. Note, for example, that a small vertical tilt β_{a} of the input mirror $\mathrm{M_a}$ (which is hit under approximately 45° from the incoming beam) causes a deflection of the reflected beam by only $\delta_{\mathrm{d}} = 1.43\beta_{\mathrm{a}}$ as compared to $\gamma_{\mathrm{d}} = 2\alpha_{\mathrm{a}}$ in the horizontal case. Another example is the tilt of $\mathrm{M_c}$ which, if horizontal, causes a pure angular misalignment at the waist. A vertical tilt of $\mathrm{M_c}$, on the other hand, shifts the cavity eigenmode downwards parallelely, without changing any angles. Angles are now counted as positive when the mirror normal moves downward from the reference direction.

As before we introduce the common- and differential mode motion of mirrors $\mathrm{M_a}$ and $\mathrm{M_b}$ by defining angles β_+ and β_-. Furthermore we introduce the “neutral” mode β_{n} as follows:

$$\begin{array}{c|c|c|c}
 & \mathrm{M_a} & \mathrm{M_b} & \mathrm{M_c} \\
\hline
\beta_+ & \beta_{\mathrm{a}} = \beta_+ & \beta_{\mathrm{b}} = \beta_+ & \beta_{\mathrm{c}} = 0 \\
\beta_- & \beta_{\mathrm{a}} = \beta_- & \beta_{\mathrm{b}} = -\beta_- & \beta_{\mathrm{c}} = 0 \\
\beta_{\mathrm{n}} & \beta_{\mathrm{a}} = \beta_{\mathrm{n}} & \beta_{\mathrm{b}} = \beta_{\mathrm{n}} & \beta_{\mathrm{c}} = -0.4986\beta_{\mathrm{n}}
\end{array} \tag{6}$$

$$\beta_+ = \frac{\beta_{\mathrm{a}} + \beta_{\mathrm{b}}}{2}, \tag{7}$$

$$\beta_- = \frac{\beta_{\mathrm{a}} - \beta_{\mathrm{b}}}{2}, \tag{8}$$

The results of the raytracing program are given in Table 2.

5 Degrees of freedom

The most important alignment task is to superimpose the axis of the cavity eigenmode with the axis of the incoming beam. This requires the control of four degrees of freedom. For this purpose, in the differential wavefront sensing method, we place two quadrant detectors with different lens systems in the beam reflected from $\mathrm{M_a}$. The interference between the directly reflected incoming beam, which is phase modulated at an RF frequency, and the beam ‘$\mathrm{Out_a}$’ leaking out of the cavity contains enough information to lock the cavity longitudinally and to obtain alignment error signals for those four degrees of freedom that determine the superposition of the incoming beam and the cavity eigenmode.

In particular, we now assume all mirrors to be slightly misoriented and compute the combined signals which are obtained by demodulating the outputs of two quadrant detectors, one (called X_I) with $\Phi = 0°$ and the other one (called X_Q) with $\Phi = 90°$ of extra phase shift. We scale parallel shifts Δy or Δz with the appropriate factor z_{R} and obtain for horizontal misalignments:

$$X_I = -1.019\,\alpha_{\mathrm{a}} - 1.019\,\alpha_{\mathrm{b}} + 2.906\,\alpha_{\mathrm{c}} = -1.019\,\alpha_+ + 2.906\,\alpha_{\mathrm{c}}, \tag{9}$$

$$X_Q = 1.366\,\alpha_{\mathrm{a}} - 1.366\,\alpha_{\mathrm{b}} = 1.366\,\alpha_-, \tag{10}$$

Cause	$\mathrm{P_a}$	waist	$\mathrm{Out_a}$	$\mathrm{Out'_a}$	θ^{w}
	Δy	Δy	δ_a	$\delta'_\mathrm{a} = \delta_\mathrm{a} - \delta_\mathrm{d}$	
β_a	$-3.670\,\mathrm{m}\cdot\beta_\mathrm{a}$	$-3.740\,\mathrm{m}\cdot\beta_\mathrm{a}$	$-0.718\,\beta_\mathrm{a}$	$0.704\,\beta_\mathrm{a}$	$-53.27°$
β_b	$-3.810\,\mathrm{m}\cdot\beta_\mathrm{b}$	$-3.740\,\mathrm{m}\cdot\beta_\mathrm{b}$	$-0.704\,\beta_\mathrm{b}$	$-0.704\,\beta_\mathrm{b}$	$53.27°$
β_c	$-15.000\,\mathrm{m}\cdot\beta_\mathrm{c}$	$-15.000\,\mathrm{m}\cdot\beta_\mathrm{c}$	$0.000\,\beta_\mathrm{c}$	$0.000\,\beta_\mathrm{c}$	$0.0°$
β_+	$-7.480\,\mathrm{m}\cdot\beta_+$	$-7.480\,\mathrm{m}\cdot\beta_+$	$-1.421\,\beta_+$	$0.000\,\beta_+$	$0.0°$
β_-	$0.141\,\mathrm{m}\cdot\beta_-$	$0.000\,\mathrm{m}\cdot\beta_-$	$-0.014\,\beta_-$	$1.407\,\beta_-$	$90.0°$
β_n	$0.000\,\mathrm{m}\cdot\beta_\mathrm{n}$	$0.000\,\mathrm{m}\cdot\beta_\mathrm{n}$	$-1.421\,\beta_\mathrm{n}$	$0.000\,\beta_\mathrm{n}$	—

Cause	$\mathrm{P_b}$	$\mathrm{Out_b}$	$\mathrm{P_c}$	$\mathrm{Out_c}$	d. refl.	End
	Δy	δ_b	Δy	δ_c	δ_d	Δy
β_a	$-3.810\,\mathrm{m}\cdot\beta_\mathrm{a}$	$0.704\,\beta_\mathrm{a}$	$-10.661\,\mathrm{m}\cdot\beta_\mathrm{a}$	$0.704\,\beta_\mathrm{a}$	$1.42\cdot\beta_\mathrm{a}$	$-12.125\,\mathrm{m}\cdot\beta_\mathrm{a}$
β_b	$-3.670\,\mathrm{m}\cdot\beta_\mathrm{b}$	$-0.704\,\beta_\mathrm{b}$	$-10.661\,\mathrm{m}\cdot\beta_\mathrm{b}$	$0.718\,\beta_\mathrm{b}$	$0.00\cdot\beta_\mathrm{b}$	$-12.155\,\mathrm{m}\cdot\beta_\mathrm{b}$
β_c	$-15.000\,\mathrm{m}\cdot\beta_\mathrm{c}$	$0.000\,\beta_\mathrm{c}$	$-15.000\,\mathrm{m}\cdot\beta_\mathrm{c}$	$0.000\,\beta_\mathrm{c}$	$0\cdot\beta_\mathrm{c}$	$-15.000\,\mathrm{m}\cdot\beta_\mathrm{c}$
β_+	$-7.480\,\mathrm{m}\cdot\beta_+$	$0.000\,\beta_+$	$-21.323\,\mathrm{m}\cdot\beta_+$	$1.421\,\beta_+$	$-1.421\cdot\beta_+$	$-24.280\,\mathrm{m}\cdot\beta_+$
β_-	$-0.141\,\mathrm{m}\cdot\beta_-$	$1.407\,\beta_-$	$0.000\,\mathrm{m}\cdot\beta_-$	$-0.014\,\beta_-$	$-1.421\cdot\beta_-$	$0.030\,\mathrm{m}\cdot\beta_-$
β_n	$0.000\,\mathrm{m}\cdot\beta_\mathrm{n}$	$0.000\,\beta_\mathrm{n}$	$-13.843\,\mathrm{m}\cdot\beta_\mathrm{n}$	$1.421\,\beta_\mathrm{n}$	$-1.420\cdot\beta_\mathrm{n}$	$-16.800\,\mathrm{m}\cdot\beta_\mathrm{n}$

Table 2: Results of the ray-tracing program for vertical misalignments of the TAMA modecleaner.

and for vertical misalignments:

$$Y_I = 0.7035\,\beta_\mathrm{a} - 0.7035\,\beta_\mathrm{b} = 0.7035\,\beta_-, \tag{11}$$

$$Y_Q = -0.524\,\beta_\mathrm{a} - 0.524\,\beta_\mathrm{b} - 2.105\,\beta_\mathrm{c} = -0.524\,\beta_+ - 2.105\,\beta_\mathrm{c}. \tag{12}$$

Using simple linear algebra we also find the "neutral" modes from these results. They are given in tables (3) and (6) above.

By inspecting tables 1 and 2 one finds that the main effect of these "neutral modes" is to shift the spot position on the far end mirror $\mathrm{M_c}$. Hence their control is not absolutely essential for the basic function of the modecleaner. In the present situation, there is neither feedback to $\mathrm{M_c}$ nor is the spot position on $\mathrm{M_c}$ monitored. One problem with this approach might be that according to equations (9) to (12), a small motion of $\mathrm{M_c}$ translates into error signals at the reflected light port which is 3…4 times larger than the error signal for a motion of the front mirrors of comparable amplitued. This huge error signal is then fed back the the *front* mirrors which are forced to compensate $\mathrm{M_c}$'s misorientation although they are not very efficient actuators for that purpose. Furthermore, such feedback shifts the spot position on $\mathrm{M_c}$ which due to the curved mirror surface again causes the whole circle to start. It is possible that some of the present dynamic range problems may be relieved by sensing and feeding back all degrees of freedom.

Because the main effect of the "neutral modes" is to shift the spot position on the far end mirror $\mathrm{M_c}$, the most straightforward approach is to sense this spot position (with a DC quadrant detector looking at the transmitted light). In tables 1 and 2 above, this shift is computed as Δx or Δy of point Pc, respectively.

However, the detector is located at a distance $e = 2.08\,\mathrm{m}$ behind $\mathrm{M_c}$, and the spot position detected is hence a linear combination[2] of Δx or Δy of point Pc, respectively with the angle δ_c.

This shift of the beam spot on the photodetector is also calculated by the program and printed as "End Δy" in Tables 1 and 2.

If we also scale it by the factor z_R for consistency (although the way of detection is different and hence an arbitrary scale factor appears anyway), and call the error signals thus obtained X_E and Y_E, we get

$$
\begin{aligned}
X_E &= -0.3383\,\alpha_\mathrm{a} + 0.2455\,\alpha_\mathrm{b} + 4.8613\,\alpha_\mathrm{c} && (13)\\
&= -0.0464\,\alpha_+ - 0.2919\,\alpha_- + 4.8613\,\alpha_\mathrm{c}, && (14)\\
Y_E &= -1.701\,\beta_\mathrm{a} - 1.705\,\beta_\mathrm{b} - 2.105\,\beta_\mathrm{c} && (15)\\
&= -1.703\,\beta_+ + 0.002\,\beta_- - 2.105\,\beta_\mathrm{c}. && (16)
\end{aligned}
$$

We see that for the horizontal direction, the additional error signal is dominated by the misorientation of $\mathrm{M_c}$ and could directly be fed back to that mirror, while for the vertical direction the signals are more mixed. However, they are still sufficiently linearly independent such that by inversion of the well-conditioned matrix error signals for all three mirrors can be found. This matrix inversion yields

$$
\begin{array}{lcrrr}
\alpha_\mathrm{a} & = & -1.6726\,X_I & +1.4273\,X_Q & +X_E \\
\alpha_\mathrm{b} & = & -1.6726\,X_I & -X_Q & +X_E \\
\alpha_\mathrm{c} & = & -0.0319\,X_I & +0.1498\,X_Q & +0.7015\,X_E
\end{array}
\qquad (17)
$$

$$
\begin{array}{lcrrr}
\beta_\mathrm{a} & = & +1.6785\,Y_I & +Y_Q & -Y_E \\
\beta_\mathrm{b} & = & -1.6785\,Y_I & +Y_Q & -Y_E \\
\beta_\mathrm{c} & = & -0.0015\,Y_I & -1.6186\,Y_Q & +0.4987\,Y_E
\end{array}
\qquad (18)
$$

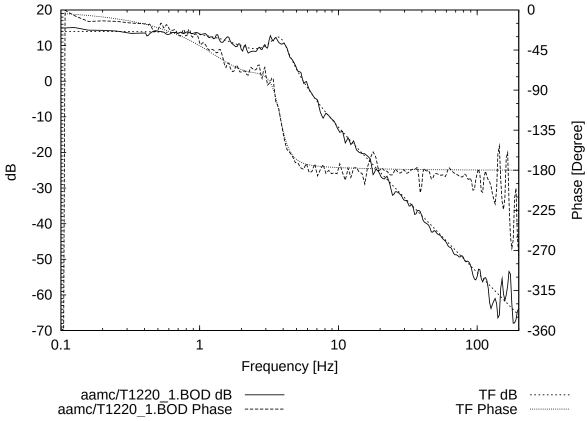
where scaling factors of 1/0.301462 and 1/0.42418, respectively, have been applied to make the most common coefficients unity. Note that the present alignment system uses only a 2×2 matrix without X_E, Y_E, α_c and β_c.

6 Actuator transfer functions

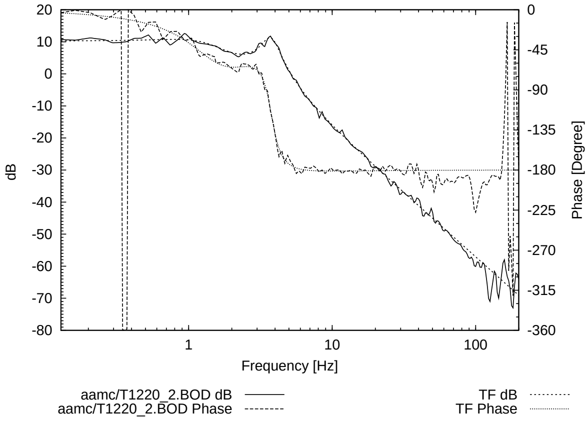
The transfer functions from all actuators to all sensors have been measured by K. Arai. The results were fitted with LISO and are shown on the following pages.

[2] Many thanks to K. Arai for noting this point.

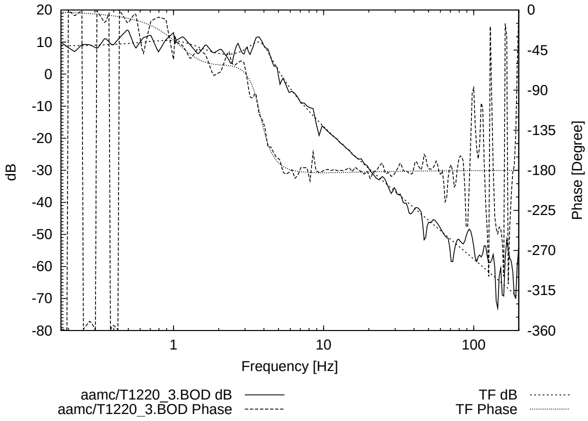
6.1 Input Mirror Yaw coil ("m1xcoil")



```
Pi Yaw error      T1220_1.BOD

pole0:f = 1.6798911 +- 75.2m (4.48%)
pole0:q = 636.47668m +- 21.6m (3.39%)
zero0:f = 3.1323094 +- 124.6m (3.98%)
zero0:q = 969.43027m +- 64m (6.6%)
pole1:f = 3.7858238 +- 25.9m (0.684%)
pole1:q = 3.3333594 +- 177.3m (5.32%)
factor = 5.0043609 +- 40.31m (0.805%)
```



```
Pi/2 Yaw error       T1220_2.BOD

pole0:f = 1.3598055 +- 43.77m (3.22%)
pole0:q = 830.24256m +- 19.97m (2.41%)
zero0:f = 2.4762582 +- 72.09m (2.91%)
zero0:q = 842.25894m +- 47.99m (5.7%)
pole1:f = 3.7674492 +- 19.53m (0.518%)
pole1:q = 3.711112 +- 128m (3.45%)
factor = 3.2644629 +- 29.84m (0.914%)
```

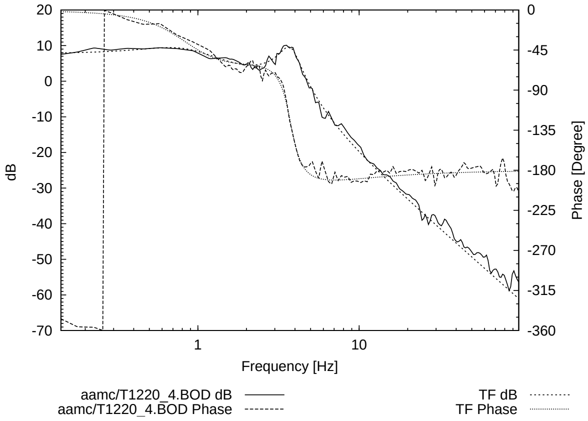


```
End QPD DC Yaw       T1220_3.BOD

pole0:f = 1.1584155 +- 66.98m (5.78%)
pole0:q = 948.86053m +- 53.28m (5.61%)
zero0:f = 2.0163492 +- 120.4m (5.97%)
zero0:q = 642.40736m +- 78.67m (12.2%)
pole1:f = 3.8080309 +- 46.85m (1.23%)
pole1:q = 3.0342956 +- 189.4m (6.24%)
factor = 2.7204019 +- 63.32m (2.33%)
```

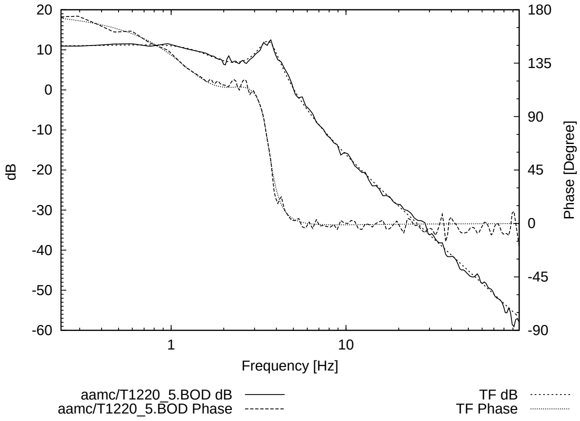
Weighted Averages:

```
pole0:f = 1.372388
pole0:q = 0.755353
pole1:f = 3.777486
pole1:q = 3.455169
zero0:f = 2.508392
zero0:q = 0.840509
```

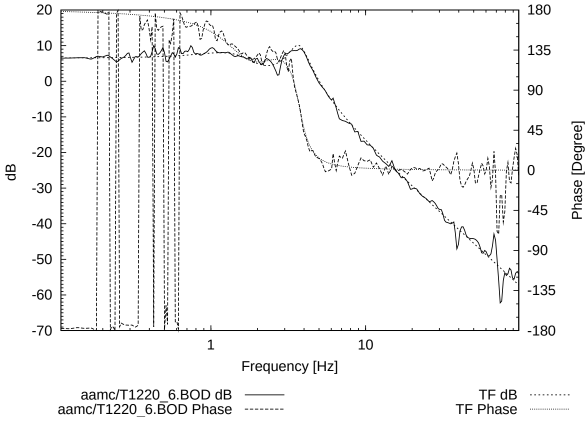
6.2 Output Mirror Yaw coil ("m2xcoil")



```
Pi Yaw error      T1220_4.BOD

pole0:f = 864.55282m +- 31.63m (3.66%)
pole0:q = 801.30084m +- 22.58m (2.82%)
zero0:f = 1.7498418 +- 67.69m (3.87%)
zero0:q = 436.75937m +- 27.63m (6.33%)
pole1:f = 3.7556255 +- 19.05m (0.507%)
pole1:q = 3.6996359 +- 118m (3.19%)
factor = 2.4554303 +- 33.68m (1.37%)
```



```
Pi/2 Yaw error       T1220_5.BOD

pole0:f = 1.3974404 +- 26.7m (1.91%)
pole0:q = 792.95797m +- 11.9m (1.5%)
zero0:f = 2.5625299 +- 44.11m (1.72%)
zero0:q = 877.60669m +- 27.81m (3.17%)
pole1:f = 3.630812 +- 9.79m (0.27%)
pole1:q = 3.9492515 +- 79.85m (2.02%)
factor = -3.5256953 +- -21.65m (0.614%)
```

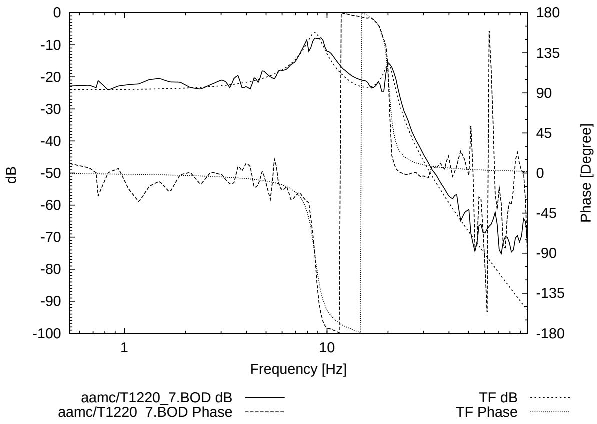


```
End QPD DC Yaw       T1220_6.BOD

pole0:f = 1.5309455 +- 68.42m (4.47%)
pole0:q = 1.0838276 +- 56.89m (5.25%)
zero0:f = 2.2872224 +- 93.83m (4.1%)
zero0:q = 1.0726809 +- 108.7m (10.1%)
pole1:f = 3.7250662 +- 31.92m (0.857%)
pole1:q = 3.6033149 +- 192.1m (5.33%)
factor = -2.1236194 +- -32.95m (1.55%)
```

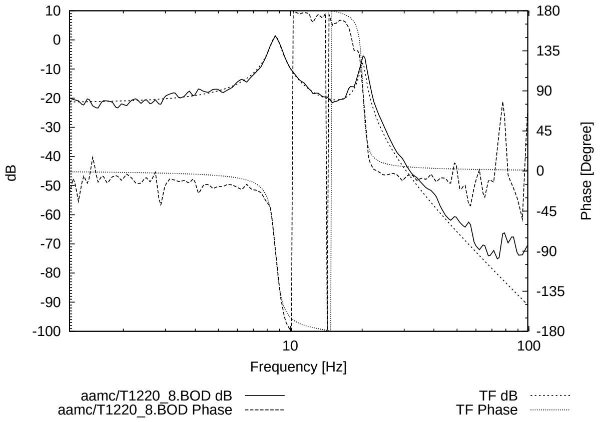
Weighted Averages:

```
pole0:f = 1.204886
pole0:q = 0.804331
pole1:f = 3.661688
pole1:q = 3.842627
zero0:f = 2.316205
zero0:q = 0.669030
```

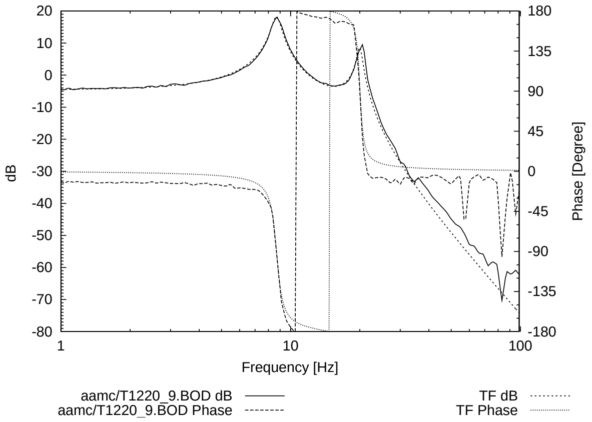
6.3 Input Mirror Pitch PZT (“m1ypzt”)



```
Pi Pitch error       T1220_7.BOD

pole0:f = 8.721352 +- 31.18m (0.357%)
pole0:q = 6.3320232 +- 330.2m (5.22%)
pole1:f = 20.249388 +- 100.9m (0.498%)
pole1:q = 11.065166 +- 1.296 (11.7%)
factor = 62.862325m +- 1.561m (2.48%)
```



```
Pi/2 Pitch error     T1220_8.BOD

pole0:f = 8.6836151 +- 12.55m (0.145%)
pole0:q = 11.122052 +- 405.8m (3.65%)
pole1:f = 20.153226 +- 42.54m (0.211%)
pole1:q = 20.638458 +- 1.887 (9.15%)
factor = 84.273794m +- 1.359m (1.61%)
```

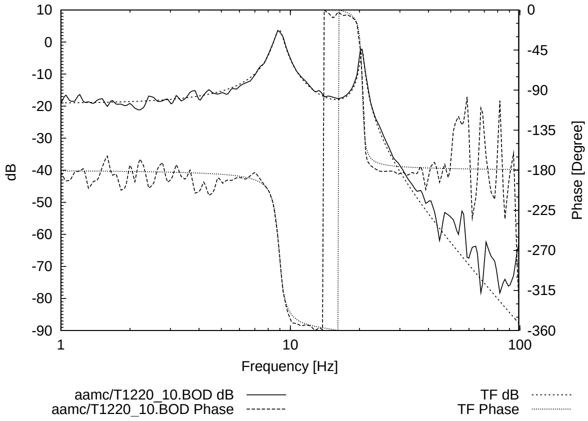


```
End QPD DC Pitch     T1220_9.BOD

pole0:f = 8.6874272 +- 10.29m (0.118%)
pole0:q = 11.243448 +- 337.4m (3%)
pole1:f = 20.031176 +- 32.46m (0.162%)
pole1:q = 19.667483 +- 1.306 (6.64%)
factor = 590.1168m +- 7.742m (1.31%)
```

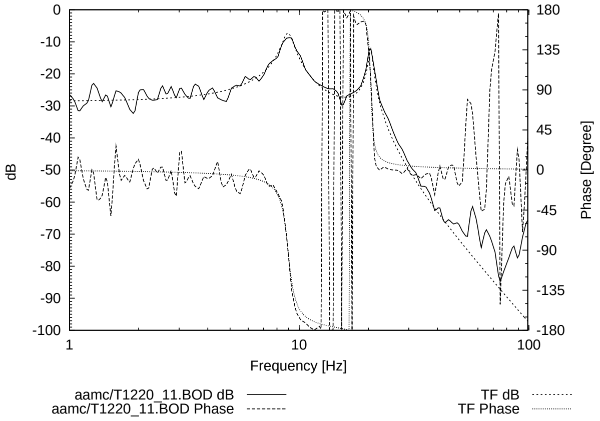
Weighted Averages:

```
pole0:f = 8.688076
pole0:q = 9.340454
pole1:f = 20.08673
pole1:q = 16.34634
```

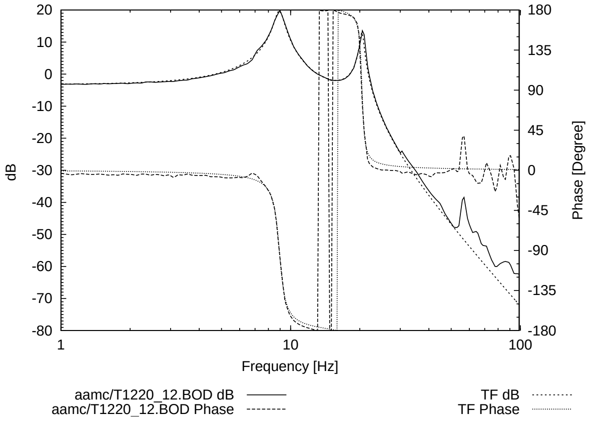
6.4 Output Mirror Pitch PZT (“m2ypzt”)



```
Pi Pitch error       T1220_10.BOD

pole0:f = 8.9312815 +- 8.936m (0.1%)
pole0:q = 11.488715 +- 297.2m (2.59%)
pole1:f = 20.561491 +- 19.32m (0.094%)
pole1:q = 29.210305 +- 1.681 (5.76%)
factor = -110.87858m +- -1.238m (1.12%)
```



```
Pi/2 Pitch error     T1220_11.BOD

pole0:f = 8.947484 +- 15.86m (0.177%)
pole0:q = 9.3417483 +- 354.6m (3.8%)
pole1:f = 20.507774 +- 30.04m (0.146%)
pole1:q = 27.878288 +- 2.424 (8.69%)
factor = 37.419176m +- 645.2u (1.72%)
```

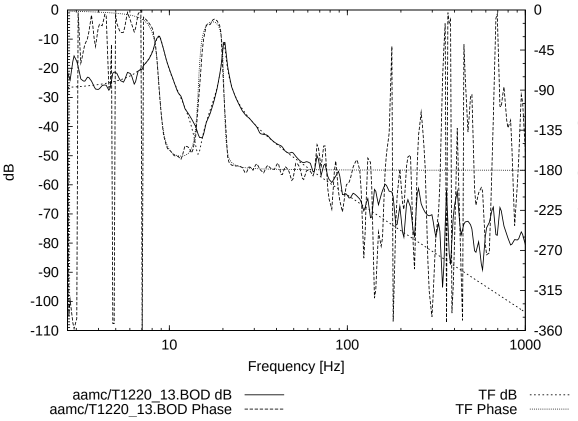


```
End QPD DC Pitch     T1220_12.BOD

pole0:f = 8.9347062 +- 5.266m (0.0589%)
pole0:q = 11.469713 +- 173.4m (1.51%)
pole1:f = 20.448379 +- 11.75m (0.0575%)
pole1:q = 27.958598 +- 926.9m (3.32%)
factor = 688.79131m +- 4.501m (0.653%)
```

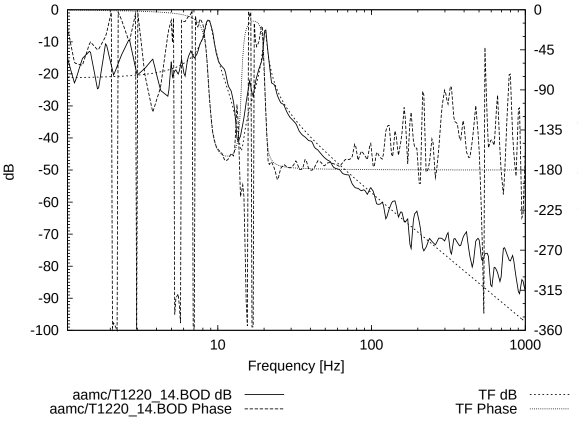
Weighted Averages:

```
pole0:f = 8.934858
pole0:q = 11.15256
pole1:f = 20.48184
pole1:q = 28.21300
```

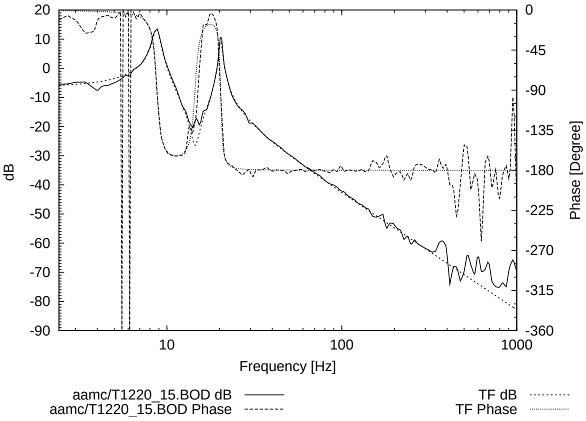
6.5 Input Mirror Pitch coil (“m1ycoil”)



```
Pi Pitch error        T1220_13.BOD

pole0:f = 8.7707005 +- 19.38m (0.221%)
pole0:q = 10.585199 +- 541.9m (5.12%)
zero0:f = 14.532192 +- 125.6m (0.864%)
zero0:q = 15.263638 +- 4.742 (31.1%)
pole1:f = 20.291927 +- 33.79m (0.167%)
pole1:q = 27.515739 +- 2.803 (10.2%)
factor = 43.351314m +- 975.7u (2.25%)
```



```
Pi/2 Pitch error      T1220_14.BOD

pole0:f = 8.7581187 +- 23.51m (0.268%)
pole0:q = 10.768086 +- 656.8m (6.1%)
zero0:f = 14.220014 +- 291m (2.05%)
zero0:q =  16.341334 */ 1.856
pole1:f = 20.299641 +- 47.29m (0.233%)
pole1:q = 23.289386 +- 3.274 (14.1%)
factor = 86.457836m +- 3.091m (3.58%)
```

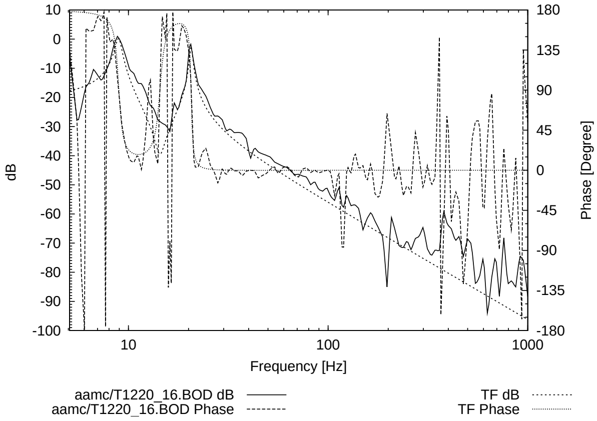


```
End QPD DC Pitch      T1220_15.BOD

pole0:f = 8.7555137 +- 8.536m (0.0975%)
pole0:q = 12.695277 +- 333.3m (2.63%)
zero0:f = 14.497233 +- 67.74m (0.467%)
zero0:q = 11.732884 +- 1.393 (11.9%)
pole1:f = 20.321918 +- 16.99m (0.0836%)
pole1:q = 28.889025 +- 1.515 (5.24%)
factor = 478.86074m +- 5.221m (1.09%)
```

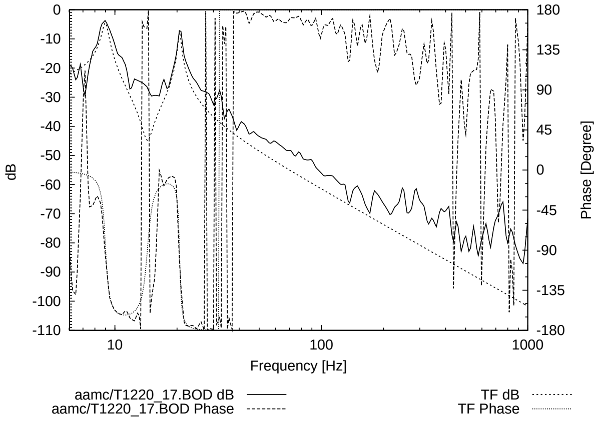
Weighted Averages:

```
pole0:f = 8.757994
pole0:q = 11.90231
pole1:f = 20.31437
pole1:q = 27.83178
zero0:f = 14.49367
zero0:q = 12.04289
```

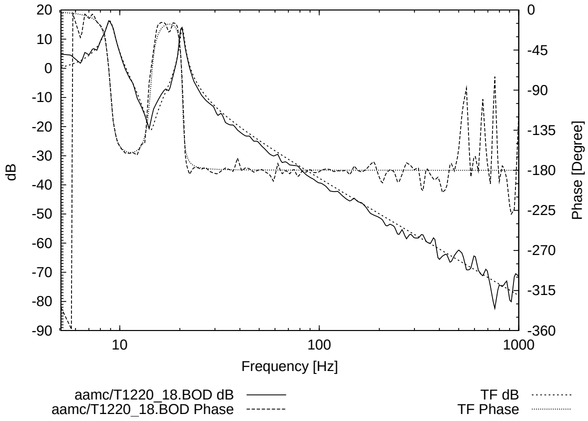
6.6 Output Mirror Pitch coil ("m2ycoil")



```
Pi Pitch error       T1220_16.BOD

pole0:f = 8.9343651 +- 27.53m (0.308%)
pole0:q = 14.281851 +- 1.225 (8.58%)
zero0:f = 14.399736 +- 235m (1.63%)
zero0:q = 10.829207 +- 272.2m (2.51%) > MAX
pole1:f = 20.60976 +- 44.38m (0.215%)
pole1:q = 33.383364 +- 5.186 (15.5%)
factor = -94.29963m +- -3.853m (4.09%)
```



```
Pi/2 Pitch error     T1220_17.BOD

pole0:f = 8.9970043 +- 32.66m (0.363%)
pole0:q = 15.935303 +- 1.807 (11.3%)
zero0:f = 14.374722 +- 317.3m (2.21%)
zero0:q = 10.838184 +- 241.2m (2.23%) > MAX
pole1:f = 20.577195 +- 61.19m (0.297%)
pole1:q = 33.687723 +- 6.336 (18.8%)
factor = 49.476234m +- 2.697m (5.45%)
```

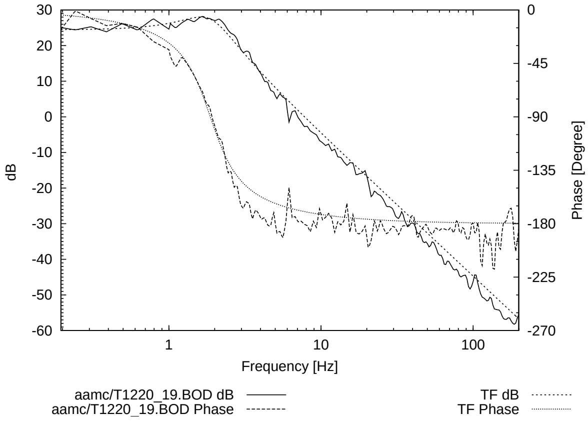


```
End QPD DC Pitch     T1220_18.BOD

pole0:f = 8.9471133 +- 10.88m (0.122%)
pole0:q = 11.299087 +- 344.5m (3.05%)
zero0:f = 14.280006 +- 68.37m (0.479%)
zero0:q = 10.79691 +- 1.226 (11.4%)
pole1:f = 20.579641 +- 17.91m (0.087%)
pole1:q = 26.245019 +- 1.311 (4.99%)
factor = 761.14975m +- 10.98m (1.44%)
```

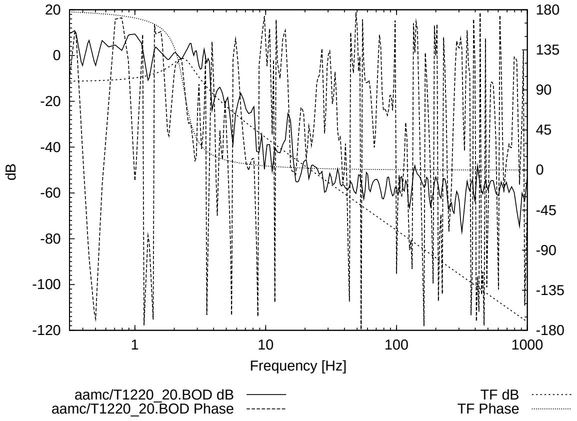
Weighted Averages:

```
pole0:f = 8.949924
pole0:q = 11.66273
pole1:f = 20.58340
pole1:q = 26.94568
zero0:f = 14.29287
zero0:q = 10.79691
```

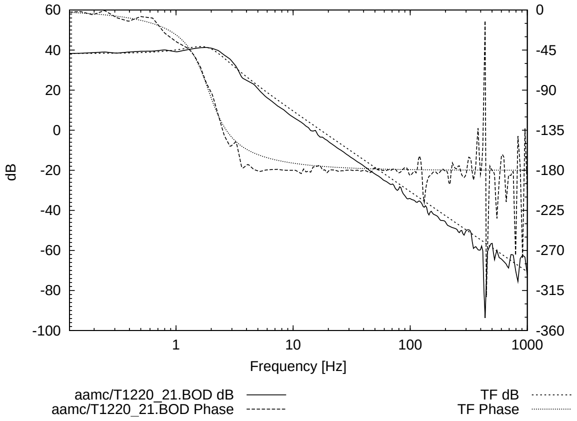
6.7 End Mirror Yaw Coil (“m3xcoil”)



```
Pi Yaw error      T1220_19.BOD

pole0:f = 1.8713374 +- 11.13m (0.595%)
pole0:q = 1.4166323 +- 29.77m (2.1%)
factor = 16.61447 +- 181.6m (1.09%)
```



```
Pi/2 Coil Yaw error  T1220_20.BOD

pole0:f = 2.3769774 +- 87.34m (3.67%)
pole0:q = 3.1497205 +- 1.04 (33%)
factor = -270.8383m +- -49.8m (18.4%)
```

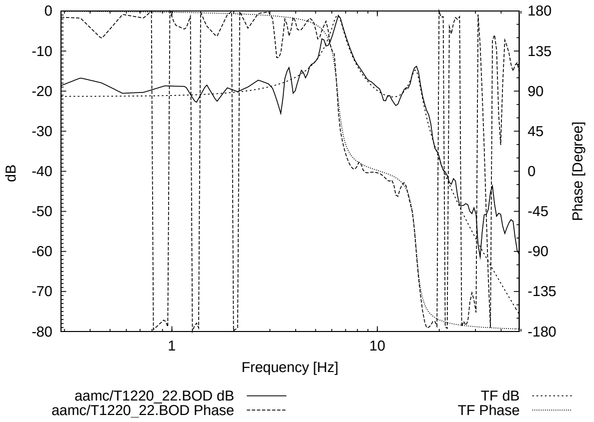


```
End QPD DC Yaw       T1220_21.BOD

pole0:f = 1.8896638 +- 8.812m (0.466%)
pole0:q = 1.3891876 +- 22.04m (1.59%)
factor = 82.249734 +- 641.4m (0.78%)
```

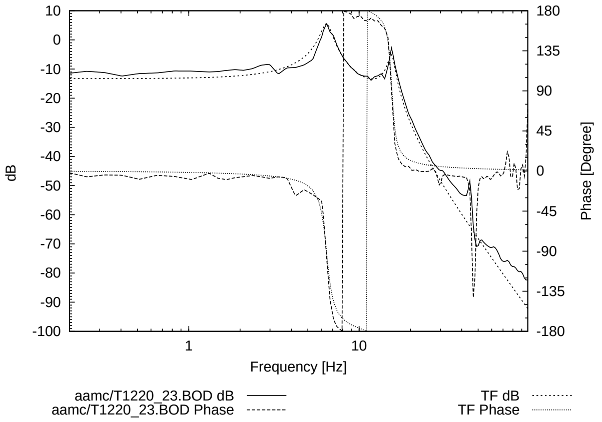
Weighted Averages:

```
pole0:f = 1.8826161
pole0:q = 1.3989960
```

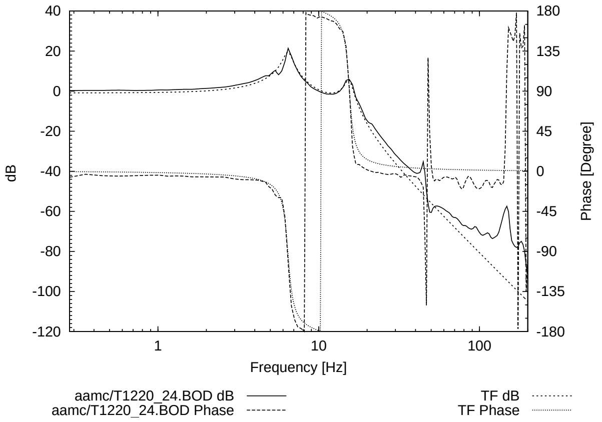
6.8 End Mirror Pitch PZT (“m3ypzt”)



```
Pi Pitch error       T1220_22.BOD

pole0:f = 6.4108286 +- 18.1m (0.282%)
pole0:q = 8.4989131 +- 459.5m (5.41%)
pole1:f = 15.438905 +- 94.27m (0.611%)
pole1:q = 10.134435 +- 1.324 (13.1%)
factor = -85.75237m +- -2.02m (2.36%)
```



```
Pi/2 Pitch error     T1220_23.BOD

pole0:f = 6.4501997 +- 17.03m (0.264%)
pole0:q = 7.2412406 +- 310.6m (4.29%)
pole1:f = 15.548908 +- 63.98m (0.411%)
pole1:q = 12.259952 +- 1.299 (10.6%)
factor = 216.30985m +- 4.018m (1.86%)
```

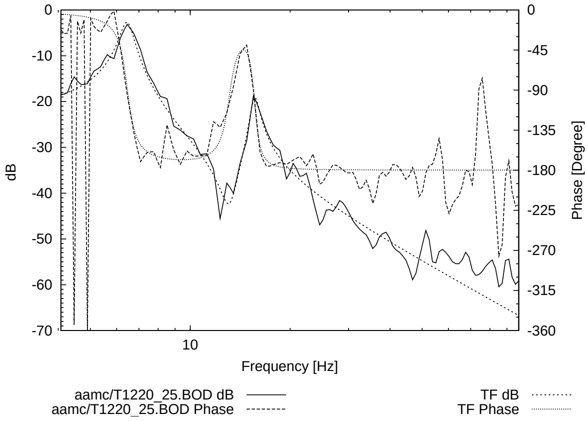


```
End QPD DC Pitch     T1220_24.BOD

pole0:f = 6.441543 +- 11.96m (0.186%)
pole0:q = 8.6539705 +- 308.9m (3.57%)
pole1:f = 15.47822 +- 62.81m (0.406%)
pole1:q = 10.059454 +- 849.4m (8.44%)
factor = 904.141m +- 14m (1.55%)
```

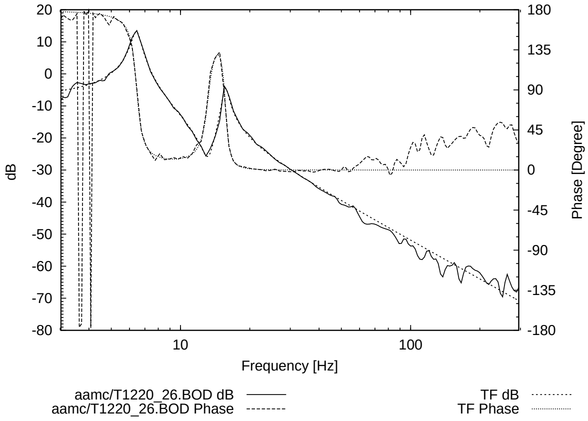
Weighted Averages:

```
pole0:f = 6.436784
pole0:q = 8.052757
pole1:f = 15.49933
pole1:q = 10.58777
```

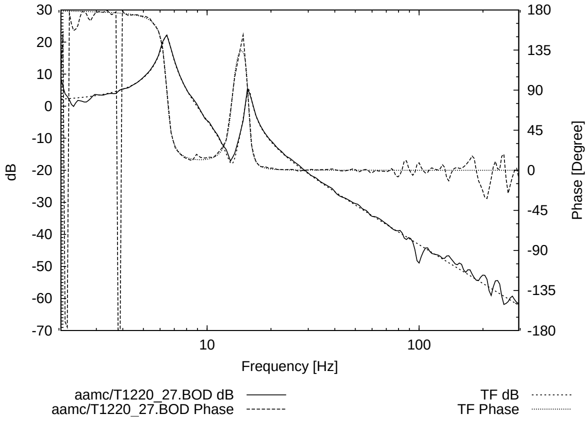
6.9 End Mirror Pitch coil ("m3ycoil")



```
Pi Pitch error       T1220_25.BOD

pole0:f = 6.4562745 +- 12.24m (0.19%)
pole0:q = 10.880457 +- 519.8m (4.78%)
zero0:f = 13.118087 +- 141.4m (1.08%)
zero0:q = 10.177679 +- 2.379 (23.4%)
pole1:f = 15.621772 +- 66.15m (0.423%)
pole1:q = 16.829252 +- 2.755 (16.4%)
factor = 73.69469m +- 1.956m (2.65%)
```



```
Pi/2 Pitch error     T1220_26.BOD

pole0:f = 6.4394746 +- 4.254m (0.0661%)
pole0:q = 11.47595 +- 191.8m (1.67%)
zero0:f = 13.214751 +- 53.88m (0.408%)
zero0:q = 8.8553113 +- 623m (7.04%)
pole1:f = 15.652186 +- 26.16m (0.167%)
pole1:q = 15.990668 +- 937.6m (5.86%)
factor = -433.8639m +- -3.66m (0.843%)
```



```
End QPD DC Pitch     T1220_27.BOD

pole0:f = 6.4356973 +- 6.681m (0.104%)
pole0:q = 11.732925 +- 311.2m (2.65%)
zero0:f = 13.167754 +- 85.29m (0.648%)
zero0:q = 9.613809 +- 1.201 (12.5%)
pole1:f = 15.663831 +- 37.03m (0.236%)
pole1:q = 17.752179 +- 1.652 (9.31%)
factor = -1.1676377 +- -14.64m (1.25%)
```

Weighted Averages:

```
pole0:f = 6.439799
pole0:q = 11.48701
pole1:f = 15.65282
pole1:q = 16.45207
zero0:f = 13.19353
zero0:q = 9.075651
```

7 Calibration Factors

Both actuators and sensors have calibration factors which are not yet known. In order to find them (and test the above theroretical model), the LISO-fitted `factor`s from the transfer functions were fitted to simple model

$$\texttt{factor} = A_{\mathrm{optical}} \times G_{\mathrm{actuator}} \times G_{\mathrm{sensor}}. \tag{19}$$

where *all* measurements were fitted simultaneously to one set of calibration constants G_i.

For that purpose a simple C program ("`prod.c`") was written which uses the Nelder-Mead Simplex algorithm to minimize

$$\chi^2(G_1,\ldots,G_m) = \sum_{i=1}^{n_{\mathrm{data}}} w_i \left| f_i - \tilde{f}_i(G_1,\ldots,G_m) \right|^2 \tag{20}$$

where f_i are the measured factors, $\tilde{f}_i$ their approximations from Equation (19), $w_i = 1/\sigma_i^2$ are the weights of each data point derived from their standard deviation (taken from the LISO fit), and G_i the unknown calibration constants.

In fact the data set consists of two independent sets of data (for x and y), but since this does not adversely affect the fit they were nevertheless fitted all in one set.

To remove an inherent ambiguity in the model (19), the calibration factor (sensitivity) for the end detector was arbitrarily fixed at unity:

$$\texttt{xe} = 1, \tag{21}$$

$$\texttt{ye} = 1. \tag{22}$$

Three measurements where the theoretically expected factor was zero (and the measured one was indeed small) were excluded from the fit. Hence there are 13 parameters and 24 data points.

After convergence of the Simplex algorithm, the solution was refined by a simplified Levenberg-Marquardt algorithm, and the standard errors of the parameters were computed (see Appendix of LISO manual for relevant formulae).

The results are:

Parameter	est.	σ	σ
xi	0.371443	0.0531476	14.308%
xq	−0.274189	0.0435769	15.893%
m1xcoil	−10.0941	1.56787	15.533%
m2xcoil	−8.55752	1.2643	14.774%
m3xcoil	16.3377	1.83309	11.220%
yi	−0.313443	0.052152	16.638%
yq	0.270632	0.0341316	12.612%
m1ypzt	−0.365368	0.0571409	15.639%
m2ypzt	−0.38773	0.0357978	9.233%
m3ypzt	−0.406073	0.077663	19.125%
m1ycoil	−0.25863	0.0416197	16.092%
m2ycoil	−0.43544	0.0878683	20.179%
m3ycoil	0.66633	0.0813667	12.211%

The following table shows the input data and residuals after the fit.

G_a	G_s	A_{opt}	σ	meas.	calc.	ratio
m1xcoil	xi	−1.0190	0.04031	5.00436	3.82063	-2.34 db
m1xcoil	xq	1.3660	0.02984	3.26446	3.78069	1.28 db
m1xcoil	1.0	−0.3383	0.06332	2.72040	3.41485	1.97 db
m2xcoil	xi	−1.0190	0.03368	2.45543	3.23903	2.41 db
m2xcoil	xq	−1.3660	0.02165	−3.52570	−3.20516	-0.83 db
m2xcoil	1.0	0.2454	0.03295	−2.12362	−2.10002	-0.10 db
m3xcoil	xi	2.9060	0.1816	16.61447	17.63516	0.52 db
m3xcoil	1.0	4.8613	0.6414	82.24973	79.42255	-0.30 db
m1ypzt	yi	0.7035	0.001561	0.06286	0.08057	2.16 db
m1ypzt	yq	−0.5240	0.001359	0.08427	0.05181	-4.23 db
m1ypzt	1.0	−1.7010	0.007742	0.59012	0.62149	0.45 db
m2ypzt	yi	−0.7035	0.001238	−0.11088	−0.08550	-2.26 db
m2ypzt	yq	−0.5240	0.0006452	0.03742	0.05498	3.34 db
m2ypzt	1.0	−1.7050	0.004501	0.68879	0.66108	-0.36 db
m1ycoil	yi	0.7035	0.0009757	0.04335	0.05703	2.38 db
m1ycoil	yq	−0.5240	0.003091	0.08646	0.03668	-7.45 db
m1ycoil	1.0	−1.7010	0.005221	0.47886	0.43993	-0.74 db
m2ycoil	yi	−0.7035	0.003853	−0.09430	−0.09602	0.16 db
m2ycoil	yq	−0.5240	0.002697	0.04948	0.06175	1.92 db
m2ycoil	1.0	−1.7050	0.01098	0.76115	0.74242	-0.22 db
m3ypzt	yq	−2.1050	0.004018	0.21631	0.23133	0.58 db
m3ypzt	1.0	−2.1050	0.014	0.90414	0.85478	-0.49 db
m3ycoil	yq	−2.1050	0.00366	−0.43386	−0.37960	-1.16 db
m3ycoil	1.0	−2.1050	0.01464	−1.16764	−1.40263	1.59 db

7.1 Choice of actuators

For the x direction, there are only coils available as actuators. On the other hand, for the y direction there are coils and PZT actuators. During the measurement of the transfer functions it was found that the y coils have only very small response at DC, while their response at their respective resonances is quite large. This is caused by the design of the suspension, which is essentially stiff against pitch actuation at the mirror. Hence it was decided to use the PZT's for the y actuation.

Note: After the measurements described above, the coil driver for the end mirror (input `m3xcoil`) was changed. The gain was increased by 6 dB, and the polarity was reversed. That is taken into account in the module by appropriate settings for the loop gain and loop feedback polarity.

8 Port usage

We use **mode 7 - Advanced single chip mode**. In this mode, only internal RAM (4k) and Flash-ROM (128k) are used, and all I/O lines are available. In the other modes, many of them are used as address / data lines for external memory.

8.1 Port 1

8-bit I/O, can drive LED (sink 10 mA), TTL + 90 pF, Darlington.

Registers P1DDR - data direction register: 0=Input, 1=Output, default=0;
P1DR - data register, default=0.

Usage Output port for 8 status LED's, via HC04 Inverter. Logic is active Low, i.e. 0 = LED On, 1=LED Off

P1.DR.BIT.B0 LED0 Ch1 Red

P1.DR.BIT.B1 LED1 Ch2 Red

P1.DR.BIT.B2 LED2 Ch3 Red

P1.DR.BIT.B3 LED3 Ch4 Red

P1.DR.BIT.B4 LED4 Ch5 Red

P1.DR.BIT.B5 LED5 Ch6 Red

P1.DR.BIT.B6 LED6 Green 1

P1.DR.BIT.B7 LED7 Green 2

8.2 Port 2

8-bit I/O, can drive TTL + 90 pF, Darlington. Input: programmable pull-up ($50 \ldots 300\mu$A = $16 \ldots 100$kΩ).

Registers P2DDR - data direction register: 0=Input, 1=Output, default=0;
P2DR - data register, default=0,
P2PCR - pull up register, 0=Off, 1=pull-up, default=0.

Usage Output port for 8 status LED's, via HC04 Inverter. Logic is active Low, i.e. 0 = LED On, 1=LED Off

P2.DR.BIT.B0 LED8 Ch1 Yellow

P2.DR.BIT.B1 LED9 Ch2 Yellow

P2.DR.BIT.B2 LED10 Ch3 Yellow

P2.DR.BIT.B3 LED11 Ch4 Yellow

P2.DR.BIT.B4 LED12 Ch5 Yellow

P2.DR.BIT.B5 LED13 Ch6 Yellow

P2.DR.BIT.B6 LED14 unused

P2.DR.BIT.B7 LED15 unused

8.3 Port 3

8-bit I/O, can drive TTL + 90 pF, Darlington.

Registers P3DDR - data direction register: 0=Input, 1=Output, default=0;
P3DR - data register, default=0.

Usage: Output port for LCD display

P3.DR.BIT.B0 LCD Data bit 4

P3.DR.BIT.B1 LCD Data bit 5

P3.DR.BIT.B2 LCD Data bit 6

P3.DR.BIT.B3 LCD Data bit 7

P3.DR.BIT.B4 LCD R/S signal

P3.DR.BIT.B5 LCD enable signal

P3.DR.BIT.B6 unused

P3.DR.BIT.B7 unused

8.4 Port 4

8-bit I/O, can drive TTL + 90 pF, Darlington. Input: programmable pull-up ($50 \ldots 300\mu$A = $16 \ldots 100$kΩ).

Registers P4DDR - data direction register: 0=Input, 1=Output, default=0;
P4DR - data register, default=0,
P4PCR - pull up register, 0=Off, 1=pull-up, default=0.

Usage Digital Input, pull up enabled

P4.DR.BIT.B0 via 74HC04 Inverter and Pullup - Digital Input 0

P4.DR.BIT.B1 via 74HC04 Inverter and Pullup - Digital Input 1

P4.DR.BIT.B2 via 74HC04 Inverter and Pullup - Digital Input 2

P4.DR.BIT.B3 via 74HC04 Inverter and Pullup - Digital Input 3

P4.DR.BIT.B4 via 74HC04 Inverter and Pullup - Digital Input 4

P4.DR.BIT.B5 via 74HC04 Inverter and Pullup - Digital Input 5

P4.DR.BIT.B6 - Digital Input 6

P4.DR.BIT.B7 - Digital Input 7

P4.DR.BIT.B0. . . P4.DR.BIT.B3 BCD switch 0...9

P4.DR.BIT.B4 Reset key 1=pressed

P4.DR.BIT.B5 left flipswitch 0=On

P4.DR.BIT.B6 centre flipswitch 0=On

P4.DR.BIT.B7 right flipswitch 0=On

8.5 Port 5

4-bit I/O, can drive LED (sink 10 mA), TTL + 90 pF, Darlington, Input: programmable pull-up ($50 \dots 300 \mu A = 16 \dots 100 k\Omega$).

Registers P5DDR - data direction register: 0=Input, 1=Output, default=0;
P5DR - data register, default=0,
P5PCR - pull up register, 0=Off, 1=pull-up, default=0.

Usage DAC8420 Control (2 Chips 2 bits each)

P5.DR.BIT.B0 DAC2 LD

P5.DR.BIT.B1 DAC2 CS

P5.DR.BIT.B2 DAC1 LD

P5.DR.BIT.B3 DAC1 CS

8.6 Port 6

7-bit I/O, can drive TTL + 30 pF, Darlington,

Registers P6DDR - data direction register: 0=Input, 1=Output, default=0;
P6DR - data register, default=0.

Usage unused

P6.DR.BIT.B0

P6.DR.BIT.B1

P6.DR.BIT.B2

P6.DR.BIT.B3

P6.DR.BIT.B4

P6.DR.BIT.B5

P6.DR.BIT.B6

8.7 Port 7

8-bit Input / Analog Input / Analog Output.
8 channel 10-bit A/D on pins 0. . . 7;
2 channel 8-bit D/A on pins 6 and 7.

Registers P7DR - data register, default=0.

Usage pin 0 – A/D group 0 : AD channel 0
pin 1 – A/D group 0 : AD channel 1
pin 2 – A/D group 0 : AD channel 2
pin 3 – A/D group 0 : AD channel 3
pin 4 – A/D group 1 : AD channel 4
pin 5 – A/D group 1 : AD channel 5
pin 6 – A/D group 1 : AD channel 6
pin 7 – A/D group 1 : AD channel 7

8.8 Port 8

5-bit I/O, also IRQ_0,. . . ,IRQ_3.
pin 0 is I/O or IRQ_0, Schmitt-trigger input
pin 1 is I/O or IRQ_1, Schmitt-trigger input
pin 2 is I/O or IRQ_2, Schmitt-trigger input
pin 3 is I/O or IRQ_3,
pin 4 is I/O.
Output can drive TTL + 90 pF, Darlington.

Registers P8DDR - data direction register: 0=Input, 1=Output, default=0;
P8DR - data register, default=0.
IER - enables interrupt regardless of P8DDR.

Usage pin 0 – IRQ 0 – user pressbutton

8.9 Port 9

6-bit I/O, can drive TTL + 30 pF, Darlington,
also used as serial port and IRQ_4, IRQ_5.
pin 0 is I/O or TxD_0,
pin 1 is I/O or TxD_1,
pin 2 is I/O or RxD_0,
pin 3 is I/O or RxD_1,
pin 4 is I/O or SCK_0 or IRQ_4,
pin 5 is I/O or SCK_1 or IRQ_5.

Registers P9DDR - data direction register: 0=Input, 1=Output, default=0;
P9DR - data register, default=0.
serial mode / IRQ overrides P9DDR.

Usage pin 1 and pin 3 (SCI1) for serial communication with PC via RS232 buffer
pins 0,4 (SCI0) for synchronous serial data transfer to DAC8420

8.10 Port A

8-bit I/O, can drive TTL + 30 pF, Darlington,
Schmitt-trigger input
also integrated timer unit I/O, DMA output, timing pattern output.
pin 0 is I/O or TP_0 (out) or $TEND_0$ (out) or TCLKA (in),
pin 1 is I/O or TP_1 (out) or $TEND_1$ (out) or TCLKB (in),
pin 2 is I/O or TP_2 (out) or $TIOCA_0$ (I/O) or TCLKC (in),
pin 3 is I/O or TP_3 (out) or $TIOCB_0$ (I/O) or TCLKD (in),
pin 4 is I/O or TP_4 (out) or $TIOCA_1$ (I/O),
pin 5 is I/O or TP_5 (out) or $TIOCB_1$ (I/O),
pin 6 is I/O or TP_6 (out) or $TIOCA_2$ (I/O),
pin 7 is I/O or TP_7 (out) or $TIOCB_2$ (I/O).

Registers PADDR - data direction register: 0=Input, 1=Output, default=0;
PADR - data register, default=0.
if timing pattern output, PADDR bit must be set.

Usage pin 0 and pin 1 input from quadrature encoder switch
pin 2 implicit use to cascade connect timers 0 and 1 (”finetime”)

8.11 Port B

8-bit I/O, can drive LED (sink 10 mA), TTL + 30 pF, Darlington,
pin 0. . . 3 have Schmitt-trigger input.
also integrated timer unit I/O, DMA input, timing pattern output, A/D trigger.

pin 0 is I/O or TP_8 (out) or TIOCA_3 (I/O),
pin 1 is I/O or TP_9 (out) or TIOCB_3 (I/O),
pin 2 is I/O or TP_{10} (out) or TIOCA_4 (I/O),
pin 3 is I/O or TP_{11} (out) or TIOCB_4 (I/O),
pin 4 is I/O or TP_{12} (out) or TIOXA_4 (out),
pin 5 is I/O or TP_{13} (out) or TIOXB_4 (out),
pin 6 is I/O or TP_{14} (out) or DREQ_0 (in),
pin 7 is I/O or TP_{15} (out) or DREQ_1 (in), ADTRG.

Registers PBDDR - data direction register: 0=Input, 1=Output, default=0;
PBDR - data register, default=0.
if timing pattern output, PBDDR bit must be set.

Usage pin 0 (TIOCA3) has 1/2 sampling frequency as monitor.

9 Analog Inputs/Outputs

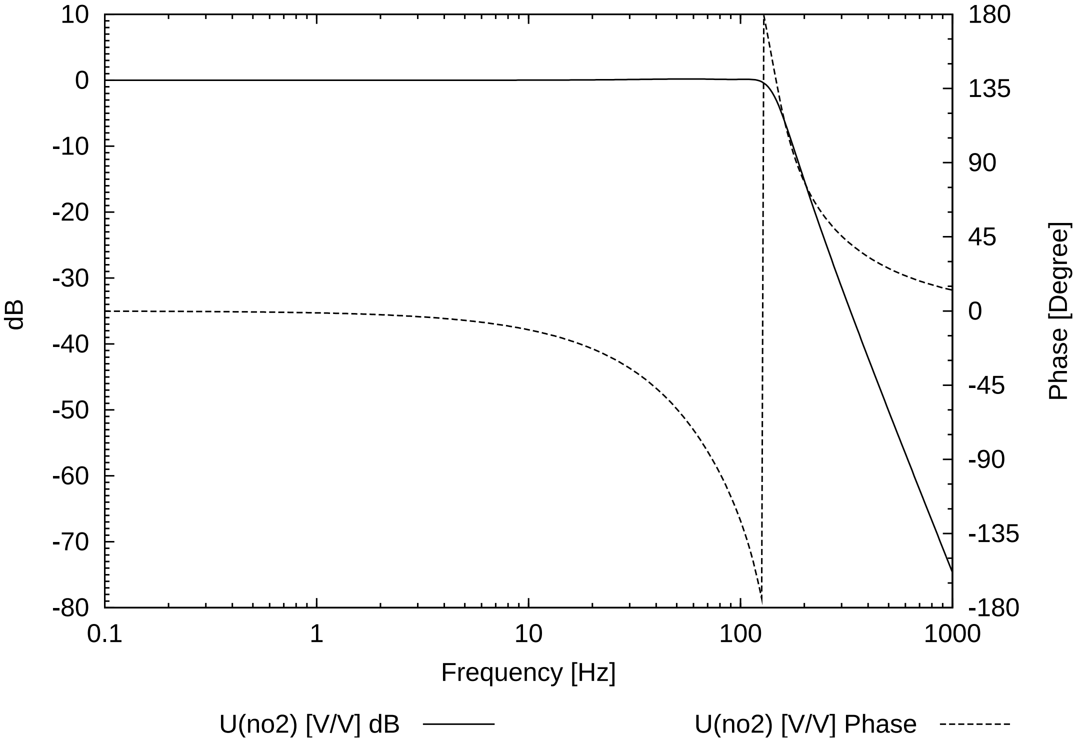
The A/D converter of the 3048 uses a stabilized +5 V reference. The nominal resolution is 10 bits, and the input range 0. . . 5 Volt. By averaging a few samples, an effective resolution of 11. . . 12 bits is achieved. All 8 analog input channels are used as follows:

Channel No.	Preamp Gain	Input Range [V]	Signal
0	5.167	-0.648 . . . +0.648	I detector x
1	5.167	-0.648 . . . +0.648	I detector y
2	5.167	-0.648 . . . +0.648	Q detector x
3	5.167	-0.648 . . . +0.648	Q detector y
4	1	-3.35 . . . 3.35	End detector x
5	1	-3.35 . . . 3.35	End detector y
6	-1. . . -10	0. . . -0.5 to 0 . . . -5 (var.)	Laser power
7	-1. . . -10	0. . . -0.5 to 0 . . . -5 (var.)	Internal power

All channels have analog 4-pole antialiasing filters with a 0.1 dB Tschebysheff characteristic and a corner frequency of 120 Hz. Their poles are

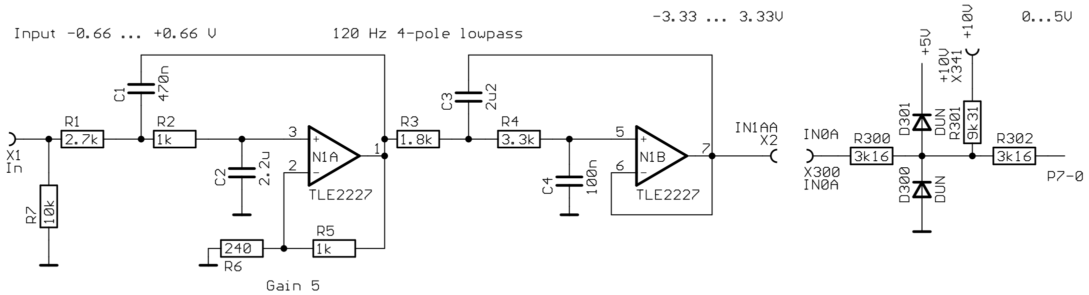
Freq [Hz]	Q
$0.789 \times 120 = 94.68$	0.619
$1.153 \times 120 = 138.36$	2.183

The following diagram shows the frequency response of the 120 Hz Tschebysheff filter (same for all input and output channels apart from a gain factor):

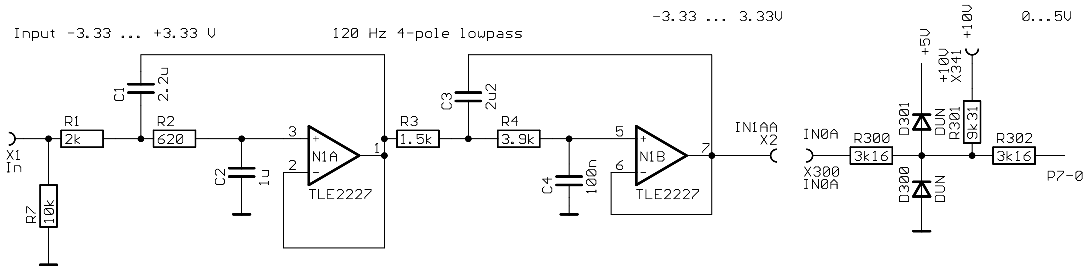


The filters are realized as 2-stage Sallen-Key circuits (see below). Furthermore, all A/D inputs of the 3048 are protected with a resistor-diode network. Channels 0. . . 5 have an additional resistor to the stabilized 10 V reference which is used for level-shifting.

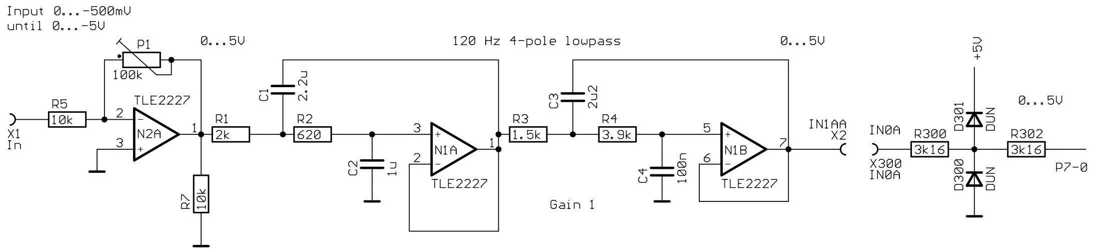
The following diagram shows the input circuit for channels 0. . . 3:



The following diagram shows the input circuit for channels 4 and 5:



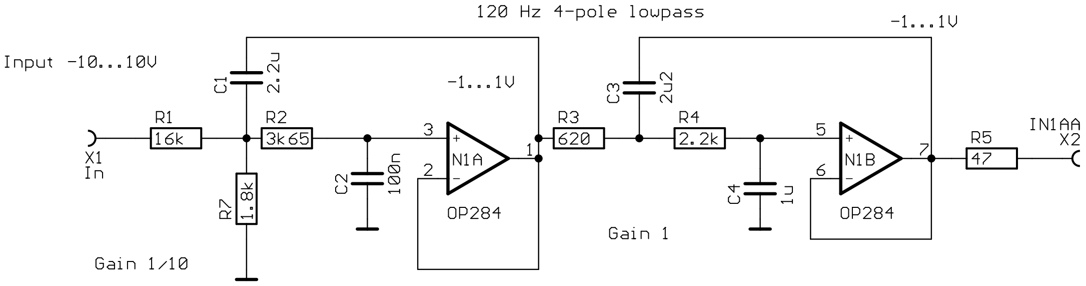
The following diagram shows the input circuit for channels 6 and 7:



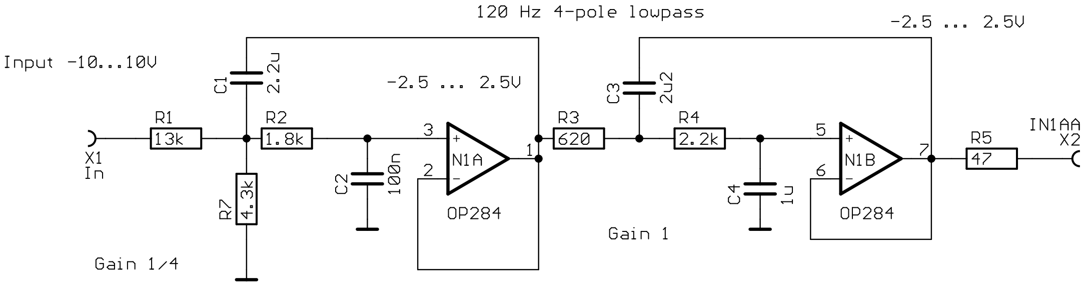
There are 2 D/A converters (Analog Devices DAC8420) which have 4 channels each. Of these 8 analog output channels, at the moment only 6 are used as follows:

Channel	Range	
No.	[V]	Signal
0	-1 . . . +1	M$_1x$ (m1xcoil)
1	-1 . . . +1	M$_2x$ (m2xcoil)
2	-1 . . . +1	M$_3x$ (m3xcoil)
3	-2.5 . . . +2.5	M$_1y$ (m1ypzt)
4	-2.5 . . . +2.5	M$_2y$ (m2ypzt)
5	-2.5 . . . +2.5	M$_3y$ (m3ypzt)

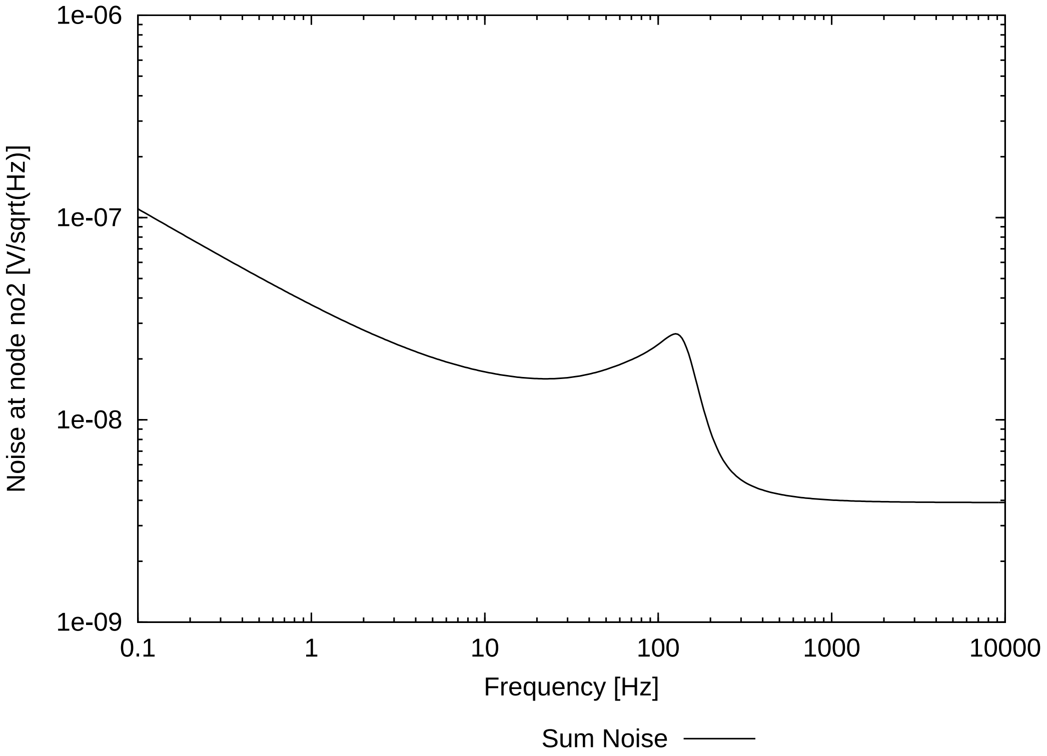
The D/A converters operate with stabilized -10 V and +10 V reference voltages. Hence the output range is -10 V. . . +10 V. Since the required range is only -1 V. . . +1 V or -2.5 V. . . +2.5 V, the output filters have a fixed gain of 1/10 or 1/4. Apart from the gain, they have the same frequency response as the input filters. The following diagram shows the output filter circuit for channels 0, 1 and 2:



The following diagram shows the output filter circuit for channels 3, 4 and 5:



The following diagram shows the output noise spectral density of the output filters. Above 200 Hz, it is dominated by the voltage noise of the final op-amp.



10 Digital Filters

We use recursive digital filters (IIR filters). Here some relevant formulae are summarized. A digital filter transforms an input sequence x_n, which is sampled with a sampling interval T (in our case $T = 1\,\mathrm{ms}$), into an output sequence y_n by the relationship

$$y_i = a_0\,x_i + a_1\,x_{i-1} + \cdots + a_n\,x_{i-n} - b_1\,y_{i-1} - b_2\,y_{i-2} - \cdots - b_m\,y_{i-m}. \tag{23}$$

The transfer function of this filter is given by

$$H(s) = \frac{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}, \tag{24}$$

where

$$z = \exp(sT), \quad s = i\omega = 2\pi i f. \tag{25}$$

Stability of the filter is determined only by the denominator of this equation. The filter is stable if and only if all complex roots of the polynomial equation

$$z^m + b_1 z^{m-1} + b_2 z^{m-2} + \cdots + b_m = 0 \tag{26}$$

are within the unit circle, i.e. $|z| \leq 1$.

For this application we investigate three types of digital filters:

Perfect Integrators These have the simple transfer function $H(s) = \alpha/s$, with gain proportional to $1/f$ and a constant $-90°$ phase shift for all frequencies. Their coeffcients can be directly computed (see Section 10.1 below).

Damped Integrators These are "stopped" integrators with the gain settling at some finite gain and the phase shift returning to zero for higher frequencies. Again their coeffcients can be directly computed (see Section 10.2 below).

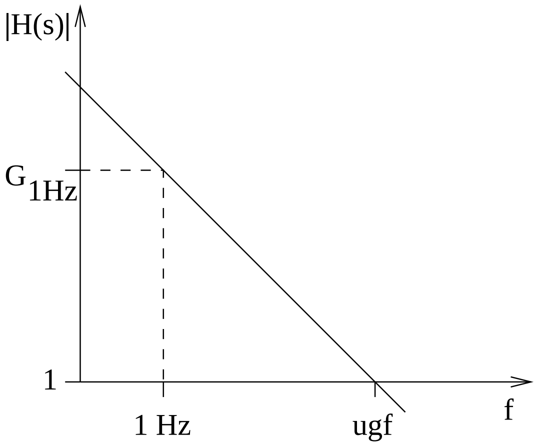
General filters In order to compensate actuator resonances etc., more complicated filters are necessary. They can be computed with LISO (see Section 11 below).

10.1 Perfect Integrators

The transfer function of a perfect integrator is given by

$$H(s) = \frac{\alpha}{s}, \tag{27}$$

and sketched here:



Some relationships (omitting the unit Hz):

$$G_{1\,\mathrm{Hz}} = \frac{\alpha}{2\pi}, \qquad \alpha = G_{1\,\mathrm{Hz}} \cdot 2\pi, \tag{28}$$

$$\mathrm{ugf} = \frac{\alpha}{2\pi}, \qquad \alpha = \mathrm{ugf} \cdot 2\pi. \tag{29}$$

The filter is realized by the recursive relation

$$y_i = a_0 x_i + y_{i-1}, \tag{30}$$

which has the transfer function

$$H(s) = \frac{a_0}{1 - z^{-1}} = \frac{a_0}{1 - \exp(-sT)} \approx \frac{a_0}{1 - (1 - sT)} = \frac{a_0}{sT}. \tag{31}$$

The approximation is valid for small frequencies $f \ll 1/T$. We see that

$$a_0 = \alpha T, \tag{32}$$

and hence

$$a_0 = 2\pi T\, G_{1\,\mathrm{Hz}} = 2\pi T\, \mathrm{ugf}. \tag{33}$$

Thanks to the simple relationship Eq. (30) the filter can be realized efficiently in software, and slew-rate limiting is straightforward by limiting the increment $a_0 x$, which is equivalent to temporarily limiting the gain.

The equation that determines the stability,

$$z - 1 = 0, \tag{34}$$

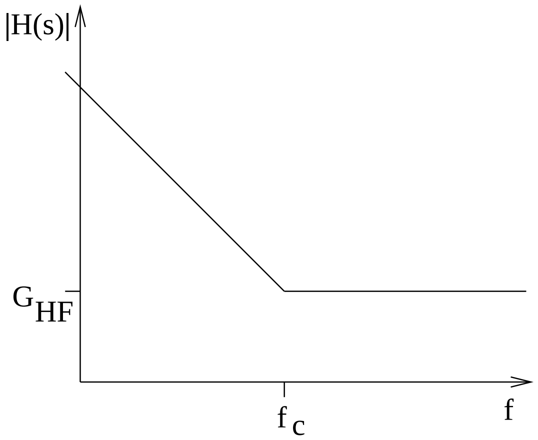
has the single root $z = 1$. Hence the filter is on the edge of the stability region $|z| \leq 1$. Indeed any tiny offset will be integrated forever and the output will increase linearly without limit, if there is no external negative feedback (as there is in a closed-loop application).

10.2 Damped Integrators

The transfer function of a damped integrator is given by

$$H(s) = G_\infty \left(1 + \frac{2\pi f_c}{s}\right) \tag{35}$$

and shown here:



It is characterized by the gain for high frequenices, G_∞, and the corner frequency f_c, and can be realized by the recursive relation

$$y_i = a_0\, x_i + a_1\, x_{i-1} + y_{i-1}, \tag{36}$$

which has the transfer function

$$H(s) = \frac{a_0 + a_1 z^{-1}}{1 - z^{-1}} = \frac{a_0 + a_1 \exp(-sT)}{1 - \exp(-sT)}. \tag{37}$$

The approximation for small frequencies by expanding the exponential function yields:

$$H(s) \approx \frac{a_0 + a_1 - a_1 sT}{sT} = -a_1 + \frac{a_0 + a_1}{sT}. \tag{38}$$

By comparing equations (35) and (38) we find

$$a_1 = -G_\infty, \qquad a_0 = G_\infty(1 + 2\pi T f_c). \tag{39}$$

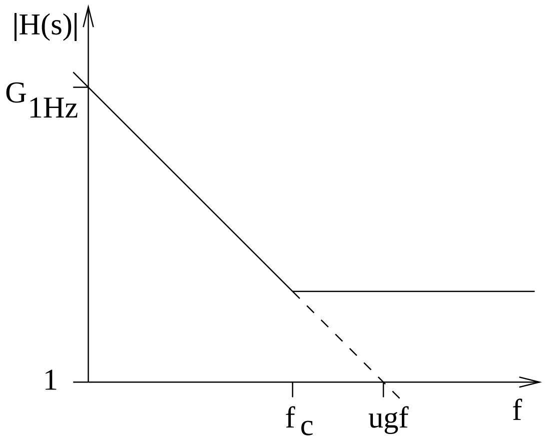
Again this filter can be implemented efficiently in software, and slew-rate limiting can be realized by limiting the increment $a_0\, x_i + a_1\, x_{i-1}$. Of course the transfer function (which is defined only for linear systems) is corrupted in the case of slew-rate limiting. The stability is the same as discussed in Section 10.1 above.

10.3 Damped Integrators - alternative formulae

For our purpose different formulae are more convenient. The transfer function is characterized by two parameters: (1) the corner frequency f_c, and (2) the gain at 1 Hz, $G_{1\,\mathrm{Hz}}$, or the unity gain frequency, 'ugf', **if there were no corner**. The numerical values of $G_{1\,\mathrm{Hz}}$ and 'ugf' are the same, and both can be used equivalently. We will use 'ugf' in the following. The transfer function is written as:

$$H(s) = \frac{2\pi\,\mathrm{ugf}}{s}\left(1 + \frac{s}{2\pi\, f_c}\right) = \frac{2\pi\,\mathrm{ugf}}{s} + \frac{\mathrm{ugf}}{f_c} \tag{40}$$

and shown here:



We also rewrite the recursive relation

$$y_i = a_0\, x_i + a_1\, x_{i-1} + y_{i-1}, \tag{41}$$

as

$$y_i = y_{i-1} + \Delta y, \tag{42}$$

with

$$\Delta y = a_0\, x_i + a_1\, x_{i-1}. \tag{43}$$

We write the coefficients a_0 and a_1 as

$$a_0 = a + \Delta a, \quad a_1 = -a. \tag{44}$$

Furthermore we introduce the increment in the input signal:

$$\Delta x = x_0 - x_1. \tag{45}$$

We obtain for the recursive relation

$$\Delta y = x_0 \Delta a + a \Delta x, \tag{46}$$

and for the coefficients

$$a = \frac{\mathrm{ugf}}{f_c}, \quad \Delta a = 2\pi \, \mathrm{ugf} \, T. \tag{47}$$

The first term of equation (46), $x_0 \Delta a$, is the same as for the perfect integrator, while the seond term, $a\Delta x$, corrects for the corner frequency f_c. This form is more suited to the integer arithmetic with its intrinsic limited dynamic range than the form (36) which involves forming the small difference between two large numbers.

11 Some notes on IIR filters on the H3048F

In this section I describe some attempts at realizing more complex IIR filters with the 3048F processor. Although theses attempts finally failed, the information may be useful for future designs.

11.1 Purpose

Looking at the transfer functions in Section 6 it is clear that without special filters the unity gain frequency (UGF) will be limited to 1 or 2 Hz for each loop due to actuator resonances. Originally I had intended to obtain a UGF of around 20 Hz for better suppression of the natural motion of the mirrors. The transfer functions are well defined up to that frequency and show only a few “clean” resonances each, which need to be compensated with filters.

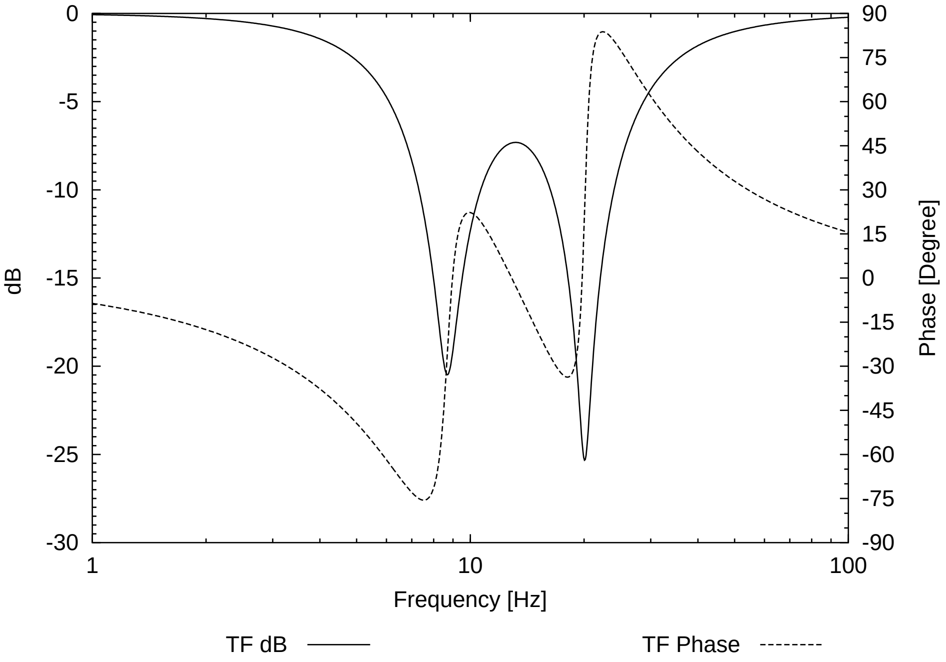
The plan was to use digital IIR filters inside the 3048F for that purpose. The main advantage of doing so is that the final output signal is under control of the microprocessor, enabling slew-rate limiting, a perfect “hold” function etc. If, on the other hand, the compensation filters are realized as analog external filters, the microprocessor has no way to achieve these desirable features.

In the following I will use the input mirror pitch PZT transfer function from Section 6.3 as example. It has two resonances at 8.7 Hz and 20 Hz with a Q of 9.3 and 16.3, respectively. The sampling frequency is assumed to be

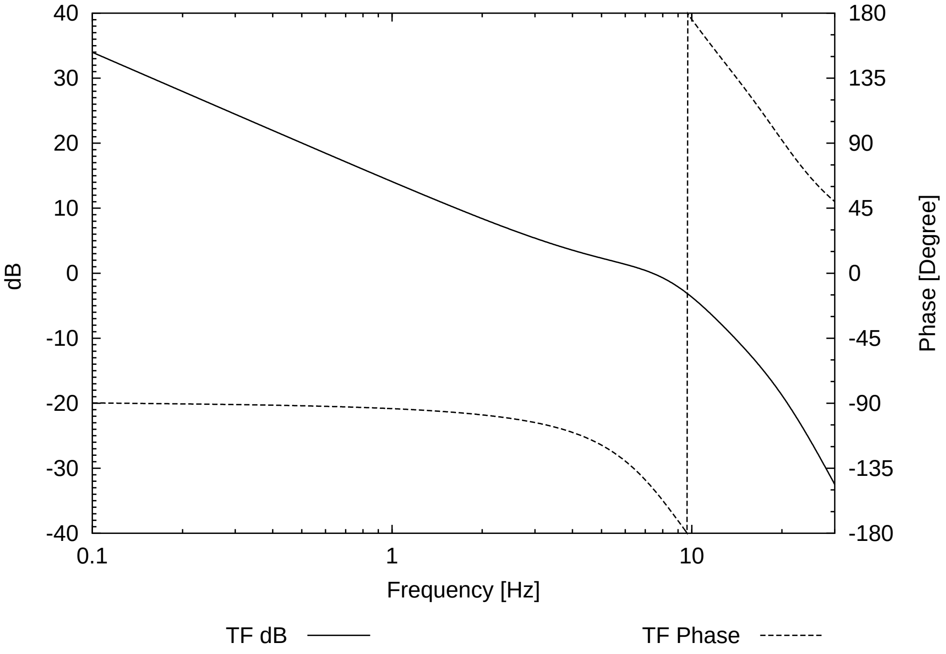
$$f_{\mathrm{samp}} = 1000\,\mathrm{Hz}. \tag{48}$$

The first idea to compensate these resonances is to introduce zeroes at these frequencies. However, without introducing new poles the required filter transfer function would rise as f^4 above 20 Hz, which introduces considerable noise and dynamic range problems. Hence the filter I discuss here introduces new poles of $Q = 1$ at the same frequencies of the old poles, i.e. effectively reducing the Q of the poles to 1 without changing their frequency. The following plot shows the filter transfer function. It was generated by LISO with

```
zero 8.688 9.34
zero 20.08 16.34
pole 8.688 1
pole 20.08 1
```



With such a compensation filter, a UGF of around 7 Hz could have been achieved by using an integrator active between DC and 10 Hz:



11.2 Design of IIR filters with LISO

In order to design these IIR filters, LISO was extended (version 1.81). The following input file (called `m1yloop.fil`)

```
zero 8.688 9.34
zero 20.08  16.34
pole 8.688 1
pole 20.08  1
freq log .5 200 300
iir 1000 start 5 5
```

produces initial values for the filter (in a file called `m1yloop_iir.fil`) which are then refined by running LISO on that file, giving the following coefficients:

$$a_0 = 0.80903621 \tag{49}$$
$$a_1 = -3.744904 \tag{50}$$
$$a_2 = 6.9131052 \tag{51}$$
$$a_3 = -6.3548303 \tag{52}$$
$$a_4 = 2.904875 \tag{53}$$
$$a_5 = -0.5272688 \tag{54}$$
$$b_0 = 1.0 \tag{55}$$
$$b_1 = -4.5362002 \tag{56}$$
$$b_2 = 8.2212406 \tag{57}$$
$$b_3 = -7.4378974 \tag{58}$$
$$b_4 = 3.3572789 \tag{59}$$
$$b_5 = -0.6044083 \tag{60}$$

and a transfer function that is reasonably close to the desired one. However, it turns out that the required precision of the coefficients is of the order of $10^{-6} \dots 10^{-7}$, which is hard to achieve with integer arithmetic. It was found that the filter can be realized more easily by cascading two filters that compensate one pole each. Therefore in the following the compensation filter that handles the first resonance at 8.7 Hz will be investigated. It can be designed by running LISO with the following input file (called `m1yl1.fil`)

```
zero 8.688 9.34
pole 8.688 1
freq log 1 100 200
iir 1000 start
```

and then running the automatically produced file (called `m1yl1_iir.fil`). This second run of LISO uses the fitting algorithm to refine the analytical initial values, requiring stability of the filter as a constraint. The coefficients are then rounded to

16-bit integers, trying various denominators and rounding up or down, and finally selecting those integer coefficients that produce the best transfer function. The result is the following filter:

$$y_i = \frac{8216\,x_i - 16360\,x_{i-1} + 8168\,x_{i-2}}{8416} - \frac{-22017\,y_{i-1} + 10723\,y_{i-2}}{11326}. \tag{61}$$

The computed transfer function of that filter (by Equation (24)) is close to the desired one, and it is also stable. After arriving at this point I had believed that the required filters *can* be realized in software, after all.

11.3 Limitations of the 3048F

Experience from earlier projects and preliminary tests of this circuit and software revealed the following limitations of the 3048F for the present application:

Arithmetic with sufficient speed is limited to (signed) integer variables. Although the gnu C compiler provides a floating point library (with 32-bit single precision `float` variables and all usual functions), these are far too slow.

Furthermore, multiply and divide operations are limited to the hardware `MULXS.W` and `DIVXS.W` instructions, which provide a 16 × 16 → 32 bit multiplication and a 32/16 → 16 bit division in 24 cyles (1.5 μsec) each.

Adding and subtracting 32-bit numbers is no problem (2 cyles = 0.125 μsec for a register–register operation). However, multiply and divide operations for 32-bit numbers require many instructions and need at least 10 μsec for multiplication and 50 μsec for division.

In order to use the 32/16 → 16 bit `DIVXS.W` instruction, it is necessary to provide assembler code for the divide operation as follows:

```
inline int
mydiv (long a, int b)
{
  long result;
   asm ("divxs.w %1,%0": "=r" (result): "r" (b), "0" (a):"cc");
  return (int) result;
}


...
c=mydiv(a,b);
...
```

Otherwise the compiler will issue a subroutine call to the full 32-bit division routine.

The same applies for the 16 × 16 → 32 bit `MULXS.W` instruction:

```
inline long
mymul (int a, int b)
{
  long result;
asm ("mulxs.w %1,%0": "=r" (result): "r" (b), "0" (a):"cc");
  return result;
}
```

Although the gnu C compiler most often issues the correct `MULXS.W` instruction by itself, I have observed some instances when a subroutine call is used. To prevent this unpredictable and potentially time-consuming behaviour, it is best to use the above `mymul` instruction.

Hence the filter operations are limited to 16-bit filter coefficients and 16-bit input and output values. Intermediate products and accumulators can use 32 bit, if only add, subtract or 32/16 → 16 bit division operations are used.

The example (61) above was designed taking into account these limitations.

Real timing data was obtained by programming the prototype, toggling digital output bits and measuring the time with an oscilloscope. The results are (for 6 input channels A/D):

Getting the data and subtracting the offset: 10 μsec for 6 channels.

Matrix operations on 16-bit integers to seperate the error signals: 118 μsec for 6 channels.

DAC output of 6 channels: 108 μsec for 6 channels.

IIR filter such as Equation (61): 50 μsec for **one channel**

The A/D converter was running continuously during these measurements, updating 16-bit accumulators via an interrupt handler. Once per sampling interval (1000 μsec) the accumulators are divided by the number of conversions and made available as input to the main program. This operation consumes around 10 . . . 20% of the CPU time.

The above results show that with filters of the type (61), operation with 6 channels at 1000 Hz should be possible. On the other hand, more complicated filters (using 32 bit or floating point variables) are out of the question.

The proposed sampling frequency of 1000 Hz may seem high for 20 Hz UGF. However, the phase shifts introduced by the the antialiasing filters (see Section 9) and those inherent in the digital filter sampling process prohibit much slower sampling.

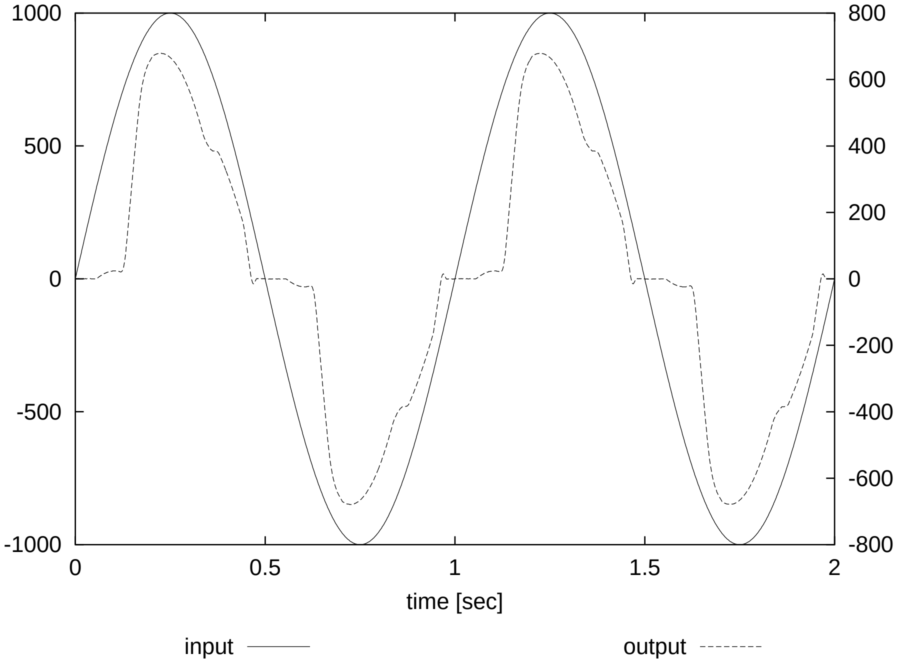
11.4 Performance of the IIR filter

The IIR filter (61) was implemented and tested in the prototype. At first the performance seemed satisfactory, i.e. a transfer function measured with the spectrum analyzer corresponded to the prediction. More detailed tests, however, revealed a severe problem:

The filter does not react well to small input signals (e.g. a few % of the maximal amplitude).

To analyze this problem, LISO was further enhanced to simulate the integer operations of the filter. The following operations are done: For all interesting frequencies, first the maximum amplitude is determined that does not cause overflow in any intermediate result. For the example discussed here, that amplitude was 23675.[3] Next, the transfer function is determined by letting the filter operate (with integer operations) on an input sine wave, and 'demodulating' the output by multiplication with $\sin(\omega\, t)$ and $\cos(\omega\, t)$ (after allowing for settling for a few periods of the input signal). The input amplitude is then reduced in steps until the transfer function deviates by 3 dB (taking into account the phase also) from the initial transfer function that was found at big amplitudes. The smallest amplitude thus found is considered the lower limit of permissible input signals.

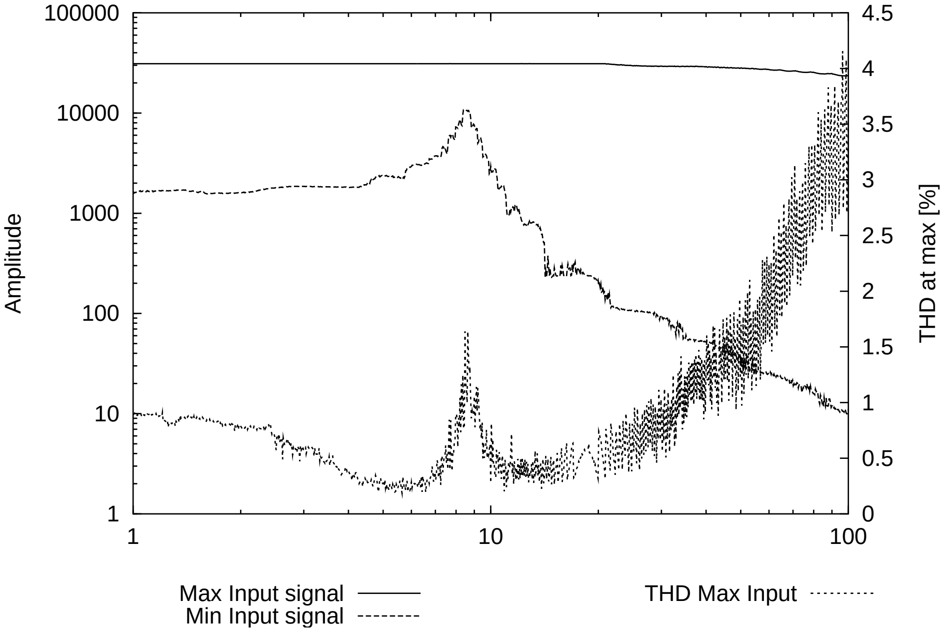
For the present example, this lower limit turned out to be 1560 at 1 Hz (i.e. well below the 8.7 Hz resonance). Here is what happens to a 1 Hz signal of amplitude 1000 (i.e. 4.2% of the full range) in the time domain:



Clearly there is considerable crossover distortion. If the input signal amplitude is further decreased the response deteriorates rapidly and soon vanishes alltogether.

The following plot shows the minimal and maximal input amplitudes for the filter (61):

[3] The limit was caused by the second fraction of Equation (61) for 100 Hz input signals.



The top curve (referred to the left y-axis) is the maximal signal that avoids overflow, the next curve (also referred to the left y-axis) is the minimal signal that yields a deviation of 3 dB in the transfer function, and the bottom curve (referred to the right y-axis) is the total harmonic distortion for a signal of maximal amplitude.

The cause of the problem lies in equation (61), shown again here in floating-point form:

$$y_i = 0.9762\, x_i - 1.9439\, x_{i-1} + 0.9705\, x_{i-2} + 1.9439\, y_{i-1} - 0.9467\, y_{i-2}. \tag{62}$$

The coefficients of x_i, x_{i-1} and x_{i-2} nearly cancel each other. Hence the transmission of a slow signal from the input to the output relies on a signal appearing in the output. For small amplitudes, that build-up of the output signal does not take place, and the filter collapses. This effect is illustrated in the following table:

t	in$*$	out$*$	in	out	a_0	a_1	a_2	b_1	b_2
0.000	0.000	0.000	0	0	0	0	0	0	0
0.001	0.628	0.613	1	0	0	0	0	0	0
0.002	1.257	1.198	1	0	0	-1	0	0	0
0.003	1.885	1.755	2	0	1	-1	0	0	0
0.004	2.513	2.286	3	0	2	-3	0	0	0
0.005	3.141	2.793	3	0	2	-5	1	0	0
0.006	3.769	3.278	4	0	3	-5	2	0	0
0.007	4.397	3.742	4	0	3	-7	2	0	0
0.008	5.024	4.186	5	0	4	-7	3	0	0
0.009	5.652	4.613	6	0	5	-9	3	0	0
0.010	6.279	5.023	6	0	5	-11	4	0	0
0.011	6.906	5.419	7	0	6	-11	5	0	0
0.012	7.533	5.801	8	0	7	-13	5	0	0
0.013	8.159	6.171	8	0	7	-15	6	0	0
0.014	8.785	6.531	9	0	8	-15	7	0	0
0.015	9.411	6.881	9	0	8	-17	7	0	0
0.016	10.036	7.224	10	1	9	-17	8	0	0
0.017	10.661	7.559	11	1	10	-19	8	1	0
0.018	11.286	7.889	11	0	10	-21	9	1	0
0.019	11.910	8.214	12	1	11	-21	10	0	0
0.020	12.533	8.536	13	1	12	-23	10	1	0
0.021	13.156	8.855	13	0	12	-25	11	1	0
0.022	13.779	9.173	14	1	13	-25	12	0	0
0.023	14.401	9.491	14	1	13	-27	12	1	0
0.024	15.023	9.808	15	1	14	-27	13	1	0
0.025	15.643	10.127	16	0	15	-29	13	1	0
0.026	16.264	10.448	16	0	15	-31	14	0	0
0.027	16.883	10.772	17	1	16	-31	15	0	0
0.028	17.502	11.099	18	1	17	-33	15	1	0
0.029	18.121	11.430	18	0	17	-34	16	1	0
0.030	18.738	11.765	19	1	18	-34	17	0	0

t is the time in msec, "in$*$" a (floating point) input signal of 1 Hz and amplitude 100 and "out$*$" the output signal of the filter (62), implemented in floating point. The rest of the table refers to the filter (61), implemented in integer arithmetic. "in" is the input signal, "out" the output. The columns labelled "a_0" to "b_2" show the individual contributions to the output signal, as given in Equation (61). In the table, they don't add up exactly, because the output signal is computed as

$$y_i = \frac{8216\,x_i - 16360\,x_{i-1} + 8168\,x_{i-2}}{8416} - \frac{-22017\,y_{i-1} + 10723\,y_{i-2}}{11326}, \tag{63}$$

whereas the table entries are shown as

$$\frac{8216\,x_i}{8416}, \quad \frac{-16360\,x_{i-1}}{8416}, \quad \frac{8168\,x_{i-2}}{8416}, \quad \frac{-22017\,y_{i-1}}{11326}, \quad \frac{10723\,y_{i-2}}{11326}, \tag{64}$$

respectively. Due to the near cancellation of the coefficients $a_0 + a_1 + a_2$, the output signal fails to build up in the presence of rounding errors. The DC gain, however, relies on an output signal building up, and hence the filter completely fails.

11.5 Attempts to improve the IIR filter

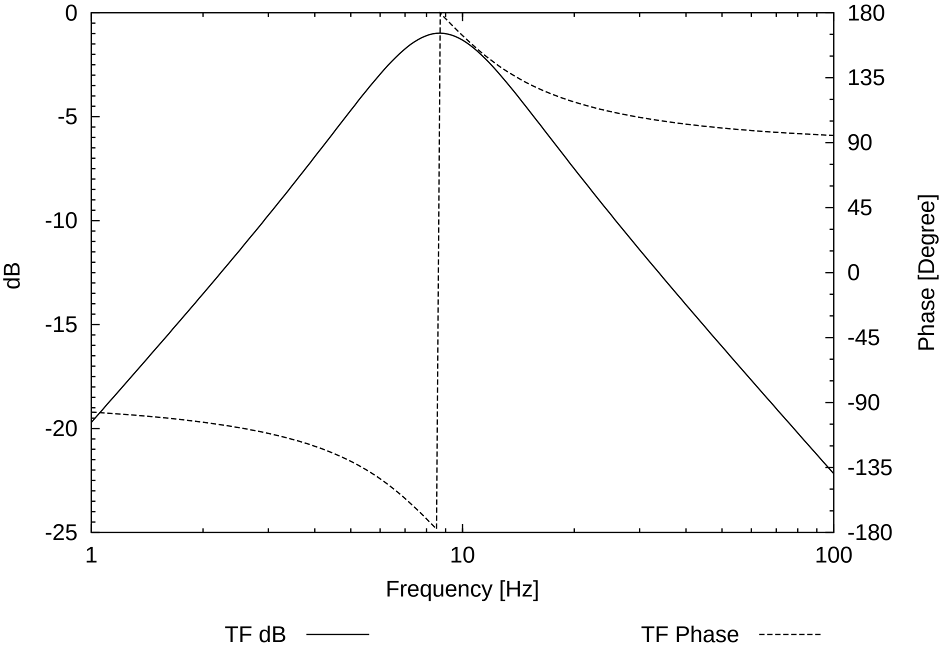
After the failure described in the previous section, I have tried to improve the filter by removing the DC gain, i.e. making an IIR filter that implements the transfer function

$$H'(s) = 1 - H(s), \tag{65}$$

such that the real output signal is found by

$$\text{output} = \text{input} + H'(s)(\text{input}). \tag{66}$$

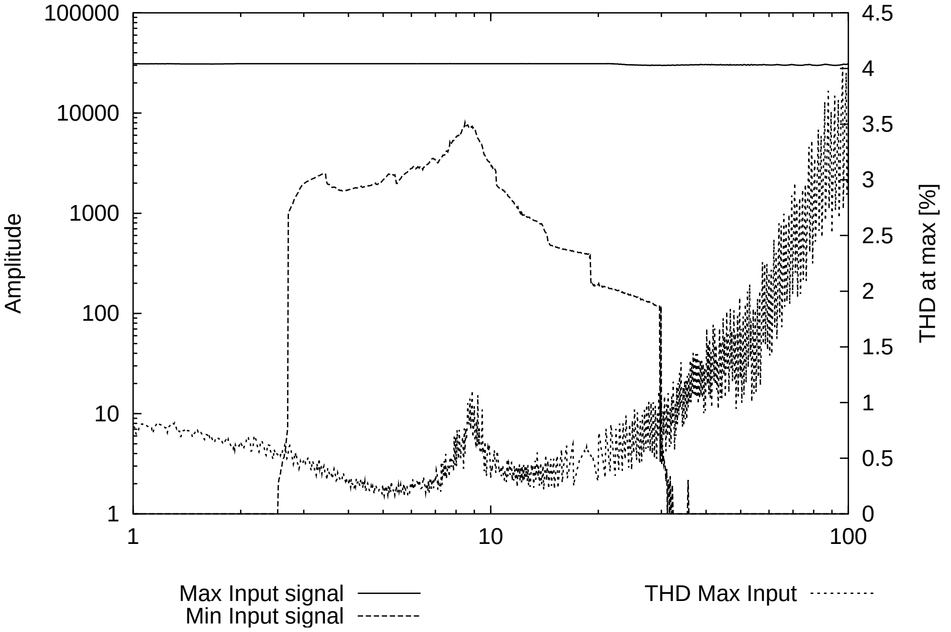
The hope was that a DC signal would be transferred directly to the output, bypassing the IIR filter, which would only become active for signals near the resonance frequency. The following plot shows the transfer function $H'(s)$:



LISO was extended once more to yield the following filter:

$$y_i = \frac{-778\, x_i + 0\, x_{i-1} + 778\, x_{i-2}}{32767} - \frac{-22053\, y_{i-1} + 10741\, y_{i-2}}{11345}. \tag{67}$$

The following plot shows again the minimal and maximal input signals for that filter (including the DC bypass), together with the harmonic distortion:



Although the performance for DC is now good, as expected, the performance remains poor for signals near the resonance. The sharp change of the minimal input signal is caused by the somewhat arbitrary limit of 3 dB deviation in the transfer function, which is not at all reached for signals far enough from the resonance.

In another attempt to improve the performance for small signals, I have modified the integer-rounding routine of LISO such that both the a_i and the b_i use the same denominator. For the example above (61) that yields

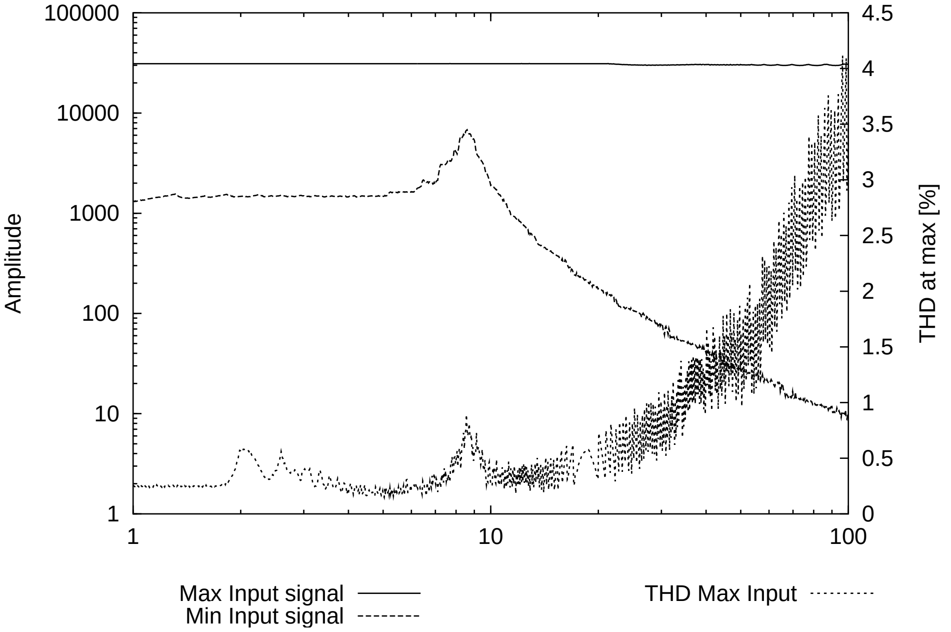
$$y_i = \frac{8211\,x_i - 16350\,x_{i-1} + 8163\,x_{i-2} + 16250\,y_{i-1} - 7963\,y_{i-2}}{8411}. \tag{68}$$

Doing so allows to use 32-bit integer variables for intermediate results:

$$y'_i = 8411\,y_i = 8211\,x_i - 16350\,x_{i-1} + 8163\,x_{i-2} + \frac{16250\,y'_{i-1} - 7963\,y'_{i-2}}{8411}, \tag{69}$$

storing the higher-precision variables y'_i instead of y_i and computing the final output as $y'_i/8411$. However, in computing the fractional term of equation (69), the division by 8411 must be carried out *before* the multiplication with 16250 or −7963, in order to prevent overflow of the 32-bit variable. In this step again precision is lost.

Although this technique yields a small improvement in the small-signal performance at moderate increase in computing cost, the result is still not good enough:



11.6 Conclusion

I have concluded that **it is not possible to implement compensation filters with 16-bit integer arithmetic for the present project.**

Since the design of IIR filters itself is no problem, there is no fundamental problem in using them. For future designs I would try to find a suitable processor which has floating point arithmetic (e.g. a DSP or something like a 80486DX with 64/80-bit floating point coprocessor). It is, however, questionable whether such processors have all the convenient I/O features of the 3048F microcontroller which are used for controlling the LCD, input switches, serial connection to the DAC etc.

12 Scaling

Being limited to integer arithmetic, the question of scaling becomes very important in the design of software. Here I describe the scaling that I have used.

12.1 A/D conversion

The A/D converter yields 10-bit unsigned results, multiplied by 64, i.e. numbers of the binary format $xxxx\,xxxx\,xx00\,0000$, with values between 0 and $64 \times 1023 = 65472$.

In previous projects it has been found that by averaging the effective resolution can be increased to about 12 bits, which is also the resolution of the D/A converter.

When 8 channels are sampled at 1000 Hz, approximately 8 samples can be averaged in each sampling period for each channel.

Hence the primary interrupt handler, `adint1.c`, which is executed every time all 8 input channels have been sampled, divides each result by 16 and adds it into 16-bit unsigned accumulators (one for each channel), up to 16 times.

The second interrupt handler, `adint2.c`, called once every sampling interval, divides the accumulators by the number of samples (1...16). For the bipolar channels (0...5), 2046 is subtracted. The result is hence a number in the range 0...4092 for the unipolar channels (6 and 7), and in the range −2046...2046 for the bipolar channels.

The input range of channels 0...3 is $-0.648 \cdots + 0.648\,\mathrm{V}$, while that for channels 4 and 5 is $-3.35 \cdots + 3.35\,\mathrm{V}$.

The compensation of the input range (multiplication by 0.648 or 3.35 included in the matrix below) achieves a uniform value of the least significant bit for all of the 6 input channels:

$$\mathrm{LSB} = \frac{1\,\mathrm{V}}{2048}, \tag{70}$$

which is the same as for the output D/A conversion.

12.2 Normalization

The error signals are proportional to light power levels. In particular, channels 0...3 (X_I, X_Q, Y_I, Y_Q) are proportional to

$$c_1 = \sqrt{P_L \times P_I}, \tag{71}$$

where P_L is the incident laser power and P_I the power in the cavity.

Channels 4 and 5 (X_E and Y_E) are proportional to

$$c_2 = P_I. \tag{72}$$

It is possible to compute c_1 in software. However, the conversion integer → floating point, the square root and the conversion floating point → integer consume too much time (between 300 and 600 μsec depending on the value) such that floating point arithmetic cannot be used.

Therefore the following integer-to-integer square root subroutine is used. For an `unsigned long` input integer a between 0 and $4095 \times 4095 = 16769025$, it computes the largest integer x that is not greater than $\sqrt{a}$, i.e. $x = \mathrm{floor}(\sqrt{a})$. For example, $a = 8$ yields $x = 2$, while $a = 9$ yields $x = 3$.

```
unsigned short
mysqrt (unsigned long a)
{
  unsigned short bit = 1 << 11;
  unsigned short res = 0, test;
  if (a == 0)
    return 0;
  if (a >= 4095UL * 4095UL)
    return 4095;
  do
  {
   test = res | bit;
   if ((unsigned long) test * (unsigned long) test <= a)
    res = test;
   bit >>= 1;
  }
 while (bit);
 return res;
}
```

The routine works by successively testing each bit of the result, starting from the highest bit, squaring the value under test and comparing it to the input, very similar to how a "successive-approximation" A/D-converter works. It takes 12 multiplications (1.5 μsec each) and some register-register operations. The running time is xxx μsec.

The error signals are then normalized by

$$e' = \frac{e \times 3996}{P}, \tag{73}$$

where P is either the power internal to the cavity (A/D channel 7 with range 0...4092) or the square root described above (range 0...4095). The value 3996 corresponds to the limit when the power level is considered to be "too big" (i.e. 97.6 % of full scale).

12.3 Matrix

The next step applies the matrix derived in Section 5 (without the scaling factors used there) in order to separate the error signals. At the same time, both the different input ranges and the detector efficiency derived in Section 7 are compensated for. The reference for the input ranges is ±1 V, and the detector efficiencies are referred to the end detector (as in Section 7).

The error signals for each mirror are obtained from the input error signals from each detector as follows:

$$\begin{pmatrix} \alpha_\mathrm{a} \\ \alpha_\mathrm{b} \\ \alpha_\mathrm{c} \end{pmatrix} = A_x \times \begin{pmatrix} X_I \\ X_Q \\ X_E \end{pmatrix} \tag{74}$$

$$\begin{pmatrix} \beta_\mathrm{a} \\ \beta_\mathrm{b} \\ \beta_\mathrm{c} \end{pmatrix} = A_y \times \begin{pmatrix} Y_I \\ Y_Q \\ Y_E \end{pmatrix} \tag{75}$$

$$\begin{aligned} A_x &= \begin{pmatrix} -0.5042 & 0.4303 & 0.3015 \\ -0.5042 & -0.3015 & 0.3015 \\ -0.0096 & 0.0452 & 0.2115 \end{pmatrix} \times \begin{pmatrix} \frac{0.648}{0.371443} & 0 & 0 \\ 0 & \frac{0.648}{-0.274189} & 0 \\ 0 & 0 & 3.35 \end{pmatrix} \\ &= \frac{1}{4096} \begin{pmatrix} -3603 & -4165 & 4137 \\ -3603 & 2918 & 4137 \\ -69 & -437 & 2902 \end{pmatrix} \end{aligned} \tag{76}$$

$$\begin{aligned} A_y &= \begin{pmatrix} 0.7120 & 0.4242 & -0.4242 \\ -0.7095 & 0.4242 & -0.4242 \\ -0.0006 & -0.6866 & 0.2115 \end{pmatrix} \times \begin{pmatrix} \frac{0.648}{-0.313443} & 0 & 0 \\ 0 & \frac{0.648}{0.270632} & 0 \\ 0 & 0 & 3.35 \end{pmatrix} \\ &= \frac{1}{4096} \begin{pmatrix} -6029 & 4160 & -5820 \\ 6008 & 4160 & -5820 \\ 5 & -6734 & 2903 \end{pmatrix} \end{aligned} \tag{77}$$

12.4 Integrator

Because of the above-mentioned difficulties with IIR compensation filters, the circuit uses a damped integrator loop filter (see Section 40 above) and external analog compensation filters (see Section 13 below). The nominal unity-gain frequency (ugf) is made user-selectable between 0.1 and 100 Hz, and the corner frequency (f_c) between 0.1 and 20 Hz.

The damped integrator filter is realized by equations (46) and (47). In this step the actuator efficiencies are also taken into account.

In order to accomodate a wide range of gain constants and to allow accurate integration even of small input signals, the accumulator is implemented as a (signed) 32-bit variable. The full range of that variable is used. Since both input and output resolution are 12 bit, the input must be scaled by 2^{20}.

The coefficients a and Δa are precomputed as follows:

$$a = 2^{20} \frac{\mathrm{ugf}}{f_c} \frac{1}{A_{\mathrm{act}}} \tag{78}$$

$$\Delta a = 2^{20}\, 2\pi\, \mathrm{ugf}\, T \frac{1}{A_{\mathrm{act}}} \tag{79}$$

Because the range of a or Δa may exceed the range of an integer variable, both a and Δa are computed and stored as ‘long’ variables (32 bit). The multiplication generated by the gcc compiler uses bitwise shift operations and is too slow. Because in our case one multiplicand has only 16 bits, a fast multiplication routine was written in assembler:

```
;   long smul (long x, int y)
;     er0        er0    r1
;
.h8300h
.section .text
.align 1
.global _smul
_smul:
sub.b r3l,r3l    ; clear sign flag
mov.l er0,er0    ; test sign of x
bge .L3        ; if x positive
```

```
neg er0        ; x is negative, change sign
not.b r3l        ; invert flag (set)
.L3:
mov.w r1,r1      ; test sign of y
bge .L4        ; if y positive
neg r1         ; y is negative, invert y
not.b r3l        ; invert flag
.L4:
mov.w r0,r2      ; r2=x.lsw
mov.w e0,r0      ; r0=x.msw
mulxu.w r1,er0     ; er0=y*x.msw
mulxu.w r1,er2     ; er2=y*x.lsw
mov.w r0,e0      ; e0=(y*x.msw)lsw
sub.w r0,r0      ; er0=y*x.msw<<16
add.l er2,er0    ; er0=result
mov.b r3l,r3l    ; test flag
beq .L7        ; if 0 exit
neg er0        ; invert result
.L7:
rts
.end
```

12.5 Slew-rate limiting

In previous circuits if was found useful to limit the rate of change of the output (slew-rate), in order to prevent sudden strong motions of the suspended mirrors. The limit is expressed as %/sec. A signed 32-bit variable is precomputed which is compared to the increment computed by the integrator. If the increment is too big, it is set to the limit. Since to full range of 100 % corresponds to 2^{31} in the accumulator, the limit is computed as

$$\text{srlim} = 2^{31}\, T \frac{x}{100\,\%} \tag{80}$$

The limiting is done by the following subroutine:

```
int
limit (long *var, long limit)
/*  limit *var to the range -limit <= *var <= *limit,
returns 1 of cropping was active, 0 otherwise */
{
  if (*var > limit)
    {
      *var = limit;
      return 1;
    }
  else if (*var < -limit)
```

```
    {
      *var = -limit;
      return 1;
    }
  return 0;
}
```

The return value of 0 or 1 is used for the LED display.

A new version of the software (installed March 5th, 2001) increases the slew-rate limit gradually to prevent loss of lock when there is a big initial misalignment and the loops are turned on.

For measuring time intervals, the main loop is used, which is executed once every millisecond. An integer variable is reset at the appropriate moment and then incremented in the loop, and hence represents the time in msec.

The slew-rate is increased in 100 steps. In the initialization routine, the increment (1/100 of the final slew-rate) is precomputed and saved as `srstep[0]` ... `srstep[5]` for each of the 6 output channels.

The state of the slew-rate for each channel is stored in integer variables `srstate[0]` ... `srstate[2]` for output channels 1/4, 2/5 and 3/6 respectively. The slew-rate is reset to zero:

- When the circuit is restarted,
- When lock is lost,
- When the respective loop is turned off.

When the loop is back in operation, the slew rate is increased by the precomputed increment every x msec, where x is the ramp-time in seconds multiplied by 10. *Example:* If the ramp time is 5 sec, x is 50 and the slew-rate is incremented once every 50 msec. This happens 100 times during 5 seconds until the slew-rate reaches its final value.

12.6 Output

After slew-rate limiting, the increment is added to the accumulator by using the following saturated addition routine, which prevents a positive overflow to yield a huge negative number, and vice-versa. It also returns 0 or 1 to indicate whether an overflow occured. This information is displayed with an LED.

```
int
satadd (long *sum, long delta)
/* addition with saturation  returns 1 if overflow, 0 if ok */
{
  if (*sum > 0 && delta > 0 && *sum + delta < 0)
    {
      *sum = LONG_MAX;
      return 1;
    }
  else if (*sum < 0 && delta < 0 && *sum + delta > 0)
    {
      *sum = LONG_MIN;
      return 1;
    }
  else
    *sum += delta;
  return 0;
}
```

Only the top 12 bits of the accumulator are fed to the DAC (after adding a bias of 2048). The remaining 20 bits are an accuracy reserve, ensuring that small input signals get accurately integrated over long timescales.

13 Compensation filters

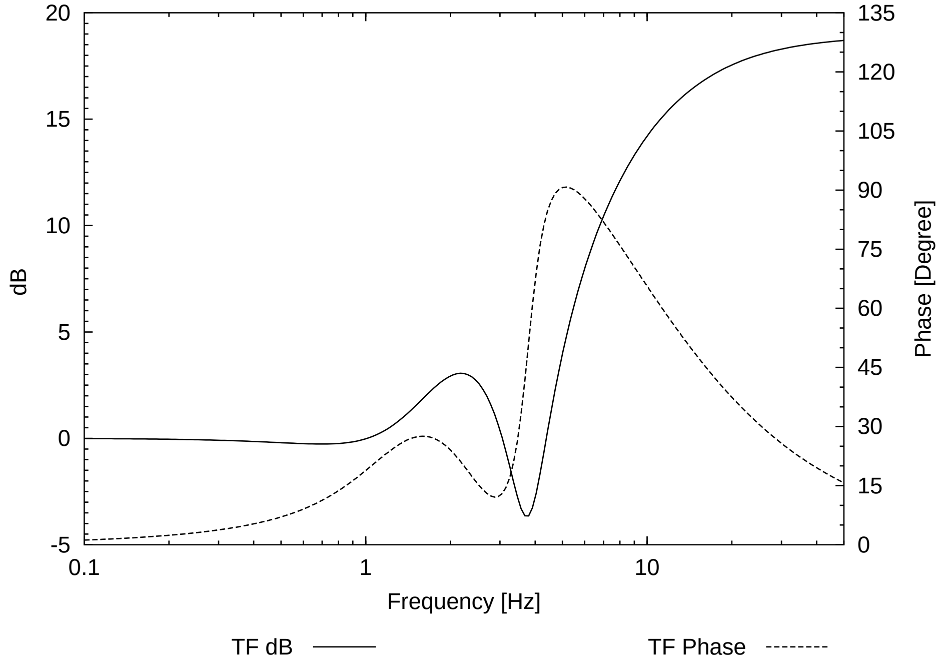
The first version of the circuit employed a simple integrator as loop filter and no compensation filters. Since the unity gain frequency was too low, the second version includes the damped integrator described above, and analog external compensation filters which are described in this section, channel by channel.

13.1 Input Mirror Yaw coil ("m1xcoil")

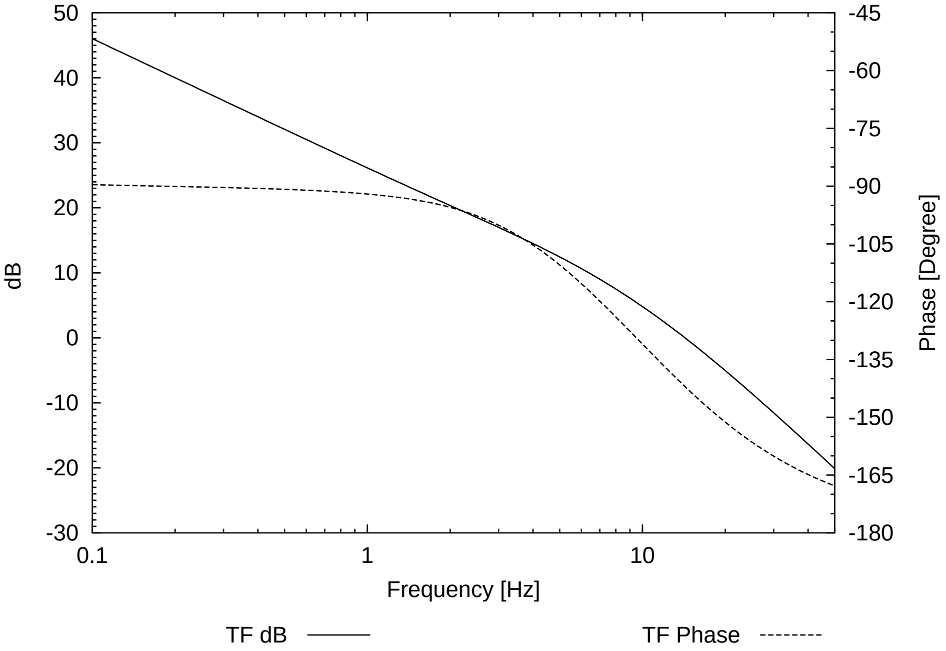
The filter is composed of two biquadratic sections as follows:

Section	Pole/Zero	*f* [Hz]	Q
1	zero	1.372	0.755
1	pole	2.508	0.84
2	zero	3.777	3.455
2	pole	3.777	(real)
2	pole	10.0	(real)

The following figure shows the transfer function of the compensation filter:

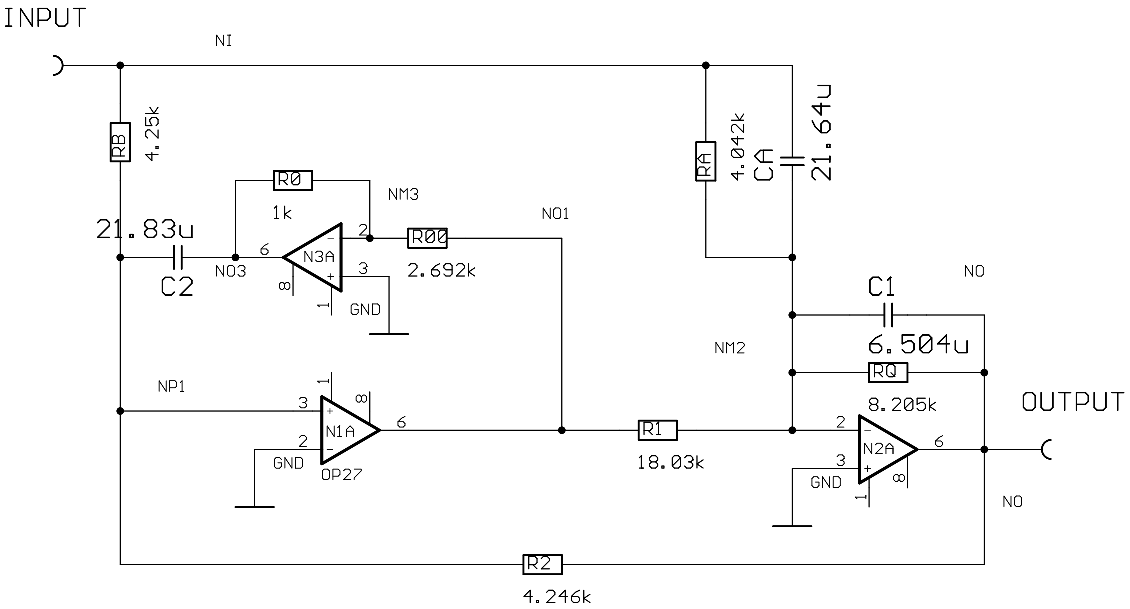


The following figure shows the predicted open-loop gain,without taking into account the additional phase delays by the antialiasing filter and the digital loop. A damped integrator active between DC and 3 Hz is included in the computed transfer function.

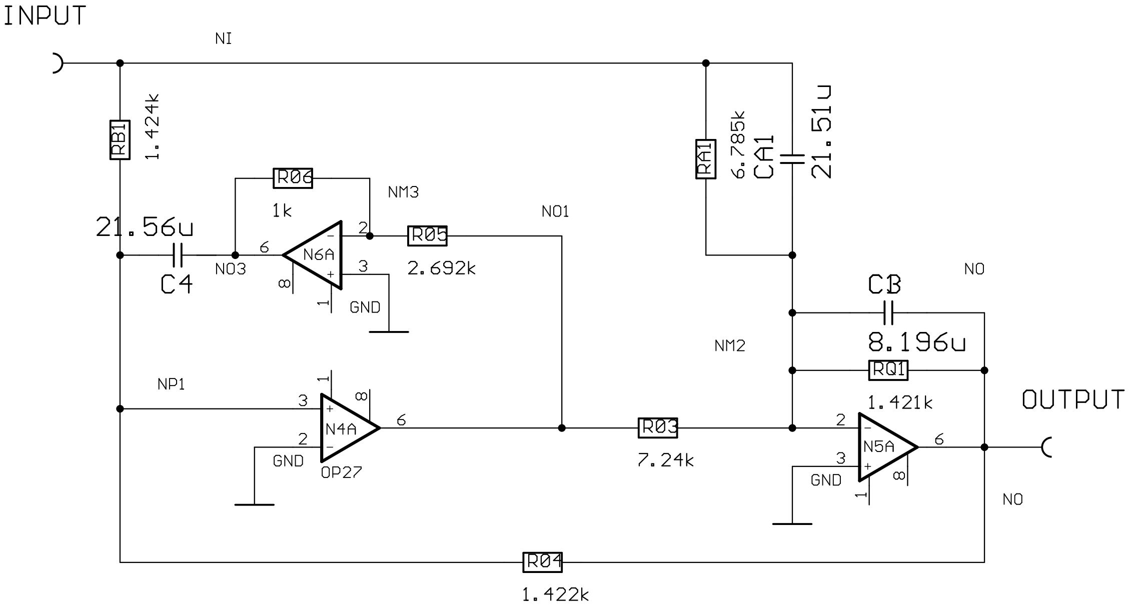


The circuit diagram of the compensation filter is shown in the following figure. It consists of two biquadratic sections realized as 'Akerberg–Mossberg' circuits. The component values were found with LISO's fit function.

M1XAKM



INPUT
NI
RB
4.25k
R0
1k
NM3
NO1
21.83u
C2
NO3
6
N3A
2
3
8
1
GND
R00
2.692k
RA
4.042k
CA
21.64u
C1
6.504u
NO
NM2
RQ
8.205k
OUTPUT
NP1
3
2
N1A
GND
OP27
6
R1
18.03k
N2A
GND
NO
R2
4.246k



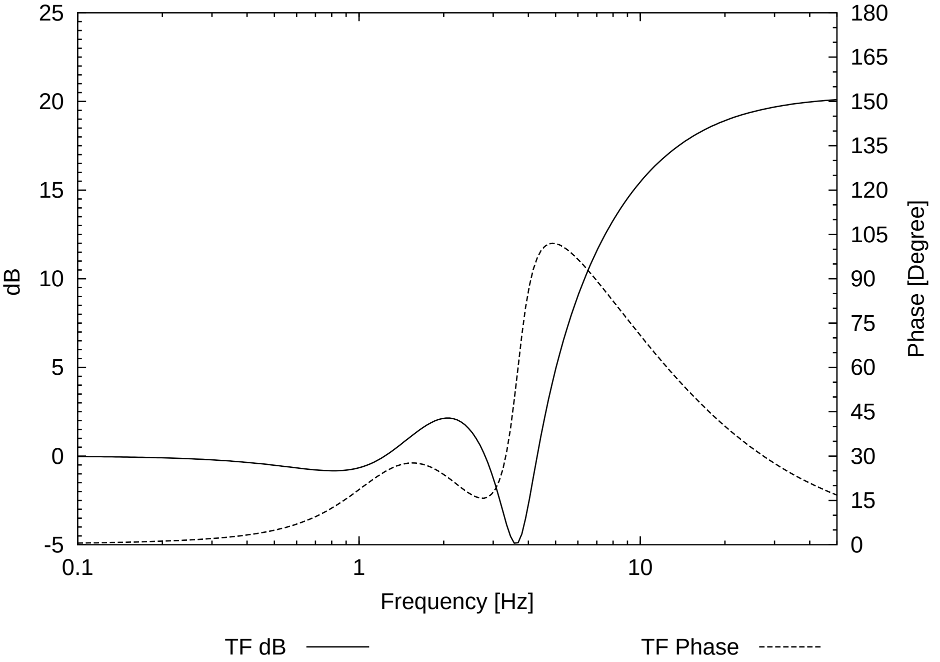
INPUT
NI
RB1
1.424k
R06
1k
NM3
NO1
21.56u
C4
NO3
6
N6A
2
3
8
1
GND
R05
2.692k
RA1
6.785k
CA1
21.51u
C3
8.196u
NO
NM2
RQ1
1.421k
OUTPUT
NP1
3
2
N4A
GND
OP27
6
R03
7.24k
N5A
GND
NO
R04
1.422k

13.2 Output Mirror Yaw coil (“m2xcoil”)

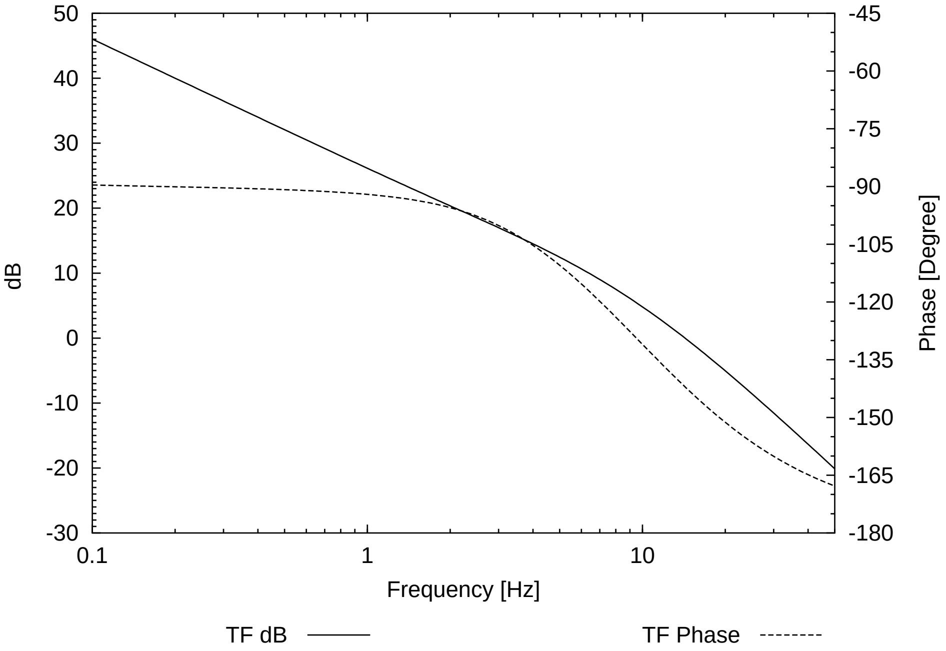
The filter is composed of two biquadratic sections as follows:

Section	Pole/Zero	f [Hz]	Q
1	zero	1.205	0.804
1	pole	2.3162	0.669
2	zero	3.6617	3.843
2	pole	3.777	(real)
2	pole	10.0	(real)

The following figure shows the transfer function of the compensation filter:

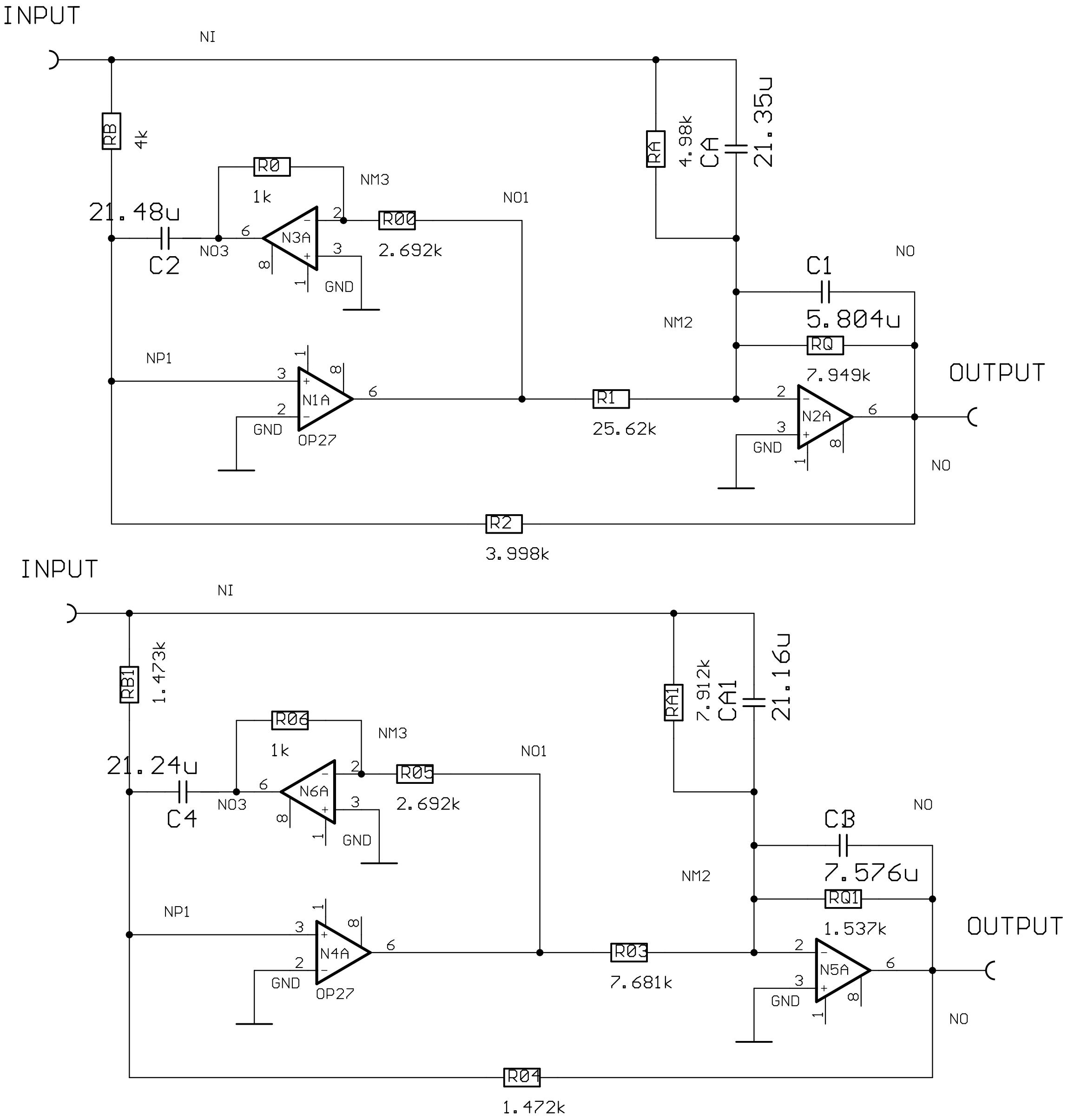


The following figure shows the predicted open-loop gain,without taking into account the additional phase delays by the antialiasing filter and the digital loop. A damped integrator active between DC and 3 Hz is included in the computed transfer function.



The circuit diagram of the compensation filter is shown in the following figure. It consists of two biquadratic sections realized as 'Akerberg–Mossberg' circuits. The component values were found with LISO's fit function.

M2XAKM



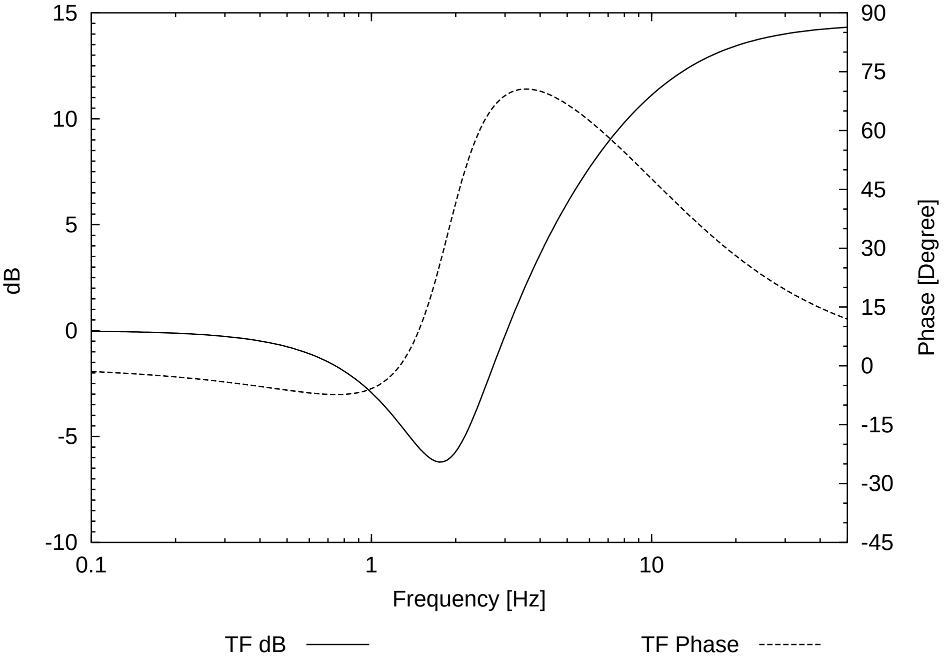
INPUT
NI
RB
4k
R0
1k
NM3
NO1
21.48u
C2
NO3
N3A
R00
2.692k
GND
RA
4.98k
CA
21.35u
C1
5.804u
NO
NM2
RQ
7.949k
NP1
N1A
OP27
R1
25.62k
N2A
OUTPUT
R2
3.998k
INPUT
NI
RB1
1.473k
R06
1k
NM3
NO1
21.24u
C4
NO3
N6A
R05
2.692k
GND
RA1
7.912k
CA1
21.16u
C3
7.576u
NO
NM2
RQ1
1.537k
NP1
N4A
OP27
R03
7.681k
N5A
OUTPUT
R04
1.472k

13.3 End Mirror Yaw Coil ("m3xcoil")

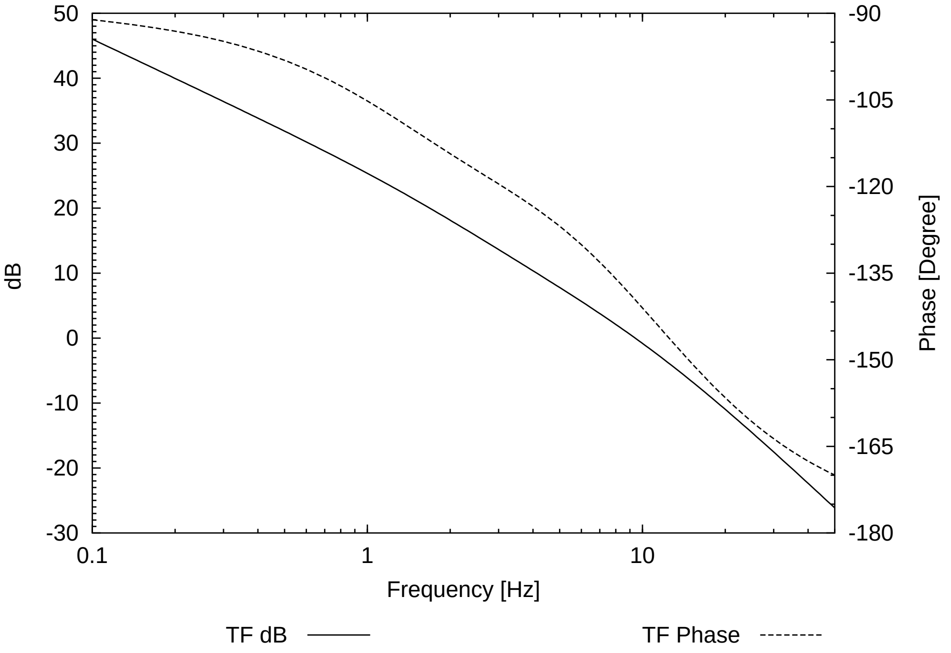
The filter is composed of one biquadratic section as follows:

Section	Pole/Zero	*f* [Hz]	Q
1	zero	1.8826	1.399
1	pole	1.8826	(real)
1	pole	10.0	(real)

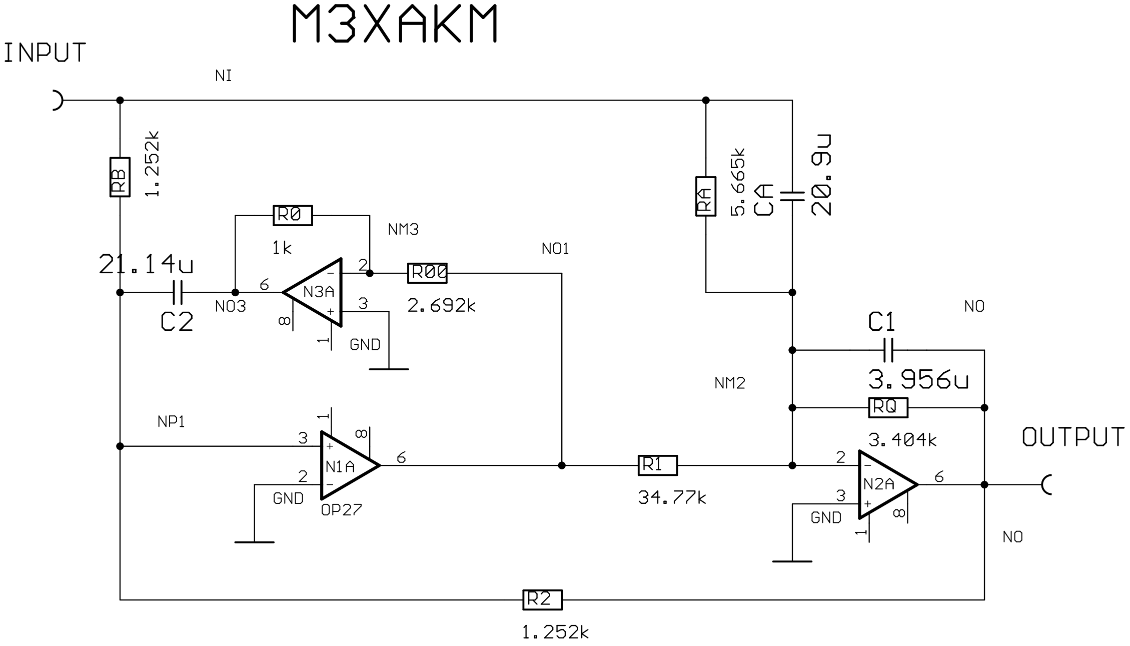
The following figure shows the transfer function of the compensation filter:



The following figure shows the predicted open-loop gain,without taking into account the additional phase delays by the antialiasing filter and the digital loop. A damped integrator active between DC and 3 Hz is included in the computed transfer function.



The circuit diagram of the compensation filter is shown in the following figure. It consists of one biquadratic section realized as 'Akerberg–Mossberg' circuit. The component values were found with LISO's fit function.

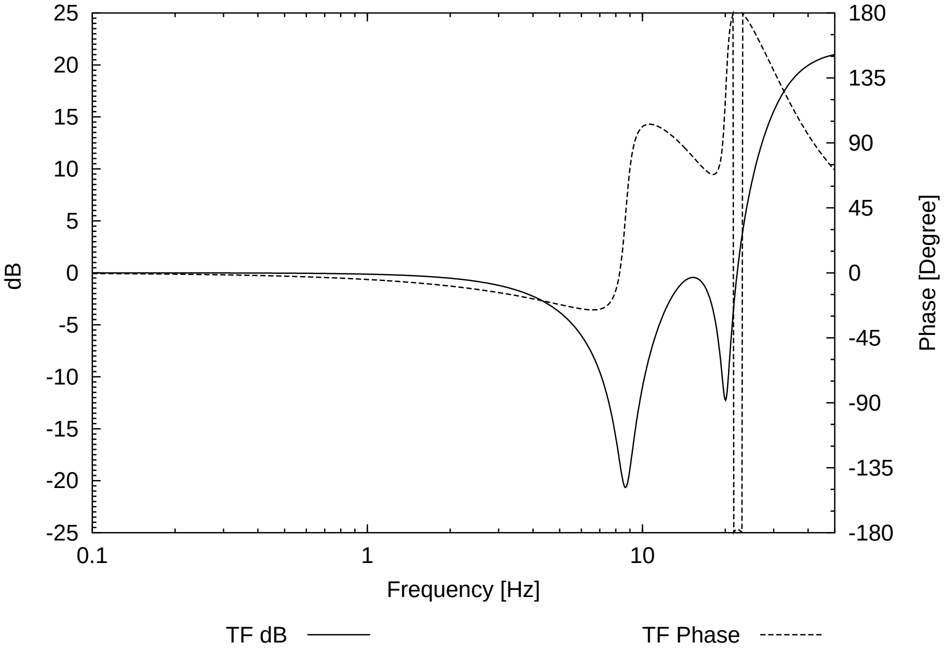


13.4 Input Mirror Pitch PZT ("m1ypzt")

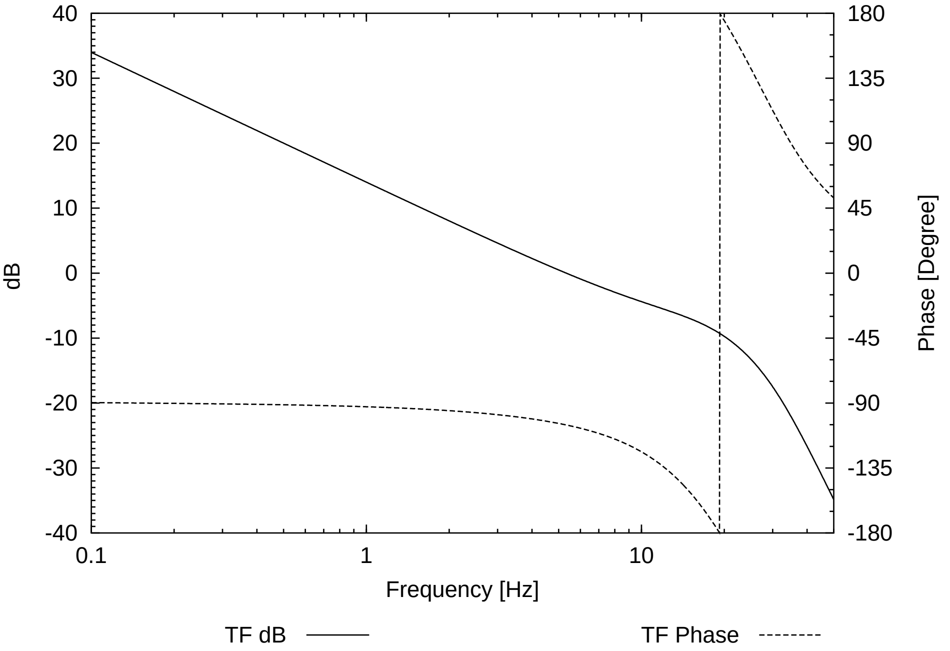
The filter is composed of two biquadratic sections as follows:

Section	Pole/Zero	f [Hz]	Q
1	zero	8.688	9.34
1	pole	20	0.8
2	zero	20.08	16.34
2	pole	30.0	1.0

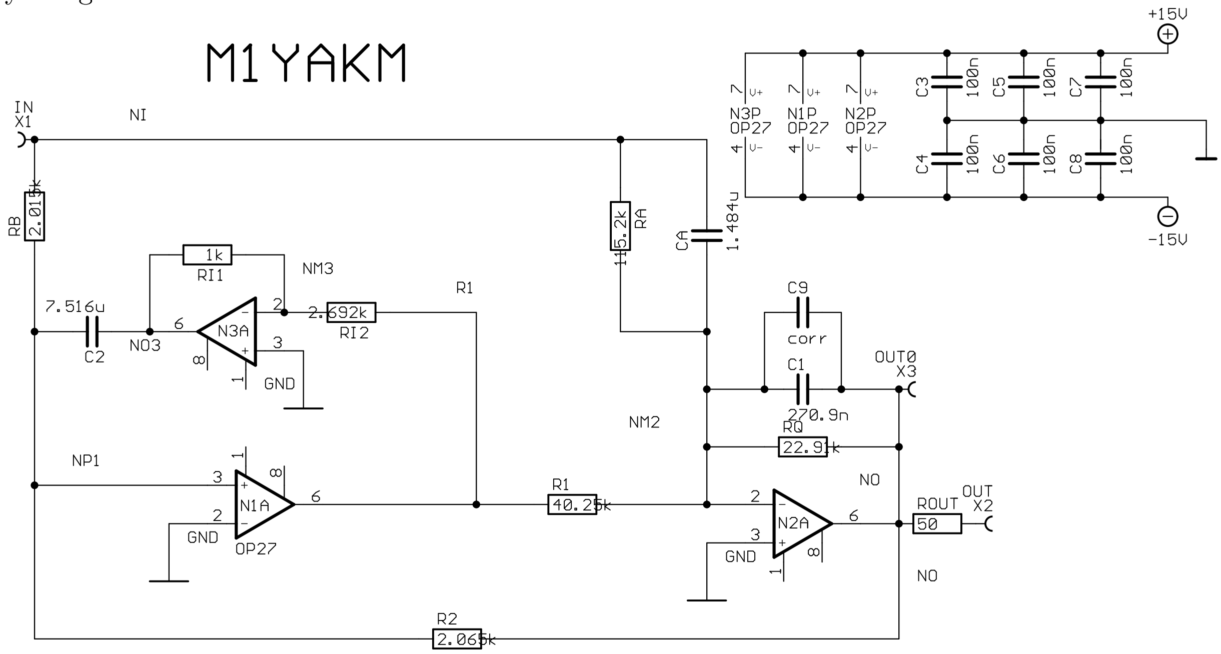
The following figure shows the transfer function of the compensation filter:



The following figure shows the predicted open-loop gain,without taking into account the additional phase delays by the antialiasing filter and the digital loop. A damped integrator active between DC and 3 Hz is included in the computed transfer function.



The circuit consists of two biquadratic sections realized as 'Akerberg–Mossberg' circuits and is shown below. The component values were found with LISO's fit function. They are given in Table 13.7 below.

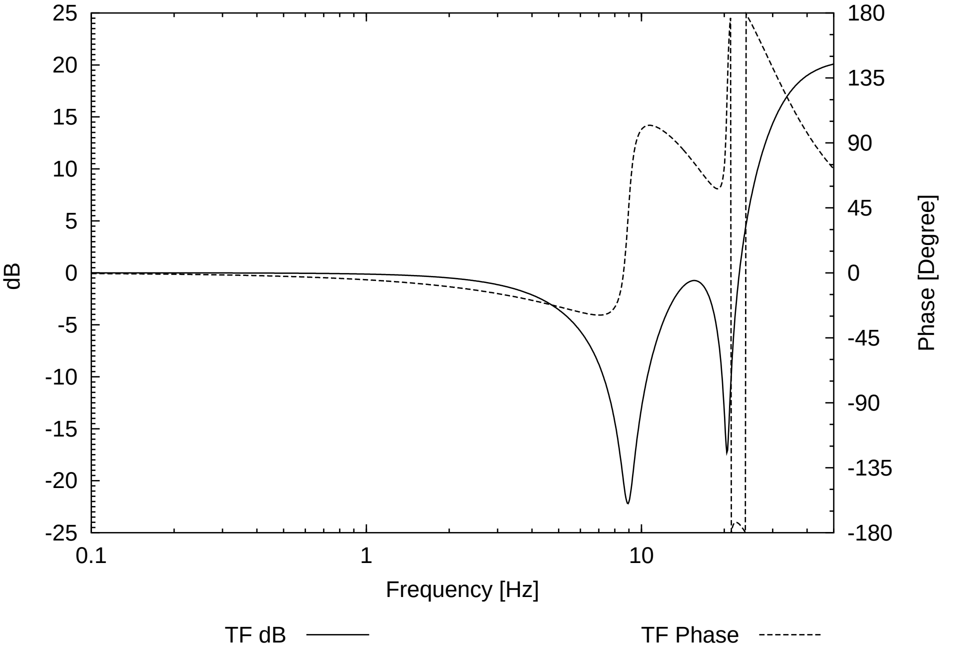


13.5 Output Mirror Pitch PZT ("m2ypzt")

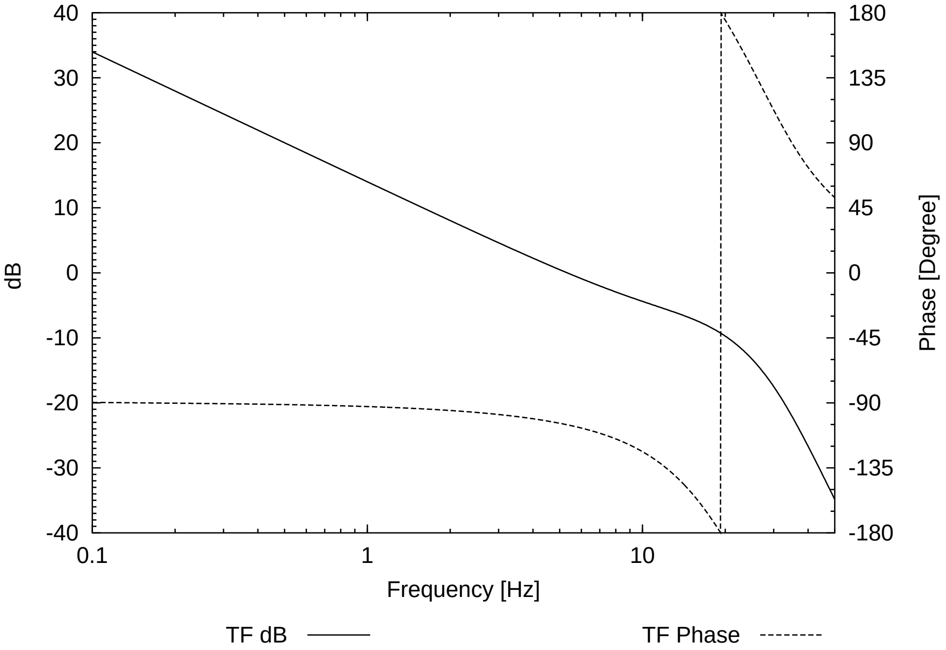
The filter is composed of two biquadratic sections as follows:

Section	Pole/Zero	*f* [Hz]	Q
1	zero	8.934858	11.15256
1	pole	20	0.8
2	zero	zero 20.48184	28.21300
2	pole	30.0	1.0

The following figure shows the transfer function of the compensation filter:



The following figure shows the predicted open-loop gain,without taking into account the additional phase delays by the antialiasing filter and the digital loop. A damped integrator active between DC and 3 Hz is included in the computed transfer function.



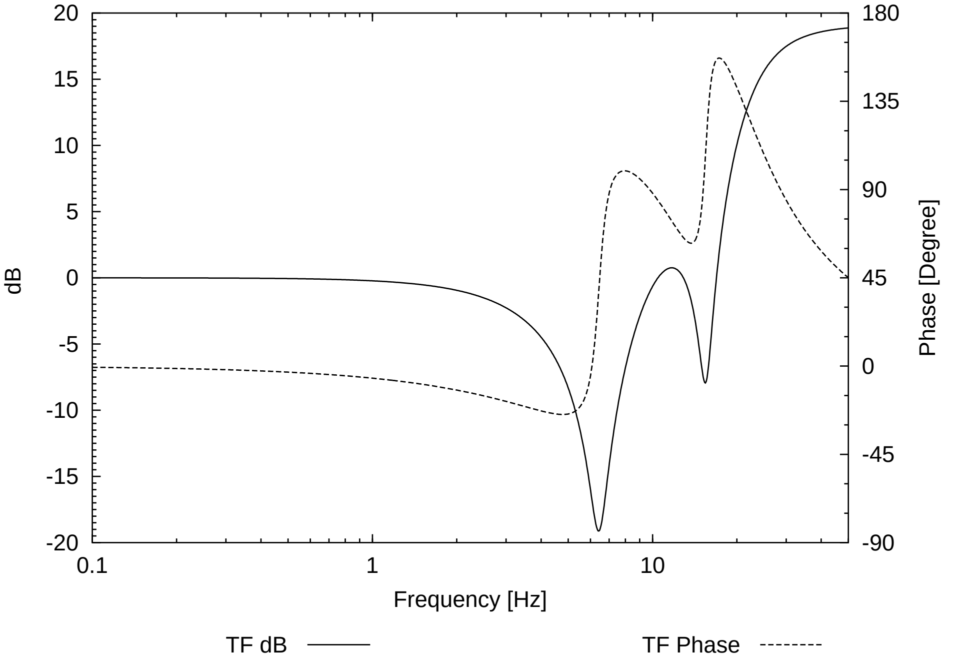
The circuit consists of two biquadratic sections realized as ‘Akerberg–Mossberg’ circuits and is the same is for the channel `m1ypzt`. The component values were found with LISO’s fit function. They are given in Table 13.7 below.

13.6 End Mirror Pitch PZT ("m3ypzt")

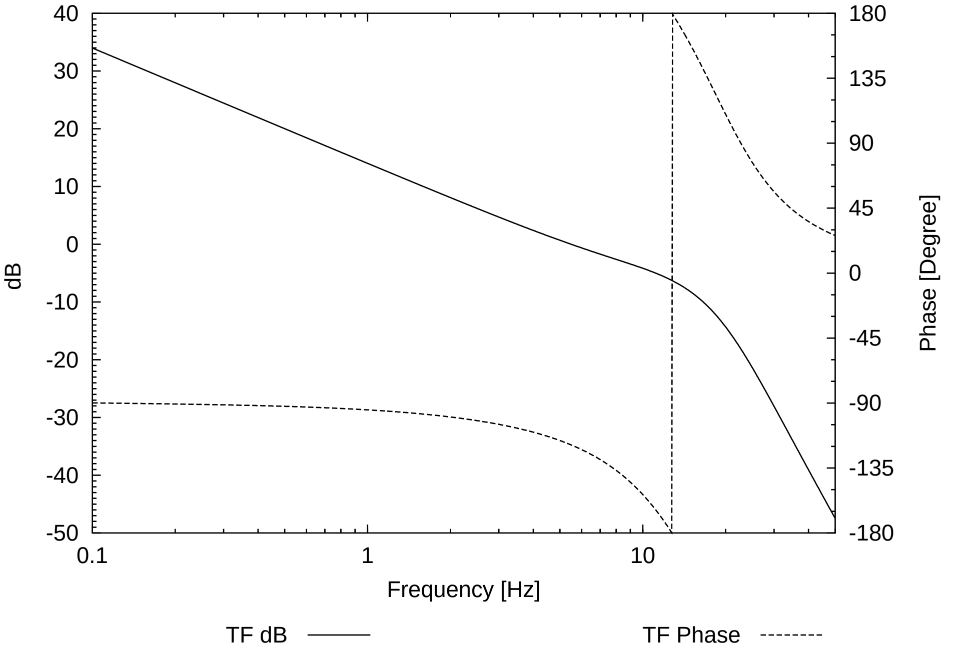
The filter is composed of two biquadratic sections as follows:

Section	Pole/Zero	*f* [Hz]	Q
1	zero	6.436784	8.052757
1	pole	15	0.8
2	zero	15.49933	10.58777
2	pole	20.0	1.0

The following figure shows the transfer function of the compensation filter:



The following figure shows the predicted open-loop gain,without taking into account the additional phase delays by the antialiasing filter and the digital loop. A damped integrator active between DC and 3 Hz is included in the computed transfer function.



The circuit consists of two biquadratic sections realized as 'Akerberg–Mossberg' circuits and is the same is for the channel `m1ypzt`. The component values were found with LISO's fit function. They are given in Table 13.7 below.

13.7 Component values for y channels

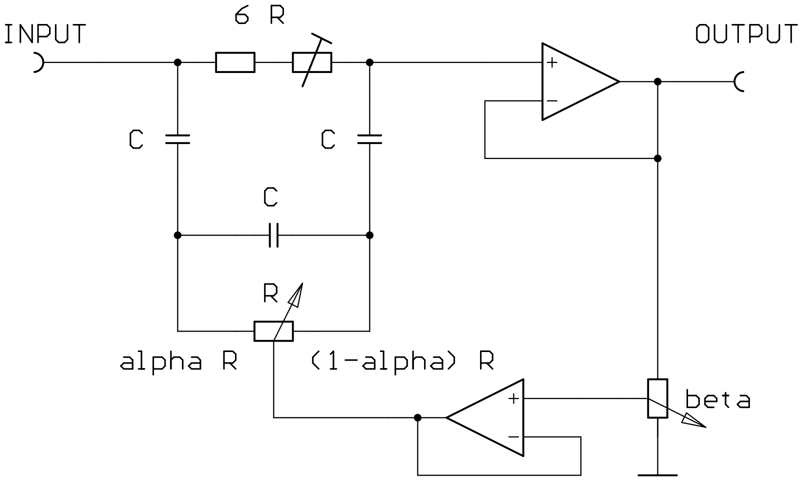
The component values for the 6 Akerberg-Mossberg sections of the y filters are given below. All resistors "`RI1`" are 1 kΩ, all resistors "`RI2`" are 2.7 kΩ. The precise frequency of the zero (notch) can be trimmed with "`RI1`", and its Q with the resistor "`RA`". These two adjustments are coupled and must hence be done iteratively. In the circuits that I built, the frequency was adjusted, but not the Q.

```
M1 Section 1:
C1 270.88486n
C2 7.516u
CA 1.484u
R1 40.246121k
R2 2.0651912k
RA 115.16129k
RB 2.0154407k
RQ 22.9111k

M1 Section 2:
C1 608.22835n
C2 7.487u
CA 1.484u
R1 25.717407k
R2 642.26046
```

```
RA 86.316015k
RB 592.61269
RQ 8.4654329k

M2 Section 1:
C1 287.81485n
C2 7.487u
CA 1.492u
R1 39.571432k
R2 1.9842416k
RA 132.7881k
RB 1.9346652k
RQ 21.565715k

M2 Section 2:
C1 635.88476n
C2 7.423u
CA 1.486u
R1 23.92768k
R2 665.65775
RA 145.10147k
RB 616.22654
RQ 8.1007099k

M3 Section 1:
C1 268.02694n
C2 7.414u
CA 1.486u
R1 42.673035k
R2 3.5556843k
RA 133.86264k
RB 3.5060136k
RQ 30.976821k

M3 Section 2:
C1 838.28844n
C2 7.343u
CA 1.478u
R1 28.598303k
R2 964.7165
RA 72.07211k
RB 915.47871
RQ 9.2504576k
```

13.8 Notch–Filter

In order to suppress high-Q suspension resonances, two notch filters were also built and included in the loops. Their notch frequencies are around 55 Hz and 145 Hz, respectively. I have used the "bridged differentiator" circuit given in [Horowitz-Hill]:



The circuit needs three *identical* capactitors, which can be found be measuring and selecting. I have used 0.15 μF and 0.47 μF for the two filters for the modecleaner. The notch frequency can be adjusted with R_α in a relatively wide range. The upper resistor needs to be exactly $6R$ and is hence also made adjustable ($R = 10\,\text{k}\Omega$, upper resistor is $54\,\text{k}\Omega + 10\,\text{k}\Omega$ variable). The trimmer β adjusts the Q of the filter. $\beta = 0$ corrresponds to a passive circuit where the wiper of R_α is grounded. The transfer function is given by

$$H(s) = \frac{(1 + 2\frac{s}{\omega_0})\left[-1 + 3(\alpha^2 - \alpha)\left(\frac{s}{\omega_0}\right)^2\right]}{-1 + (6\beta - 8)\frac{s}{\omega_0} + (6\alpha\beta + 6\beta + 3\alpha^2 - 9\alpha - 6)\left(\frac{s}{\omega_0}\right)^2 + 6(\alpha^2 - \alpha)\left(\frac{s}{\omega_0}\right)^3} \tag{81}$$

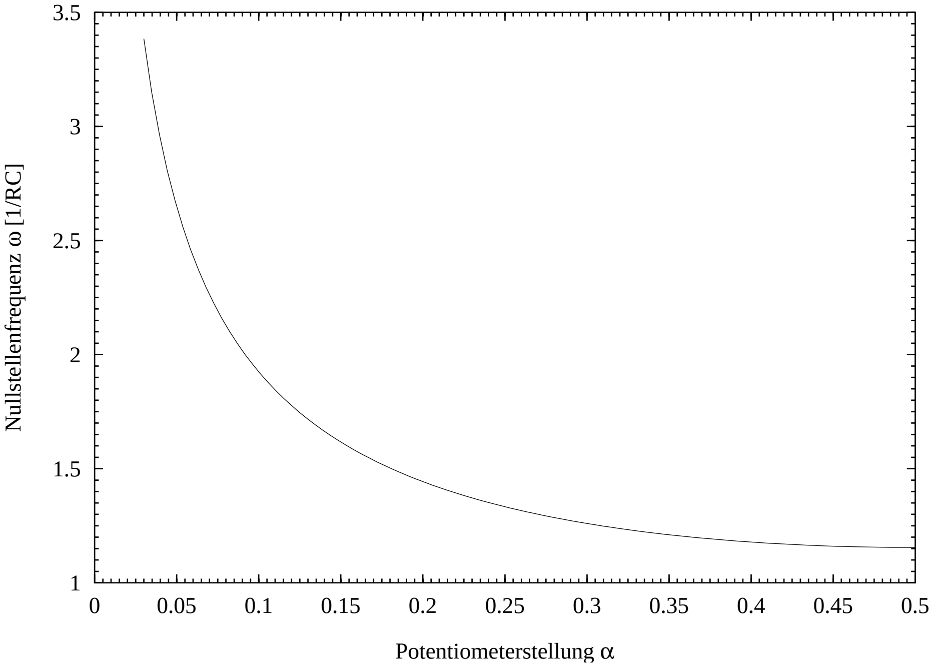
with

$$\omega_0 = \frac{1}{RC}\ . \tag{82}$$

The notch frequency is at

$$\omega_{\text{Notch}} = \frac{\omega_0}{\sqrt{3\alpha(1-\alpha)}}\ . \tag{83}$$

This function is shown here:



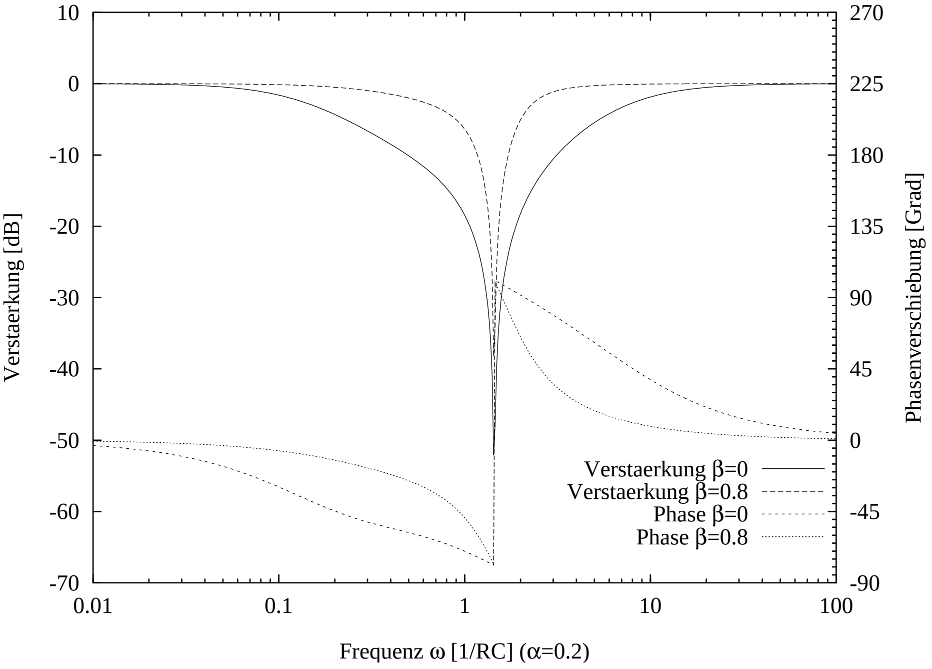
A useful initial estimate is

$$\alpha = 0.2\ . \tag{84}$$

Then we have

$$\omega_{\text{Notch}} = \frac{1.44}{RC} \quad \text{oder} \quad RC = \frac{0.23}{f_{\text{Notch}}}\ . \tag{85}$$

The next figure shows the computed transfer function for $\beta = 0$ and $\beta = 0.8$ (with $\alpha = 0.2$ in both cases). This was *not* computed with LISO, but analytically assuming ideal op-amps. In practice β may not be too high because otherwise the depth of the notch is degraded.


10
0
-10
-20
-30
-40
-50
-60
-70
270
225
180
135
90
45
0
-45
-90
0.01
0.1
1
10
100
Verstaerkung [dB]
Phasenverschiebung [Grad]
Frequenz ω [1/RC] (α=0.2)
Verstaerkung β=0
Verstaerkung β=0.8
Phase β=0
Phase β=0.8